# TMS320F28004x Piccolo Microcontrollers

# Technical Reference Manual

Literature Number: SPRUI33A
November 2015–Revised December 2017

# Contents

# List of Figures

# List of Tables

# Read This First

## About This Manual

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

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

## Notational Conventions

This document uses the following conventions.

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

## Glossary

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

## Related Documentation From Texas Instruments

For a complete listing of related documentation and development-support tools for these devices, visit the Texas Instruments website at http://www.ti.com. Additionally, the *TMS320C28x CPU and Instruction Set Reference Guide* (SPRU430) and *TMS320C28x Floating Point Unit and Instruction Set Reference Guide* (SPRUEO2) must be used in conjunction with this TRM.

## Trademarks

C2000, LaunchPad, Code Composer Studio are trademarks of Texas Instruments.

# C2000Ware Quick Start Guide

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

C2000Ware requires:

- CCS v6.2.0 or newer
- C2000 Compiler v16.9.0 or newer

**Topic** **Page**

## 1.1 Package Structure

The C2000Ware softwareblah package is organized into the following directory structure: Checking to see if this changes in the pOtenza doc.

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

| Directory Name | Description |
|---|---|
| .metadata | Contains the C2000Ware resource explorer standalone GUI files. Do not modify. |
| boards | Contains the hardware design schematics, BOM, gerber files, and documentation for C2000 controlCARDS, controlSTICKs, Experimenter Kit, and LaunchPads |
| device_support | Contains all device-specific support files, bit field headers, bit field device peripheral examples (including LaunchPad™ demos), 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. |
| uninstallers | Contains the C2000Ware uninstaller. |
| utilities | Contains development utility applications such as flash programmers, windows drivers, and third party software. |

### *1.1.1 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 Launchpads. 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.

### *1.1.2 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.

### *1.1.3 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.

## 1.2 C2000Ware GUI

C2000Ware provides a graphical user interface (GUI) for intuitive navigation of software, libraries, user's guides, and other package content. Within Code Composer Studio™ (CCS) v6.2.0 and newer, go to "View->Resource Explorer" to locate the C2000Ware GUI explorer.

C2000Ware uses the new online Resource Explorer, which includes many updated features. This includes full package navigation on the web or in CCS without requiring installation of C2000Ware. Additionally, on the web there is the ability to import to CCS Cloud and download individual files or examples.

View C2000Ware Resource Explorer on the web: dev.ti.com/tirex/#/

The C2000Ware Standalone GUI (without requiring CCS) is currently in development.

## 1.3 Updating C2000Ware

Within the C2000Ware installation directory there is an update checking application, “C2000WareUpdater.exe”, which checks for a newer version of C2000Ware. The updater (currently Windows only) will periodically check automatically in the background for any newly available C2000Ware packages. A notification will appear detailing any updates found. When the updater is run manually, a window opens to display either that the current version installed is the latest version or that a newer version is available for download. If a new version is available, the option is given to download and install the new version. The application will then proceed to download the latest C2000Ware installer to the specified download directory. Upon completing the download, the installer will automatically run and continue with the standard installer steps.

## 1.4 Code Composer Studio

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

http://www.ti.com/ccstudio

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

# C28x Processor

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

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

- *Accelerators: Enhancing the Capabilities of the C2000 MCU Family Technical Brief*
- *TMS320C28x FPU Primer*

**Topic** **Page**

## 2.1 Overview

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

The features include:

- CPU - modified Harvard architecture and circular addressing. The modified Harvard architecture of the CPU enables instruction and data fetches to be performed in parallel. The CPU can read instructions and data while it writes data simultaneously to maintain the single-cycle instruction operation across the pipeline. The CPU does this over six separate address/data buses
- RISC - single-cycle instruction execution, register-to-register operations, and modified Harvard architecture.
- Microcontroller - ease of use through an intuitive instruction set, byte packing and unpacking, and bit manipulation.

For more information on CPU architecture and instruction set, see the *TMS320C28x CPU and Instruction Set Reference Guide*. For more information on the C28x Floating Point Unit (FPU), see the *TMS320C28x Extended Instruction Sets Technical Reference Guide* ). All of the features of the C28x documented in the *TMS320C28x DSP CPU and Instruction Set Reference Guide* apply to the C28x+VCU. All features documented in the *TMS320C28x Floating Point Unit and Instruction Set Reference Guide* apply to the C28x+FPU+VCU. A brief overview of the FPU, TMU, and VCU-Type 0 is provided here.

An overview of the VCU instructions can be found in the *TMS320C28x Extended Instruction Sets Technical Reference Manual*.

### 2.1.1 Floating-Point Unit

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

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

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

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

### 2.1.2 Trigonometric Math Unit

The TMU extends the capabilities of a C28x+FPU by adding instructions and leveraging existing FPU instructions to speed up the execution of common trigonometric and arithmetic operations listed in Table 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 their operations.

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

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

- **Viterbi decoding**

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

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

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

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

- **Complex math**

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

  – Fast fourier transform (FFT)

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

  – Complex filters

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

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

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

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

Throughout this document the following notations are used:

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

# System Control

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

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

- *A Technical Introduction to the TMS320F28004x Microcontroller*
- *The TMS320F28004x MCU: A Comparison to the TMS320F2806x and TMS320F2803x MCUs*

**Topic** **Page**

## 3.1 Introduction

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

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

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

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

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

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

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

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

## 3.2 Power Management

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

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

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

6. EDIS

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

## 3.3 Device Identification and Configuration Registers

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

All of the device information is part of the DEV_CFG_REGS space. The identification registers are PARTIDL, PARTIDH, and REVID.

## 3.4 Resets

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

### *3.4.1 Reset Sources*

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

**Table 3-2. Reset Signals**

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

The resets can be divided into two groups:

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

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

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

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

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

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

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

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

### 3.4.3 Power-On Reset (POR)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## 3.5 Peripheral Interrupts

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

### 3.5.1 Interrupt Concepts

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

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

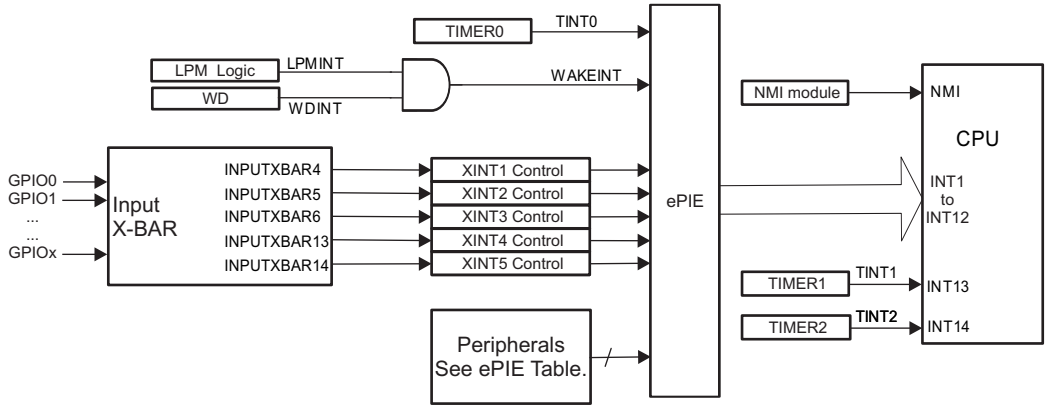
### 3.5.2 Interrupt Architecture

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

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

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

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




#### 3.5.2.1 Peripheral Stage

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

#### 3.5.2.2 PIE Stage

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

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

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

#### 3.5.2.3 CPU Stage

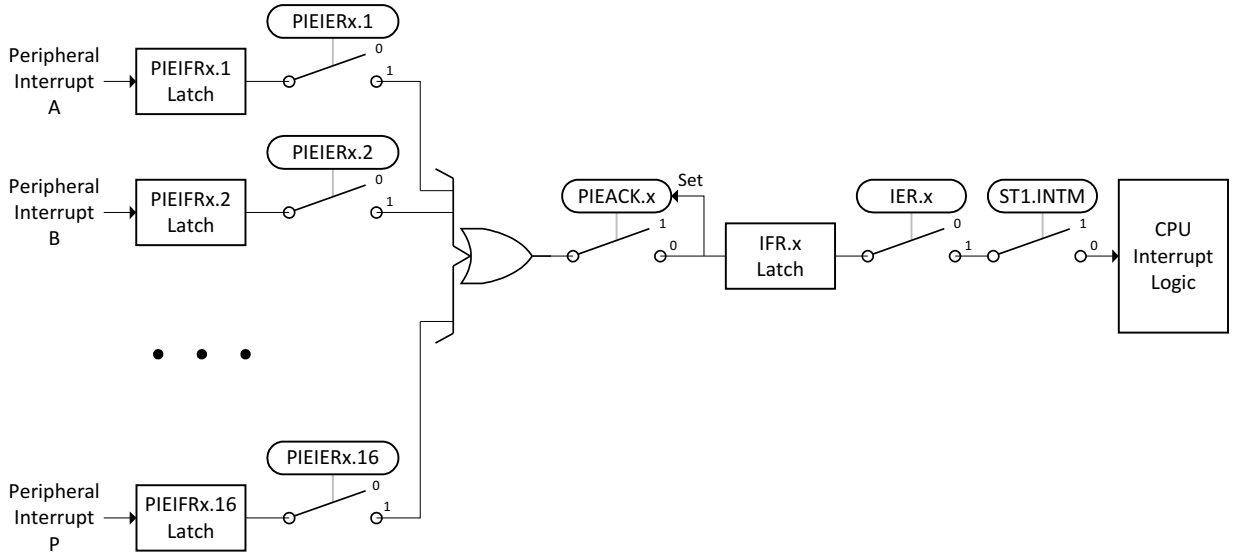
Like the PIE, the CPU provides flag and enable register bits for each of its interrupts. There is one enable register (IER) and one flag register (IFR), both of which are internal CPU registers. There is also a global interrupt mask, which is controlled by the INTM bit in the ST1 register. This mask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

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

### 3.5.3 Interrupt Entry Sequence

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

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




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

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

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

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

### *3.5.4 Configuring and Using Interrupts*

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

#### 3.5.4.1 Enabling Interrupts

To enable a peripheral interrupt, perform the following steps:

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

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

#### 3.5.4.2 Handling Interrupts

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

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

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

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

#### 3.5.4.3 Disabling Interrupts

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

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

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

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

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

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

#### 3.5.4.4 Nesting Interrupts

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

#### 3.5.4.5 Vector Address Validity Check

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

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

### 3.5.5 PIE Channel Mapping

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

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

| | INTx.1 | INTx.2 | INTx.3 | INTx.4 | INTx.5 | INTx.6 | INTx.7 | INTx.8 | INTx.9 | INTx.10 | INTx.11 | INTx.12 | INTx.13 | INTx.14 | INTx.15 | INTx.16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INT1.y | ADCA1 | ADCB1 | ADCC1 | XINT1 | XINT2 | - | TIMER0 | WAKE /WDOG | - | - | - | - | - | - | - | - |
| INT2.y | EPWM1_ TZ | EPWM2_ TZ | EPWM3_ TZ | EPWM4_ TZ | EPWM5_ TZ | EPWM6_ TZ | EPWM7_ TZ | EPWM8_ TZ | - | - | - | - | - | - | - | - |
| INT3.y | EPWM1 | EPWM2 | EPWM3 | EPWM4 | EPWM5 | EPWM6 | EPWM7 | EPWM8 | - | - | - | - | - | - | - | - |
| INT4.y | ECAP1 | ECAP2 | ECAP3 | ECAP4 | ECAP5 | ECAP6 | ECAP7 | - | - | - | - | - | - | ECAP6_ HRCAL | ECAP7_ HRCAL | - |
| INT5.y | EQEP1 | EQEP2 | - | - | - | - | - | - | SDFM1 | - | - | - | SDFM1DR1 | SDFM1DR2 | SDFM1DR3 | SDFM1DR4 |
| INT6.y | SPIA_RX | SPIA_TX | SPIB_RX | SPIB_TX | - | - | - | - | - | - | - | - | - | - | - | - |
| INT7.y | DMA_CH1 | DMA_CH2 | DMA_CH3 | DMA_CH4 | DMA_CH5 | DMA_CH6 | - | - | - | - | FSITX_ INT1 | FSITX_ INT2 | FSIRX_ INT1 | FSIRX_ INT2 | CLAPROMC RC | DCC |
| INT8.y | I2CA | I2CA_ FIFO | - | - | - | - | - | - | LINA_0 | LINA_1 | - | - | PMBUSA | - | - | - |
| INT9.y | SCIA_RX | SCIA_TX | SCIB_RX | SCIB_TX | CANA_0 | CANA_1 | CANB_0 | CANB_1 | - | - | - | - | - | - | - | - |
| INT10.y | ADCA_ EVT | ADCA2 | ADCA3 | ADCA4 | ADCB_EVT | ADCB2 | ADCB3 | ADCB4 | ADCC_EVT | ADCC2 | ADCC3 | ADCC4 | - | - | - | - |
| INT11.y | CLA1_1 | CLA1_2 | CLA1_3 | CLA1_4 | CLA1_5 | CLA1_6 | CLA1_7 | CLA1_8 | - | - | - | - | - | - | - | - |
| INT12.y | XINT3 | XINT4 | XINT5 | - | - | - | FPU_OVER FLOW | FPU_ UNDER FLOW | - | RAM_COR RECTABLE _ERROR | FLASH_CO RRECTABL E_ERROR | RAM_ACCE SS_VIOLAT ION | SYS_PLL_ SLIP | - | CLA OVER FLOW | CLA UNDER FLOW |

Note: Cells marked "-" are Reserved

### 3.5.6 Vector Tables

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

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

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

Table 3-5 shows the pie vector table.

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

| Name | Vector ID | Address | Size (x16) | Description | Core priority | ePIE group Priority |
|---|---|---|---|---|---|---|
| **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 | Reserved | 5 | 6 |
| INT1.7 | 38 | 0x0000 0D4C | 2 | TIMER0 interrupt | 5 | 7 |
| INT1.8 | 39 | 0x0000 0D4E | 2 | WAKE interrupt | 5 | 8 |
| INT1.9 | 128 | 0x0000 0E00 | 2 | Reserved | 5 | 9 |
| INT1.10 | 129 | 0x0000 0E02 | 2 | Reserved | 5 | 10 |
| INT1.11 | 130 | 0x0000 0E04 | 2 | Reserved | 5 | 11 |
| INT1.12 | 131 | 0x0000 0E06 | 2 | Reserved | 5 | 12 |
| INT1.13 | 132 | 0x0000 0E08 | 2 | Reserved | 5 | 13 |
| INT1.14 | 133 | 0x0000 0E0A | 2 | Reserved | 5 | 14 |
| INT1.15 | 134 | 0x0000 0E0C | 2 | Reserved | 5 | 15 |
| INT1.16 | 135 | 0x0000 0E0E | 2 | Reserved | 5 | 16 (Lowest) |
| **PIE Group 2 Vectors - Muxed into CPU INT2** | | | | | | |
| INT2.1 | 40 | 0x0000 0D50 | 2 | EPWM1 trip zone interrupt | 6 | 1 (Highest) |
| INT2.2 | 41 | 0x0000 0D52 | 2 | EPWM2 trip zone interrupt | 6 | 2 |
| INT2.3 | 42 | 0x0000 0D54 | 2 | EPWM3 trip zone interrupt | 6 | 3 |
| INT2.4 | 43 | 0x0000 0D56 | 2 | EPWM4 trip zone interrupt | 6 | 4 |
| INT2.5 | 44 | 0x0000 0D58 | 2 | EPWM5 trip zone interrupt | 6 | 5 |
| INT2.6 | 45 | 0x0000 0D5A | 2 | EPWM6 trip zone interrupt | 6 | 6 |
| INT2.7 | 46 | 0x0000 0D5C | 2 | EPWM7 trip zone interrupt | 6 | 7 |
| INT2.8 | 47 | 0x0000 0D5E | 2 | EPWM8 trip zone interrupt | 6 | 8 |
| INT2.9 | 136 | 0x0000 0E10 | 2 | Reserved | 6 | 9 |
| INT2.10 | 137 | 0x0000 0E12 | 2 | Reserved | 6 | 10 |
| INT2.11 | 138 | 0x0000 0E14 | 2 | Reserved | 6 | 11 |
| INT2.12 | 139 | 0x0000 0E16 | 2 | Reserved | 6 | 12 |
| INT2.13 | 140 | 0x0000 0E18 | 2 | Reserved | 6 | 13 |
| INT2.14 | 141 | 0x0000 0E1A | 2 | Reserved | 6 | 14 |
| INT2.15 | 142 | 0x0000 0E1C | 2 | Reserved | 6 | 15 |
| INT2.16 | 143 | 0x0000 0E1E | 2 | Reserved | 6 | 16 (Lowest) |
| **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 |

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

| Name | Vector ID | Address | Size (x16) | Description | Core priority | ePIE group Priority |
|---|---|---|---|---|---|---|
| INT3.6 | 53 | 0x0000 0D6A | 2 | EPWM6 interrupt | 7 | 6 |
| INT3.7 | 54 | 0x0000 0D6C | 2 | EPWM7 interrupt | 7 | 7 |
| INT3.8 | 55 | 0x0000 0D6E | 2 | EPWM8 interrupt | 7 | 8 |
| INT3.9 | 144 | 0x0000 0E20 | 2 | Reserved | 7 | 9 |
| INT3.10 | 145 | 0x0000 0E22 | 2 | Reserved | 7 | 10 |
| INT3.11 | 146 | 0x0000 0E24 | 2 | Reserved | 7 | 11 |
| INT3.12 | 147 | 0x0000 0E26 | 2 | Reserved | 7 | 12 |
| INT3.13 | 148 | 0x0000 0E28 | 2 | Reserved | 7 | 13 |
| INT3.14 | 149 | 0x0000 0E2A | 2 | Reserved | 7 | 14 |
| INT3.15 | 150 | 0x0000 0E2C | 2 | Reserved | 7 | 15 |
| INT3.16 | 151 | 0x0000 0E2E | 2 | Reserved | 7 | 16 (Lowest) |
| **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 | ECAP7 interrupt | 8 | 7 |
| INT4.8 | 63 | 0x0000 0D7E | 2 | Reserved | 8 | 8 |
| INT4.9 | 152 | 0x0000 0E30 | 2 | Reserved | 8 | 9 |
| INT4.10 | 153 | 0x0000 0E32 | 2 | Reserved | 8 | 10 |
| INT4.11 | 154 | 0x0000 0E34 | 2 | Reserved | 8 | 11 |
| INT4.12 | 155 | 0x0000 0E36 | 2 | Reserved | 8 | 12 |
| INT4.13 | 156 | 0x0000 0E38 | 2 | Reserved | 8 | 13 |
| INT4.14 | 157 | 0x0000 0E3A | 2 | ECAP6 HR calibration interrupt | 8 | 14 |
| INT4.15 | 158 | 0x0000 0E3C | 2 | ECAP7 HR calibration interrupt | 8 | 15 |
| INT4.16 | 159 | 0x0000 0E3E | 2 | Reserved | 8 | 16 (Lowest) |
| **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 | Reserved | 9 | 3 |
| INT5.4 | 67 | 0x0000 0D86 | 2 | Reserved | 9 | 4 |
| INT5.5 | 68 | 0x0000 0D88 | 2 | Reserved | 9 | 5 |
| INT5.6 | 69 | 0x0000 0D8A | 2 | Reserved | 9 | 6 |
| INT5.7 | 70 | 0x0000 0D8C | 2 | Reserved | 9 | 7 |
| INT5.8 | 71 | 0x0000 0D8E | 2 | Reserved | 9 | 8 |
| INT5.9 | 160 | 0x0000 0E40 | 2 | SDFM1 interrupt | 9 | 9 |
| INT5.10 | 161 | 0x0000 0E42 | 2 | Reserved | 9 | 10 |
| INT5.11 | 162 | 0x0000 0E44 | 2 | Reserved | 9 | 11 |
| INT5.12 | 163 | 0x0000 0E46 | 2 | Reserved | 9 | 12 |
| INT5.13 | 164 | 0x0000 0E48 | 2 | SDFM1 DR interrupt 1 | 9 | 13 |

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

| Name | Vector ID | Address | Size (x16) | Description | Core priority | ePIE group Priority |
|---|---|---|---|---|---|---|
| INT5.14 | 165 | 0x0000 0E4A | 2 | SDFM1 DR interrupt 2 | 9 | 14 |
| INT5.15 | 166 | 0x0000 0E4C | 2 | SDFM1 DR interrupt 3 | 9 | 15 |
| INT5.16 | 167 | 0x0000 0E4E | 2 | SDFM1 DR interrupt 4 | 9 | 16 (Lowest) |
| **PIE Group 6 Vectors - Muxed into CPU INT6** | | | | | | |
| INT6.1 | 72 | 0x0000 0D90 | 2 | SPIA_RX interrupt | 10 | 1 (Highest) |
| INT6.2 | 73 | 0x0000 0D92 | 2 | SPIA_TX interrupt | 10 | 2 |
| INT6.3 | 74 | 0x0000 0D94 | 2 | SPIB_RX interrupt | 10 | 3 |
| INT6.4 | 75 | 0x0000 0D96 | 2 | SPIB_TX interrupt | 10 | 4 |
| INT6.5 | 76 | 0x0000 0D98 | 2 | Reserved | 10 | 5 |
| INT6.6 | 77 | 0x0000 0D9A | 2 | Reserved | 10 | 6 |
| INT6.7 | 78 | 0x0000 0D9C | 2 | Reserved | 10 | 7 |
| INT6.8 | 79 | 0x0000 0D9E | 2 | Reserved | 10 | 8 |
| INT6.9 | 168 | 0x0000 0E50 | 2 | Reserved | 10 | 9 |
| INT6.10 | 169 | 0x0000 0E52 | 2 | Reserved | 10 | 10 |
| INT6.11 | 170 | 0x0000 0E54 | 2 | Reserved | 10 | 11 |
| INT6.12 | 171 | 0x0000 0E56 | 2 | Reserved | 10 | 12 |
| INT6.13 | 172 | 0x0000 0E58 | 2 | Reserved | 10 | 13 |
| INT6.14 | 173 | 0x0000 0E5A | 2 | Reserved | 10 | 14 |
| INT6.15 | 174 | 0x0000 0E5C | 2 | Reserved | 10 | 15 |
| INT6.16 | 175 | 0x0000 0E5E | 2 | Reserved | 10 | 16 (Lowest) |
| **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 | FSITX_INT1 | 11 | 11 |
| INT7.12 | 179 | 0x0000 0E66 | 2 | FSITX_INT2 | 11 | 12 |
| INT7.13 | 180 | 0x0000 0E68 | 2 | FSIRX_INT1 | 11 | 13 |
| INT7.14 | 181 | 0x0000 0E6A | 2 | FSIRX_INT2 | 11 | 14 |
| INT7.15 | 182 | 0x0000 0E6C | 2 | CLAPROMCRC interrupt | 11 | 15 |
| INT7.16 | 183 | 0x0000 0E6E | 2 | Reserved | 11 | 16 (Lowest) |

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

| Name | Vector ID | Address | Size (x16) | Description | Core priority | ePIE group Priority |
|---|---|---|---|---|---|---|
| **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 | Reserved | 12 | 3 |
| INT8.4 | 91 | 0x0000 0DB6 | 2 | Reserved | 12 | 4 |
| INT8.5 | 92 | 0x0000 0DB8 | 2 | Reserved | 12 | 5 |
| INT8.6 | 93 | 0x0000 0DBA | 2 | Reserved | 12 | 6 |
| INT8.7 | 94 | 0x0000 0DBC | 2 | Reserved | 12 | 7 |
| INT8.8 | 95 | 0x0000 0DBE | 2 | Reserved | 12 | 8 |
| INT8.9 | 184 | 0x0000 0E70 | 2 | LINA interrupt 0 | 12 | 9 |
| INT8.10 | 185 | 0x0000 0E72 | 2 | LINA interrupt 1 | 12 | 10 |
| INT8.11 | 186 | 0x0000 0E74 | 2 | Reserved | 12 | 11 |
| INT8.12 | 187 | 0x0000 0E76 | 2 | Reserved | 12 | 12 |
| INT8.13 | 188 | 0x0000 0E78 | 2 | PMBUSA interrupt | 12 | 13 |
| INT8.14 | 189 | 0x0000 0E7A | 2 | Reserved | 12 | 14 |
| INT8.15 | 190 | 0x0000 0E7C | 2 | Reserved | 12 | 15 |
| INT8.16 | 191 | 0x0000 0E7E | 2 | Reserved | 12 | 16 (Lowest) |
| **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 | Reserved | 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 event interrupt | 14 | 1 (Highest) |
| INT10.2 | 105 | 0x0000 0DD2 | 2 | ADCA2 interrupt | 14 | 2 |
| INT10.3 | 106 | 0x0000 0DD4 | 2 | ADCA3 interrupt | 14 | 3 |
| INT10.4 | 107 | 0x0000 0DD6 | 2 | ADCA4 interrupt | 14 | 4 |
| INT10.5 | 108 | 0x0000 0DD8 | 2 | ADCB event interrupt | 14 | 5 |
| INT10.6 | 109 | 0x0000 0DDA | 2 | ADCB2 interrupt | 14 | 6 |

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

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

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

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

## 3.6 Exceptions and Non-Maskable Interrupts

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

### *3.6.1 Configuring and Using NMIs*

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

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

### *3.6.2 Emulation Considerations*

The NMI watchdog counter behaves as follows under debug conditions:

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

### *3.6.3 NMI Sources*

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

#### 3.6.3.1 Missing Clock Detection

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

#### 3.6.3.2 RAM Uncorrectable ECC Error

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

#### 3.6.3.3 Flash Uncorrectable ECC Error

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

#### 3.6.3.4 Software-Forced Error

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

### 3.6.4 Illegal Instruction Trap (ITRAP)

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

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

### 3.6.5 Error Pin

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

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

## 3.7 Clocking

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

**Figure 3-3. Clocking System**

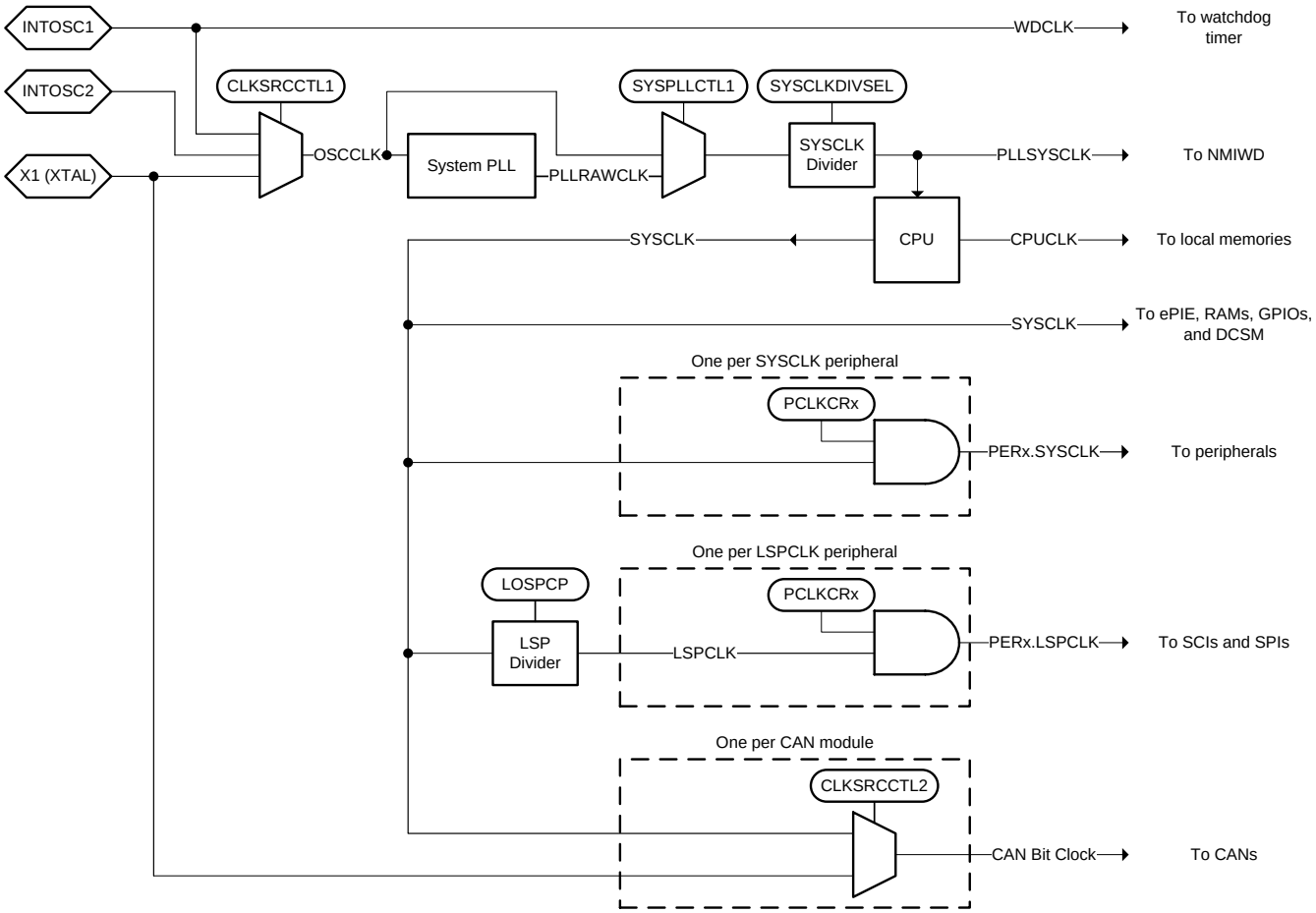
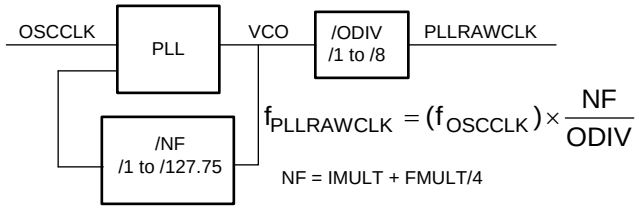



**Figure 3-4. System PLL**



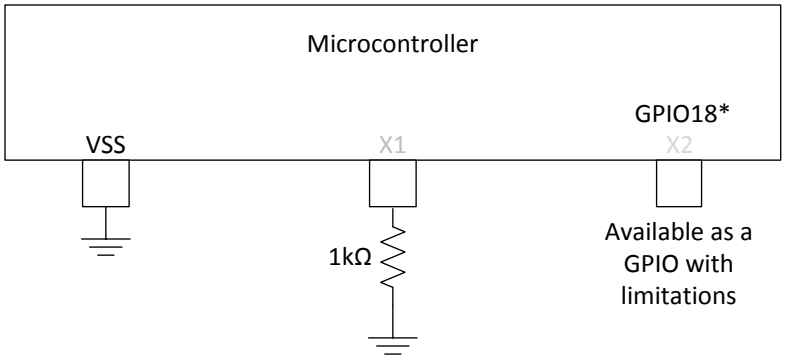

### 3.7.1 Clock Sources

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

#### 3.7.1.1 Primary Internal Oscillator (INTOSC2)

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

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




#### 3.7.1.2 Backup Internal Oscillator (INTOSC1)

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

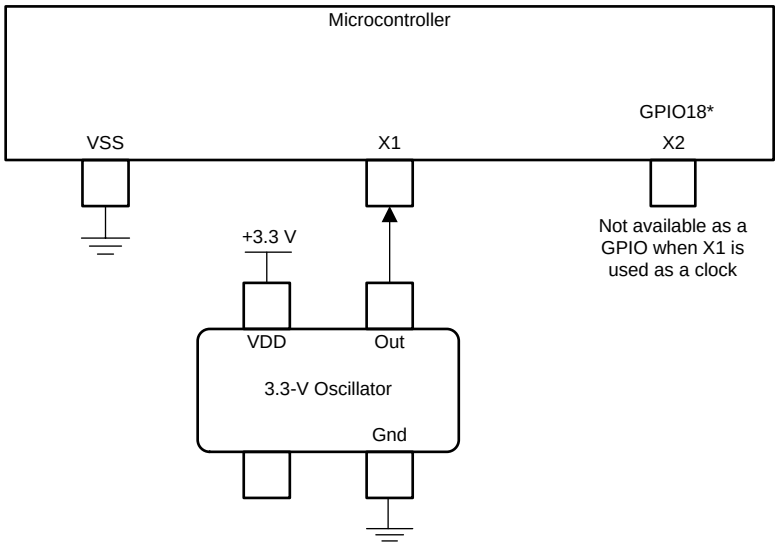
#### 3.7.1.3 External Oscillator (XTAL)

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

Three types of external clock sources are supported:

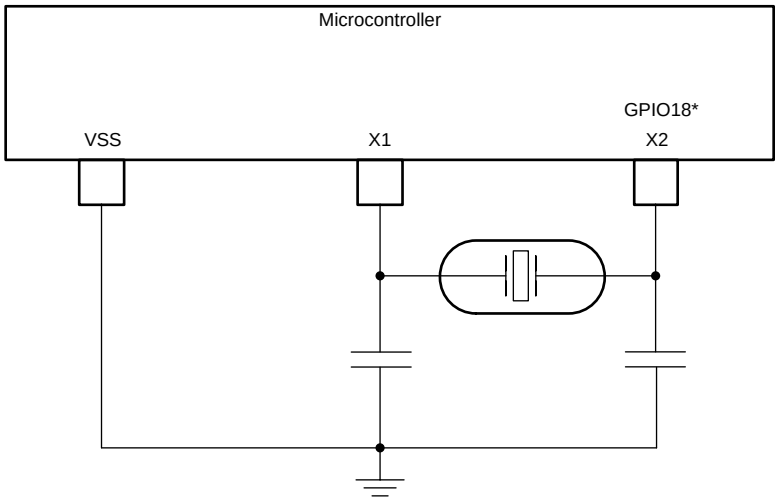
- A single-ended 3.3V external clock. The clock signal should be connected to X1, as shown in Figure 3-6. X2/GPIO18 is not available as a GPIO.

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



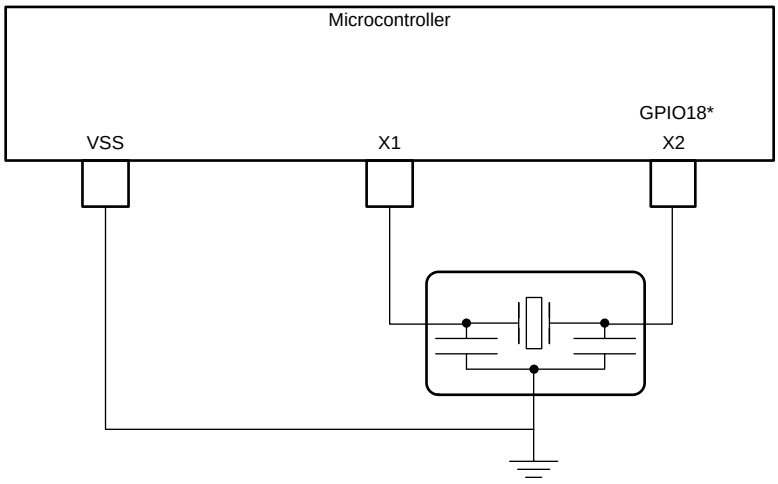

- An external crystal. The crystal should be connected across X1 and X2 with its load capacitors connected to VSS as shown in Figure 3-7.

**Figure 3-7. External Crystal**




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

**Figure 3-8. External Resonator**




### 3.7.2 Derived Clocks

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

#### 3.7.2.1 Oscillator Clock (OSCCLK)

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

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

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

### 3.7.3 Device Clock Domains

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

#### 3.7.3.1 System Clock (PLLSYSCLK)

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

#### 3.7.3.2 CPU Clock (CPUCLK)

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

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

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

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

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

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

#### 3.7.3.5 CAN Bit Clock

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

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

#### 3.7.3.6 CPU Timer2 Clock (TIMER2CLK)

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

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

### *3.7.4 XCLKOUT*

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

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

### *3.7.5 Clock Connectivity*

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

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

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

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

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

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

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

### 3.7.6 Clock Source and PLL Setup

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

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

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

### 3.7.7 Using an External Crystal or Resonator

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

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

### 3.7.8 Using an External Oscillator

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

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

### 3.7.9 Choosing PLL Settings

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

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

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

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

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

### 3.7.10 System Clock Setup

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

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

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

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

### 3.7.11 Clock Configuration Examples

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

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

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

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

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

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

### 3.7.12 Missing Clock Detection

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

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

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

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

- The MCDSCNT counter is frozen to prevent further missing clock detection.

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

Missing clock detection in enabled at startup.

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

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

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

**Figure 3-9. CPU Timers**

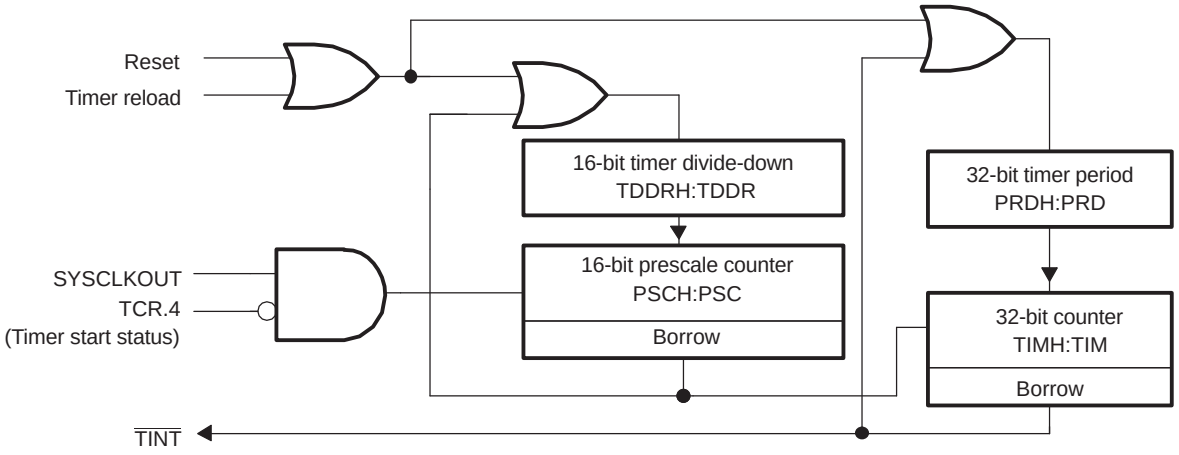
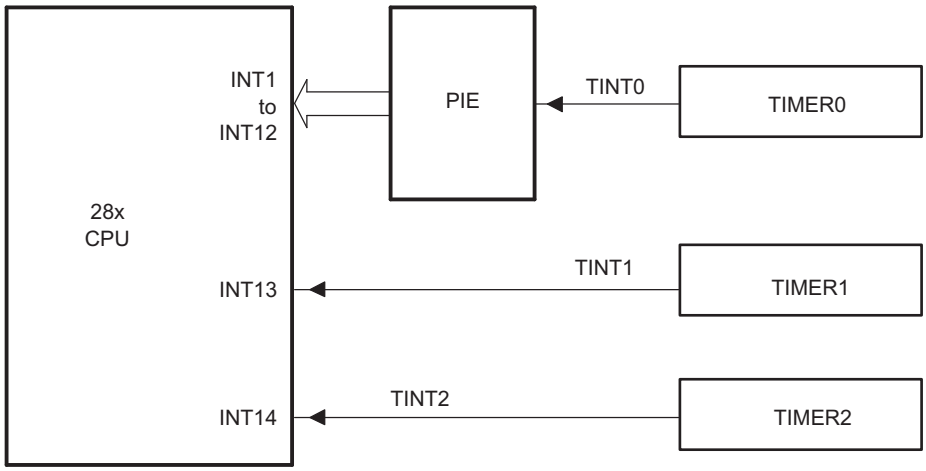



**Figure 3-10. CPU Timer Interrupt Signals and Output Signal**




A The timer registers are connected to the memory bus of the C28x processor.

B The CPU Timers are synchronized to SYSCLKOUT.

The general operation of a CPU timer is as follows:

- The 32-bit counter register, TIMH:TIM, is loaded with the value in the period register PRDH:PRD
- The counter decrements once every (TPR[TDDRH:TDDR]+1) SYSCLK 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 3.14 are used to configure the timers.

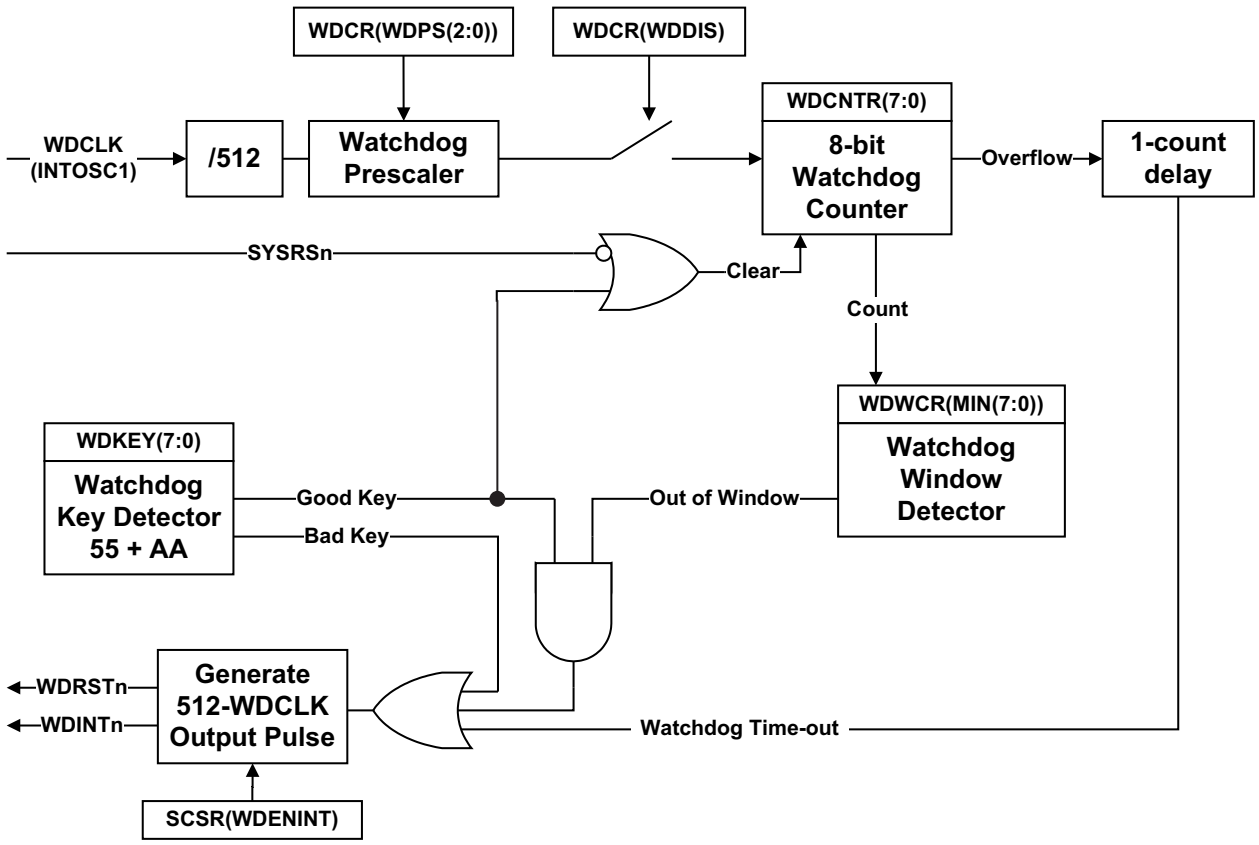
## 3.9 Watchdog Timer

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

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

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

**Figure 3-11. Watchdog Timer Module**




### 3.9.1 Servicing the Watchdog Timer

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

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

| Step | Value Written to WDKEY | Result |
|---|---|---|
| 1 | 0xAA | No action |
| 2 | 0xAA | No action |
| 3 | 0x55 | WDCNTR is enabled to be reset if next value is 0xAA. |
| 4 | 0x55 | WDCNTR is enabled to be reset if next value is 0xAA. |
| 5 | 0x55 | WDCNTR is enabled to be reset if next value is 0xAA. |
| 6 | 0xAA | WDCNTR is reset. |
| 7 | 0xAA | No action |
| 8 | 0x55 | WDCNTR is enabled to be reset if next value is 0xAA. |
| 9 | 0xAA | WDCNTR is reset. |
| 10 | 0x55 | WDCNTR is enabled to be reset if next value is 0xAA. |
| 11 | 0x32 | Improper value written to WDKEY.<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. |

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

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

### 3.9.2 Minimum Window Check

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

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

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

### 3.9.3 Watchdog Reset or Watchdog Interrupt Mode

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

- **Reset mode:**

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

- **Interrupt mode:**

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

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

### 3.9.4 Watchdog Operation in Low Power Modes

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

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

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

### 3.9.5 Emulation Considerations

The watchdog module behaves as follows under various debug conditions:

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

## 3.10 Low Power Modes

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

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

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

### 3.10.1 IDLE

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

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

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

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

### 3.10.2 Guidelines on Software Emulation of STANDBY Mode

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

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

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

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

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

### 3.10.3 HALT

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

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

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

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

**To enter HALT mode:**

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

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

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

**To wake up from HALT mode:**

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

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

### 3.10.4 Flash Power-down Considerations

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

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

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

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

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

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

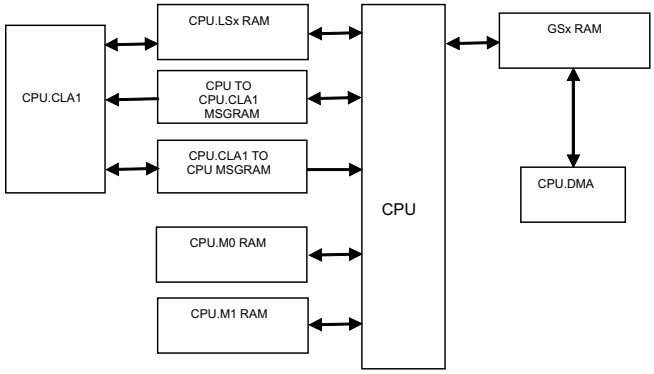
## 3.11 Memory Controller Module

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

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

All these RAMs are highly configurable to achieve control for write access and fetch access from different masters. All dedicated RAMs are enabled with the ECC feature (both data and address) and shared RAMs, are enabled with the parity (both data and address) feature. Some of the dedicated memories are secure memory as well. Refer to Section 3.13 for more details. Each RAM has its own controller which takes care of the access protection/security related checks and ECC/Parity features for that RAM. Figure 3-12 shows the configuration of these RAMs.

**Figure 3-12. Memory Architecture**




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

### *3.11.1 Functional Description*

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

#### 3.11.1.1 Dedicated RAM (Mx RAM)

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

All dedicated RAMs have the ECC feature.

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

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

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

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

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

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

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

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

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

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

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

#### 3.11.1.4 Message RAM (CLA MSGRAM)

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

This RAM has parity.

#### 3.11.1.5 Access Arbitration

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

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

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

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

1. Data Write
2. Data Read/Program Fetch

Figure 3-13 represents the arbitration scheme on global shared memories:

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

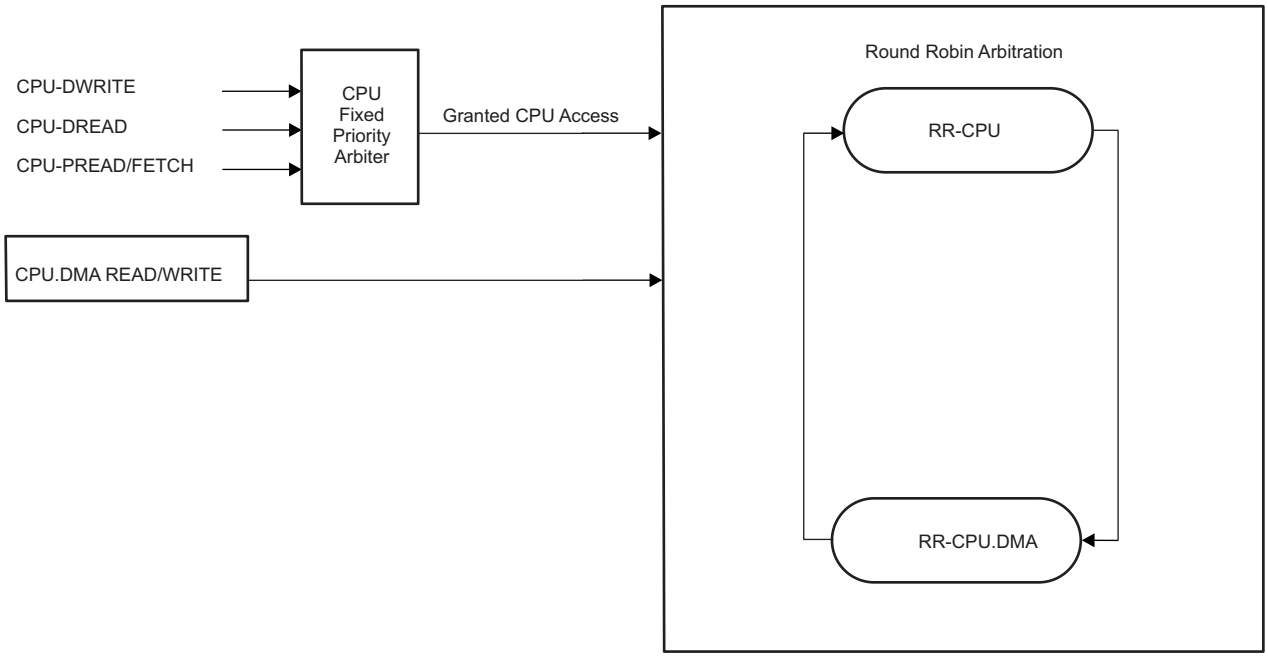
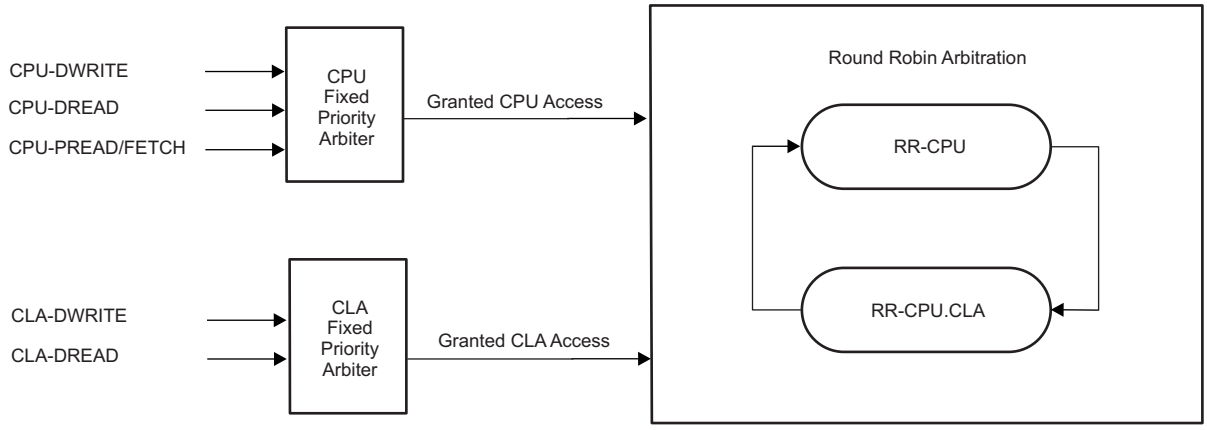



Figure 3-14 represents arbitration scheme on local shared memories.

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




### 3.11.1.6 Access Protection

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

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

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

#### *3.11.1.6.1 CPU Fetch Protection*

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

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

#### 3.11.1.6.2 CPU Write Protection

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

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

#### 3.11.1.6.3 CPU Read Protection

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

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

#### 3.11.1.6.4 CLA Fetch Protection

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

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

#### 3.11.1.6.5 CLA Write Protection

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

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

#### 3.11.1.6.6 CLA Read Protection

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

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

#### 3.11.1.6.7 DMA Write Protection

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

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

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

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

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

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

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

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

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

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

##### *3.11.1.7.1 Error Detection and Correction*

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

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

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

##### *3.11.1.7.2 Error Handling*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

#### 3.11.1.9 RAM Initialization

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

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

## 3.12 Flash and OTP Memory

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

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

### *3.12.1 Features*

Features of flash memory include:

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

### *3.12.2 Flash Tools*

Texas Instruments provides the following tools for flash:

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

### 3.12.3 Default Flash Configuration

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

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

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

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

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

### 3.12.4 Flash Bank, OTP and Pump

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

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

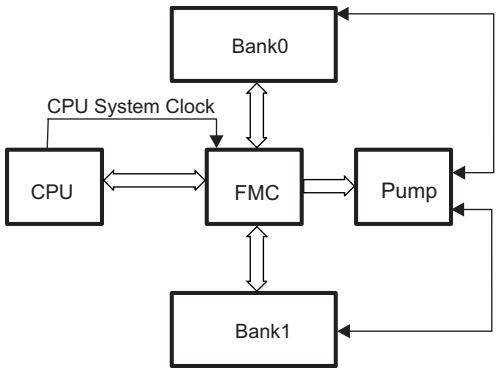
Bank0 and Bank1 share a common flash pump; therefore, only one bank can be programmed or erased at a time. Execution or reads from one bank, while erase or program is in progress on the other bank, is supported.

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

### 3.12.5 Flash Module Controller (FMC)

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

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



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

Section 3.12.6 through Section 3.12.10 describe FMC in detail.

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

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

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

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

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

The charge pump operates in two power modes:

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

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

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

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

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

#### 3.12.6.2 Active Grace Period

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

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

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

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

### 3.12.7 Flash and OTP Performance

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

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

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

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

where SYSCLK is the system operating frequency

FCLK is flash clock frequency. FCLK should 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. Please see the device data manual for the RWAIT configuration details.

### 3.12.8 Flash Access Interface

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

#### 3.12.8.1 Standard Access Mode

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

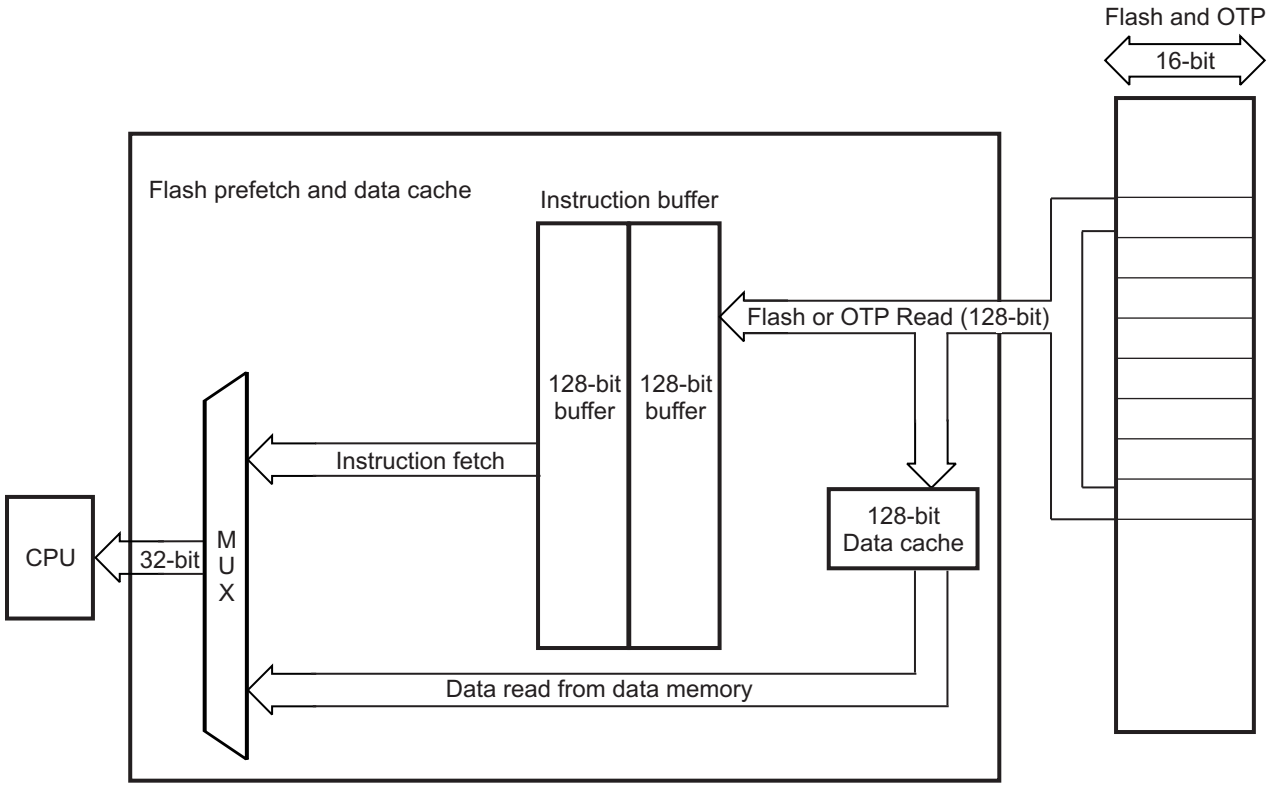
The prefetch buffer associated with the prefetch mechanism and data cache is bypassed in standard access mode; therefore, every access to the flash/OTP is used by the CPU immediately, and every access creates a unique flash bank access.

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

### 3.12.8.2 Prefetch Mode

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

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




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

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

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

**NOTE:** If the prefetch mechanism is enabled, then the last row of 128 bits in the bank should not be used, because the prefetch logic which does a look-ahead prefetch, will try to fetch from outside the bank and would result in an ECC error.

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

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

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

#### 3.12.8.3 Data Cache

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

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

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

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

### 3.12.9 Erase/Program Flash

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

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

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

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

#### 3.12.9.1 Erase

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

Fapi_issueAsyncCommandWithAddress();

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

Fapi_doBlankCheck();

#### 3.12.9.2 Program

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

This command is implemented by the following Flash API function:

Fapi_issueProgrammingCommand();

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

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

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

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

#### 3.12.9.3 Verify

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

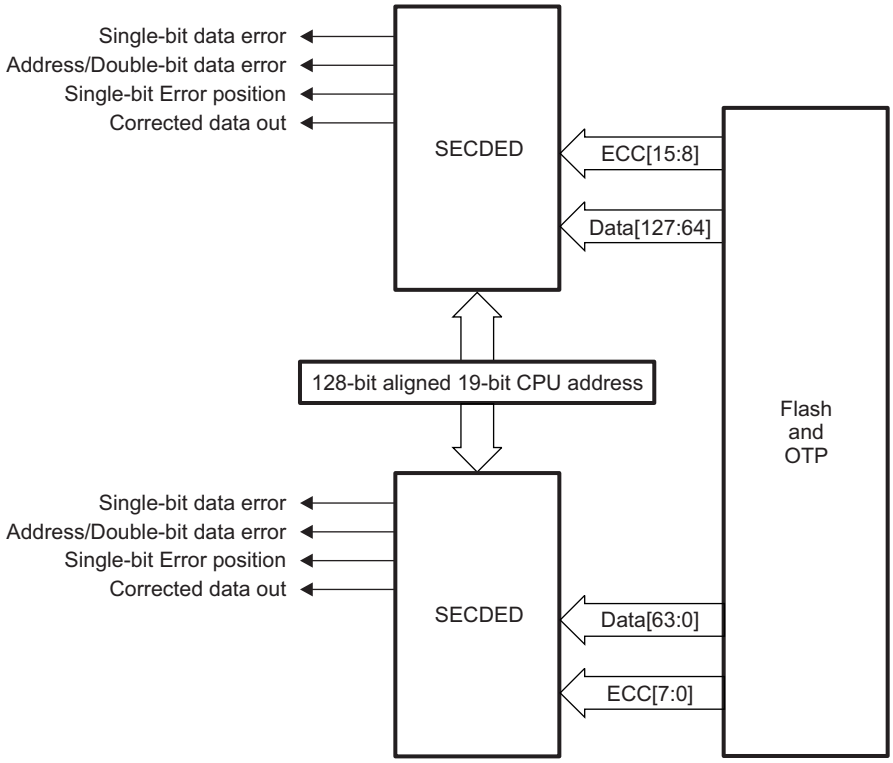
Application software typically perform a CRC check of the Flash memory contents during power-up and at regular intervals during runtime (as needed). Apart from this, ECC logic, when enabled (enabled by default) catches single-bit errors, double-bit errors, and address errors whenever the CPU reads or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

### 3.12.10 Error Correction Code (ECC) Protection

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

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

**Figure 3-17. ECC Logic Inputs and Outputs**




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

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

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

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

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

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

#### 3.12.10.1 Single-Bit Data Error

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

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

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

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

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

#### 3.12.10.2 Uncorrectable Error

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

- Address where the error occurred – if the uncorrectable error occurs in the lower 64-bits of a 128-bit memory-aligned data, the lower 64-bit memory-aligned address will be captured in the UNC_ERR_ADDR_LOW register. If the uncorrectable error occurs in the upper 64-bits of a 128-bit memory-aligned data, the upper 64-bit memory-aligned address will be captured in the UNC_ERR_ADDR_HIGH register.
- A flag is set indicating that an uncorrectable error occurred – the UNC_ERR_L and UNC_ERR_H flags in the ERR_STATUS register indicate the uncorrectable error occurrence in the lower 64-bits/lower 8-bit ECC, or the upper 64-bits/upper 8-bit ECC, respectively, of a 128-bit memory-aligned data.
- A flag is set indicating that an uncorrectable error interrupt is fired (UNC_ERR_INTFLG in ERR_INTFLG register)

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

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

#### 3.12.10.3 SECDED Logic Correctness Check

Since error detection and correction logic are part of safety-critical logic, safety applications may need to ensure that the SECDED logic is always working properly. For these safety concerns, in order to ensure the correctness of the SECDED logic, an ECC test mode is provided to test the correctness of ECC logic periodically. In 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 should not read/fetch from Flash when ECC test mode is enabled. For this reason, ECC test mode code should be executed from RAM and not from flash.

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

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

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

- Write 64-bit data in FDATAH_TEST (upper 32 bits) and FDATAL_TEST (lower 32 bits) registers
- Write the corresponding 8-bit ECC in the FECC_TEST register
- In any of the above three steps, users can insert errors (single-bit data error or double-bit data error or address error or single-bit ECC error or double-bit ECC error) so that they can check whether or not ECC logic is able to catch the errors
- Select the ECC logic block (lower 64-bits or upper 64-bits) which needs to be tested using the ECC_SELECT bit in the FECC_CTRL register
- Enable ECC test mode usingthe ECC_TEST_EN bit in FECC_CTRL register
- Write a value of 1 in the DO_ECC_CALC bit in FECC_CTRL register to enable ECC test logic for a single cycle to evaluate the address, data, ECC in FADDR_TEST, FDATAx_TEST and FECC_TEST registers for ECC errors

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

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

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

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

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

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

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

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

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

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

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

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

## 3.13 Dual Code Security Module (DCSM)

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

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

### *3.13.1 Functional Description*

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

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

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

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

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

| Zone 1 GRAMRAMx/GRABSECTx Bits | Zone 2 GRAMRAMx/GRABSECT | Ownership and Accessibility |
|---|---|---|
| 01 | 10 | RAM block/Flash Sector belongs to Zone1 |
| 01 | 11[1] | RAM block/Flash Sector belongs to Zone1 |
| 10 | 01 | RAM block/Flash Sector belongs to Zone2 |
| 11[2] | 01 | RAM block/Flash Sector belongs to Zone2 |
| 10 | 10 | RAM block/Flash Sector is unsecure |
| 11[2] | 11[1] | RAM block/Flash Sector is unsecure |

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

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

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

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

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

**Table 3-16. Security Levels**

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

### 3.13.1.1 CSM Passwords

Unlike earlier C2000 devices, on this device ALL_1 value (0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF) for CSM password for a zone does not unsecure the zone. Instead, if for any zone the CSM password values get loaded as ALL_1 from USER OTP, the device will be in BLOCKED state. Due to this reason TI will program a few bits in the second 32-bit password value (ZxOTP_CSMPSWD1) in every zone select block of each zone with value '0'. The default value for this password location is chosen in a manner that the respective ECC value remains ALL_1. Due to this, the CSMPSWD1 value programmed by TI for every zone select block is different. Please see Table 3-17 for ZxOTP_CSMPSWD1 value, programmed by TI on every device. Since ECC is not programmed, the user will be able to change this value by flipping the bits which are '1' to '0' but leaving the ones which are already programmed by TI as '0'. BOOTROM code will write the default password value into the KEYx register to unlock the device as part of device initialization sequence.

If the password locations of a zone have all 128 bits as zeros (ALL_0), that zone becomes permanently secure (LOCKED state), regardless of the contents of the CSMKEYx registers which means the zone cannot be unlocked using PMF, the password match flow described in Section 3.13.7.4. Therefore, the user should never use ALL_0 as password. A password of ALL_0 will prevent debug of secure code or reprogramming the Flash sectors. CSMKEYx registers are user-accessible registers that are used to unsecure the zones.

**Table 3-17. Default Value of ZxOTP_CSMPSWD1 (programmed by TI)**

| Zone Select Block | Zone1 (Z1OTP_CSMPSWD1) | | Zone2 (Z2OTP_CSMPSWD1) | |
|---|---|---|---|---|
| | Address | Value | Address | Value |
| Zone_Select_Block0 | 0x0007802a | 0x47ffffff | 0x0007822a | 0xe3ffffff |
| Zone_Select_Block1 | 0x0007803a | 0xdb7fffff | 0x0007823a | 0x977fffff |
| Zone_Select_Block2 | 0x0007804a | 0x4bffffff | 0x0007824a | 0xf1ffffff |
| Zone_Select_Block3 | 0x0007805a | 0x3f7fffff | 0x0007825a | 0x9b7fffff |
| Zone_Select_Block4 | 0x0007806a | 0xcfbfffff | 0x0007826a | 0x5b7fffff |
| Zone_Select_Block5 | 0x0007807a | 0x8bffffff | 0x0007827a | 0x2fffffff |
| Zone_Select_Block6 | 0x0007808a | 0x53ffffff | 0x0007828a | 0x1fffffff |
| Zone_Select_Block7 | 0x0007809a | 0xcf7fffff | 0x0007829a | 0x6b7fffff |
| Zone_Select_Block8 | 0x000780aa | 0xe77fffff | 0x000782aa | 0xab7fffff |
| Zone_Select_Block9 | 0x000780ba | 0x93ffffff | 0x000782ba | 0x37ffffff |

**Table 3-17. Default Value of ZxOTP_CSMPSWD1 (programmed by TI) (continued)**

| Zone Select Block | Zone1 (Z1OTP_CSMPSWD1) | | Zone2 (Z2OTP_CSMPSWD1) | |
|---|---|---|---|---|
| Zone_Select_Block10 | 0x000780ca | 0xeb7fffff | 0x000782ca | 0x4f7fffff |
| Zone_Select_Block11 | 0x000780da | 0x69ffffff | 0x000782da | 0x3bffffff |
| Zone_Select_Block12 | 0x000780ea | 0xa9ffffff | 0x000782ea | 0xe5ffffff |
| Zone_Select_Block13 | 0x000780fa | 0xdd7fffff | 0x000782fa | 0x8f7fffff |
| Zone_Select_Block14 | 0x0007810a | 0x8bffffff | 0x0007830a | 0x2fffffff |
| Zone_Select_Block15 | 0x0007811a | 0xcfbfffff | 0x0007831a | 0x5b7fffff |
| Zone_Select_Block16 | 0x0007812a | 0x3f7fffff | 0x0007832a | 0x9b7fffff |
| Zone_Select_Block17 | 0x0007813a | 0x4bffffff | 0x0007833a | 0xf1ffffff |
| Zone_Select_Block18 | 0x0007814a | 0xdb7fffff | 0x0007834a | 0x977fffff |
| Zone_Select_Block19 | 0x0007815a | 0x47ffffff | 0x0007835a | 0xe3ffffff |
| Zone_Select_Block20 | 0x0007816a | 0x87ffffff | 0x0007836a | 0xcbffffff |
| Zone_Select_Block21 | 0x0007817a | 0xf37fffff | 0x0007837a | 0x577fffff |
| Zone_Select_Block22 | 0x0007818a | 0xdd7fffff | 0x0007838a | 0x8f7fffff |
| Zone_Select_Block23 | 0x0007819a | 0xa9ffffff | 0x0007839a | 0xe5ffffff |
| Zone_Select_Block24 | 0x000781aa | 0x69ffffff | 0x000783aa | 0x3bffffff |
| Zone_Select_Block25 | 0x000781ba | 0xeb7fffff | 0x000783ba | 0x4f7fffff |
| Zone_Select_Block26 | 0x000781ca | 0x93ffffff | 0x000783ca | 0x37ffffff |
| Zone_Select_Block27 | 0x000781da | 0xe77fffff | 0x000783da | 0xab7fffff |
| Zone_Select_Block28 | 0x000781ea | 0xcf7fffff | 0x000783ea | 0x6b7fffff |
| Zone_Select_Block29 | 0x000781fa | 0x53ffffff | 0x000783fa | 0x1fffffff |

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

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

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

The solution to this problem is:

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

#### 3.13.1.3 CPU Secure Logic

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

#### 3.13.1.4 Execute-Only Protection

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

#### 3.13.1.5 Password Lock

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

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

#### 3.13.1.6 Link Pointer and Zone Select

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

**BANK0 USER OTP**

- B0_ZxOTP_LINKPOINTER1
- B0_ZxOTP_LINKPOINTER2
- B0_ZxOTP_LINKPOINTER3
- ZxOTP_GPREG1
- ZxOTP_GPREG2
- ZxOTP_PSWDLOCK
- ZxOTP_CRCLOCK
- ZxOTP_BOOTCTRL
- ZxOTP_GPREG3
- ZxOTP_EXEONLYRAM
- B0_ZxOTP_EXEONLYSECT
- ZxOTP_GRABRAM
- B0_ZxOTP_GRABSECT
- ZxOTP_CSMPSWD0
- ZxOTP_CSMPSWD1
- ZxOTP_CSMPSWD2
- ZxOTP_CSMPSWD3

**BANK1 USER OTP**

- B1_ZxOTP_LINKPOINTER1
- B1_ZxOTP_LINKPOINTER2
- B1_ZxOTP_LINKPOINTER3
- B1_ZxOTP_EXEONLYSECT

- B1_ZxOTP_GRABSECT

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

- ZxOTP_EXEONLYRAM
- B0_ZxOTP_EXEONLYSECT
- ZxOTP_GRABRAM
- B0_ZxOTP_GRABSECT
- ZxOTP_CSMPSWD0
- ZxOTP_CSMPSWD1
- ZxOTP_CSMPSWD2
- ZxOTP_CSMPSWD3
- B1_ZxOTP_EXEONLYSECT
- B1_ZxOTP_GRABSECT

The location of the zone select block in OTP is decided based on the value of three 29-bit link pointers (Bx-Zx-LINKPOINTERx) programmed in the OTP of each zone. All OTP locations except link pointer locations are protected with ECC. Since the link pointer locations are not protected with ECC, three link pointers are provided that need to be programmed with the same value. The final value of the link pointer is resolved in hardware when a dummy read is done to all the link pointers by comparing all the three values (bit-wise voting logic). Since in OTP, a '1' can be flipped by the user to '0' but '0' can't be flipped to '1' (no erase operation for OTP), the most significant bit position in the resolved link pointer which is '0', defines the valid base address for the zone select block. While generating the final link pointer value, if the bit pattern is not one of those listed in Figure 3-18, the final link pointer value becomes All_1 (0xFFFF_FFFF) which selects the Zone-Select-Block1 (also known as the default zone select block).

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

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

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

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

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

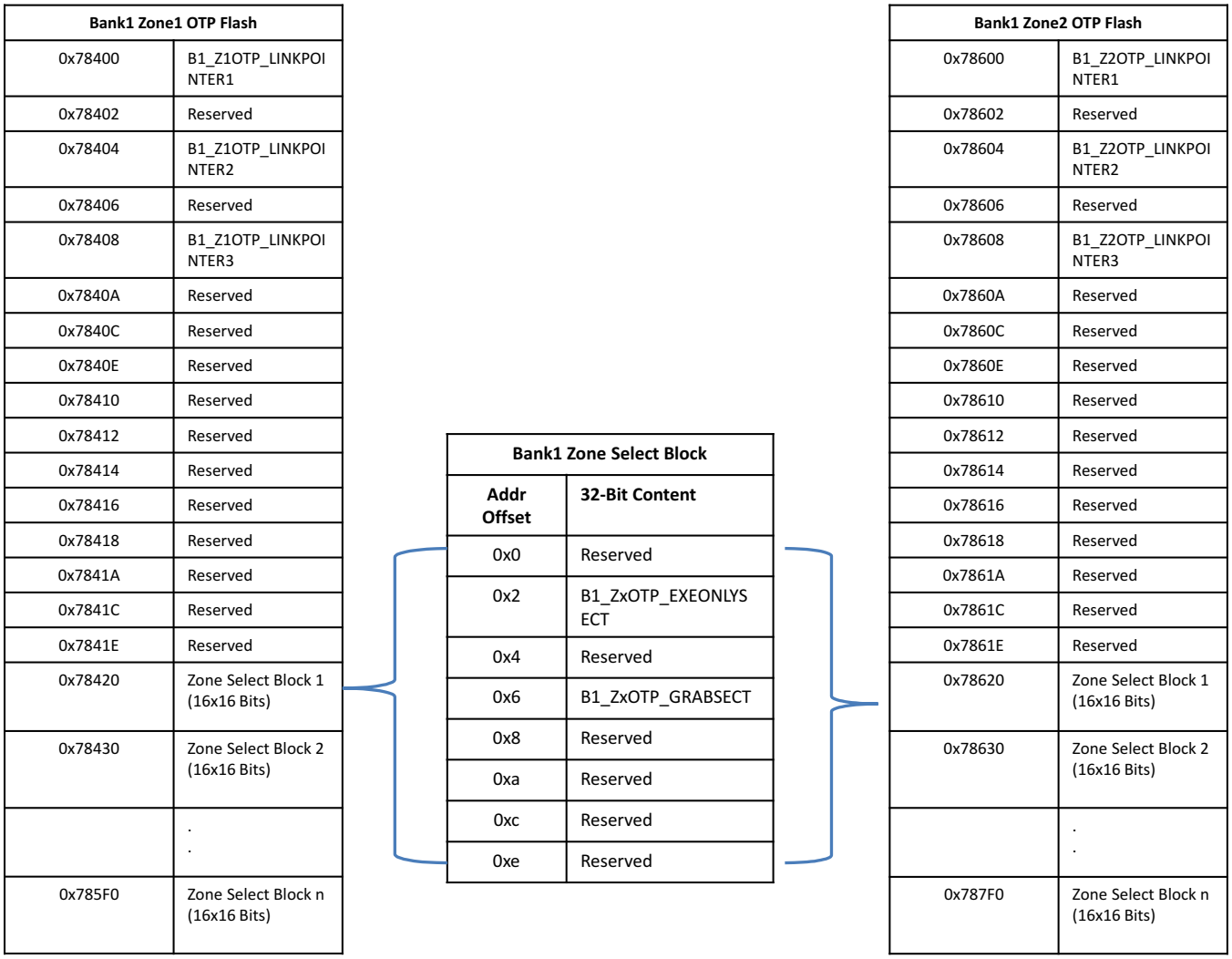
**Figure 3-19. Location of Zone-Select Block Based on Link-Pointer for Bank0**

| Bank0 Zone1 OTP Flash | |
|---|---|
| 0x78000 | B0_Z1OTP_LINKPOINTER1 |
| 0x78002 | Reserved |
| 0x78004 | B0_Z1OTP_LINKPOINTER2 |
| 0x78006 | Reserved |
| 0x78008 | B0_Z1OTP_LINKPOINTER3 |
| 0x7800A | Reserved |
| 0x7800C | Z1OTP_GPREG1 |
| 0x7800E | Z1OTP_GPREG2 |
| 0x78010 | Z1OTP_PSWDLOCK |
| 0x78012 | Reserved |
| 0x78014 | Z1OTP_CRCLOCK |
| 0x78016 | Reserved |
| 0x78018 | Reserved |
| 0x7801A | Reserved |
| 0x7801C | Z1OTP_GPREG3 |
| 0x7801E | Z1OTP_BOOTCTRL |
| 0x78020 | Zone Select Block 1 (16x16 Bits) |
| 0x78030 | Zone Select Block 2 (16x16 Bits) |
| | . . |
| 0x781F0 | Zone Select Block n (16x16 Bits) |

| Bank0 Zone Select Block | |
|---|---|
| **Addr Offset** | **32-Bit Content** |
| 0x0 | ZxOTP-EXEONLYRAM |
| 0x2 | B0_ZxOTP_EXEONLYSECT |
| 0x4 | ZxOTP_GRABRAM |
| 0x6 | B0_ZxOTP_GRABSECT |
| 0x8 | ZxOTP_CSMPSWD0 |
| 0xa | ZxOTP_CSMPSWD1 |
| 0xc | ZxOTP_CSMPSWD2 |
| 0xe | ZxOTP_CSMPSWD3 |

| Bank0 Zone2 OTP Flash | |
|---|---|
| 0x78200 | B0_Z2OTP_LINKPOINTER1 |
| 0x78202 | Reserved |
| 0x78204 | B0_Z2OTP_LINKPOINTER2 |
| 0x78206 | Reserved |
| 0x78208 | B0_Z2OTP_LINKPOINTER3 |
| 0x7820A | Reserved |
| 0x7800C | Z2OTP_GPREG1 |
| 0x7800E | Z2OTP_GPREG2 |
| 0x78210 | Z2OTP_PSWDLOCK |
| 0x78212 | Reserved |
| 0x78214 | Z2OTP_CRCLOCK |
| 0x78216 | Reserved |
| 0x78218 | Reserved |
| 0x7821A | Reserved |
| 0x7801C | Z2OTP_GPREG3 |
| 0x7821E | Z2OTP_BOOTCTRL |
| 0x78220 | Zone Select Block 1 (16x16 Bits) |
| 0x78230 | Zone Select Block 2 (16x16 Bits) |
| | . . |
| 0x783F0 | Zone Select Block n (16x16 Bits) |

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



| Bank1 Zone1 OTP Flash | |
|---|---|
| 0x78400 | B1_Z1OTP_LINKPOINTER1 |
| 0x78402 | Reserved |
| 0x78404 | B1_Z1OTP_LINKPOINTER2 |
| 0x78406 | Reserved |
| 0x78408 | B1_Z1OTP_LINKPOINTER3 |
| 0x7840A | Reserved |
| 0x7840C | Reserved |
| 0x7840E | Reserved |
| 0x78410 | Reserved |
| 0x78412 | Reserved |
| 0x78414 | Reserved |
| 0x78416 | Reserved |
| 0x78418 | Reserved |
| 0x7841A | Reserved |
| 0x7841C | Reserved |
| 0x7841E | Reserved |
| 0x78420 | Zone Select Block 1 (16x16 Bits) |
| 0x78430 | Zone Select Block 2 (16x16 Bits) |
| | . . |
| 0x785F0 | Zone Select Block n (16x16 Bits) |

| Bank1 Zone Select Block | |
|---|---|
| **Addr Offset** | **32-Bit Content** |
| 0x0 | Reserved |
| 0x2 | B1_ZxOTP_EXEONLYSECT |
| 0x4 | Reserved |
| 0x6 | B1_ZxOTP_GRABSECT |
| 0x8 | Reserved |
| 0xa | Reserved |
| 0xc | Reserved |
| 0xe | Reserved |

| Bank1 Zone2 OTP Flash | |
|---|---|
| 0x78600 | B1_Z2OTP_LINKPOINTER1 |
| 0x78602 | Reserved |
| 0x78604 | B1_Z2OTP_LINKPOINTER2 |
| 0x78606 | Reserved |
| 0x78608 | B1_Z2OTP_LINKPOINTER3 |
| 0x7860A | Reserved |
| 0x7860C | Reserved |
| 0x7860E | Reserved |
| 0x78610 | Reserved |
| 0x78612 | Reserved |
| 0x78614 | Reserved |
| 0x78616 | Reserved |
| 0x78618 | Reserved |
| 0x7861A | Reserved |
| 0x7861C | Reserved |
| 0x7861E | Reserved |
| 0x78620 | Zone Select Block 1 (16x16 Bits) |
| 0x78630 | Zone Select Block 2 (16x16 Bits) |
| | . . |
| 0x787F0 | Zone Select Block n (16x16 Bits) |

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

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

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

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

// Bits 31 30 and 29 as most-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;
}
```

### 3.13.3 Flash and OTP Erase/Program

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

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.

### 3.13.4 Safe Copy Code

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

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

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

### 3.13.5 SafeCRC

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

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

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

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

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

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

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

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

**Security Initialization**

- Dummy Read to address location of B0_Z1OTP_LINKPOINTER1 in Z1 OTP of Bank0
- Dummy Read to address location of B0_Z1OTP_LINKPOINTER2 in Z1 OTP of Bank0
- Dummy Read to address location of B0_Z1OTP_LINKPOINTER3 in Z1 OTP of Bank0
- Dummy Read to address location of B0_Z2OTP_LINKPOINTER1 in Z2 OTP of Bank0
- Dummy Read to address location of B0_Z2OTP_LINKPOINTER2 in Z2 OTP of Bank0
- Dummy Read to address location of B0_Z2OTP_LINKPOINTER3 in Z2 OTP of Bank0
- Dummy Read to address location of B1_Z1OTP_LINKPOINTER1 in Z1 OTP of Bank1
- Dummy Read to address location of B1_Z1OTP_LINKPOINTER2 in Z1 OTP of Bank1
- Dummy Read to address location of B1_Z1OTP_LINKPOINTER3 in Z1 OTP of Bank1
- Dummy Read to address location of B1_Z2OTP_LINKPOINTER1 in Z2 OTP of Bank1
- Dummy Read to address location of B1_Z2OTP_LINKPOINTER2 in Z2 OTP of Bank1
- Dummy Read to address location of B1_Z2OTP_LINKPOINTER3 in Z2 OTP of Bank1

- Dummy Read to address location of SECDC (0x703F0, TI-reserved register) in TI OTP
- Dummy Read to address location of Z1OTP_PSWDLOCK in Z1 OTP
- Dummy Read to address location of Z1OTP_CRCLOCK in Z1 OTP
- Dummy Read to address location 0x78018 in Z1 OTP
- Dummy Read to address location of Z1OTP_GPREG1, Z1OTP_GPREG2, Z1OTP_GPREG3 in Z1 OTP of Bank0
- Dummy Read to address location of Z1OTP_BOOTCTRL in Z1 OTP
- Dummy Read to address location of Z2OTP_PSWDLOCK in Z2 OTP of Bank0
- Dummy Read to address location of Z2OTP_CRCLOCK in Z2 OTP of Bank0
- Dummy Read to address location 0x78218 in Z2 OTP of Bank0
- Dummy Read to address location of Z2OTP_GPREG1, Z2OTP_GPREG2, Z2OTP_GPREG3 in Z2 OTP of Bank0
- Dummy Read to address location of Z2OTP_BOOTCTRL in Z2 OTP of Bank0
- Read to memory map register of B0_Z1_LINKPOINTER in DCSM module to calculate the address of zone select block for Z1
- Dummy read to address location of Z1OTP_EXEONLYRAM in Z1 OTP of Bank0
- Dummy read to address location of B0_Z1OTP_EXEONLYSECT in Z1 OTP of Bank0
- Dummy read to address location of Z1OTP_GRABRAM in Z1 OTP of Bank0
- Dummy read to address location of B0_Z1OTP_GRABSECT in Z1 OTP of Bank0
- Read to memory map register of B0_Z2_LINKPOINTER in DCSM module to calculate the address of zone select block for Z2
- Dummy read to address location of Z2OTP_EXEONLYRAM in Z2 OTP of Bank0
- Dummy read to address location of B0_Z2OTP_EXEONLYSECT in Z2 OTP of Bank0
- Dummy read to address location of Z2OTP_GRABRAM in Z2 OTP of Bank0
- Dummy read to address location of B0_Z2OTP_GRABSECT in Z2 OTP of Bank0
- Read to memory map register of B1_Z1_LINKPOINTER in DCSM module to calculate the address of zone select block for Z1
- Dummy read to address location of B1_Z1OTP_EXEONLYSECT in Z1 OTP of Bank1
- Dummy read to address location of B1_Z1OTP_GRABSECT in Z1 OTP of Bank1
- Read to memory map register of B1_Z2_LINKPOINTER in DCSM module to calculate the address of zone select block for Z2
- Dummy read to address location of B1_Z2OTP_EXEONLYSECT in Z2 OTP of Bank1
- Dummy read to address location of B1_Z2OTP_GRABSECT in Z2 OTP of Bank1

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

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

### 3.13.7 Incorporating Code Security in User Applications

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

#### 3.13.7.1 Environments That Require Security Unlocking

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

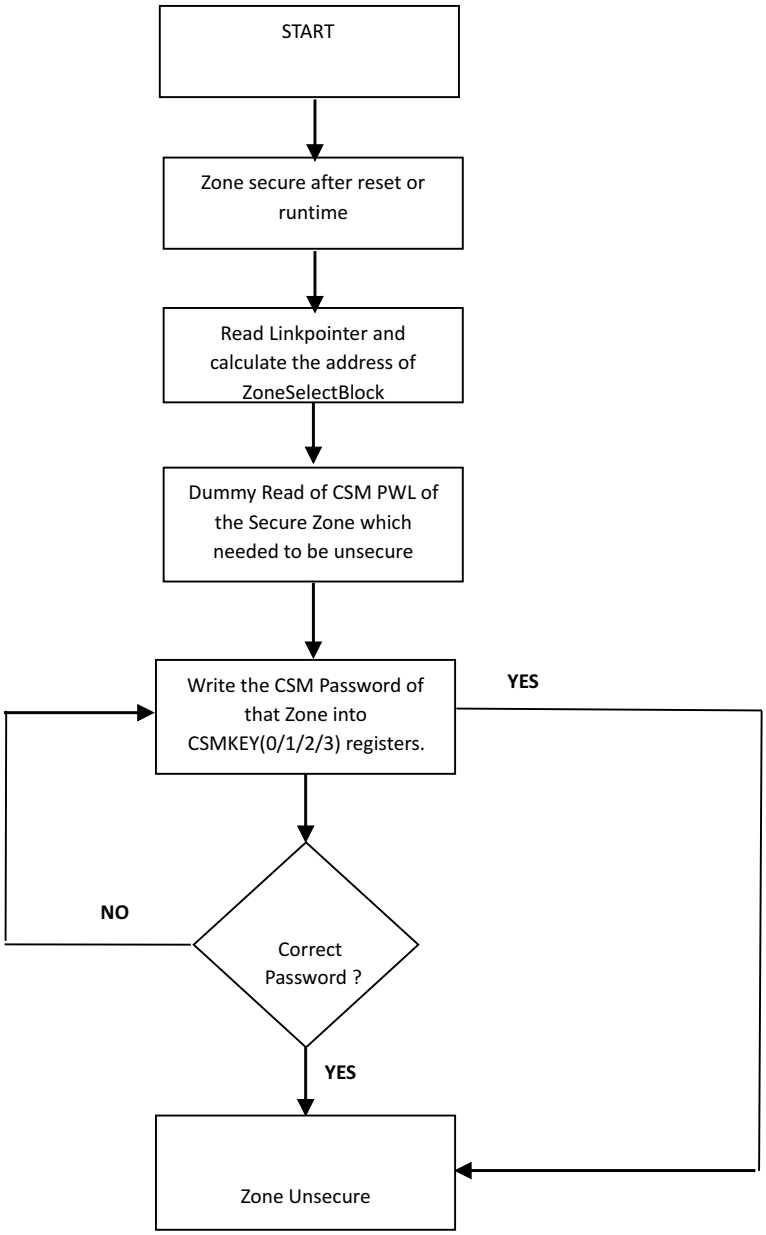
- Code development using debuggers (such as Code Composer Studio). This is the most common environment during the design phase of a product.
- Flash programming using TI's Flash utilities such as Code Composer Studio On-Chip Flash Programmer plug-in or the Uniflash tool. Flash programming is common during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

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

### 3.13.7.2 CSM Password Match Flow

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

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



### 3.13.7.3 C Code Example to Unsecure C28x Zone1

```
volatile long int *CSM = (volatile long int *)0x5F000;    //CSM register file
volatile long int *CSMPWL = (volatile long int *)0x78028; //CSM Password location (assuming
default Zone select block)
                                                            volatile int tmp;
int I;
// Read the 128-bits of the CSM password locations (PWL)
//
for (I=0;I<4; I++) tmp = *CSMPWL++;
// If the password locations (CSMPWL) are all = ones (0xFFFF),
// then the zone will now be unsecure. If the password
// is not all ones (0xFFFF), then the code below is required
// to unsecure the CSM.
// Write the 128-bit password to the CSMKEY registers
// If this password matches that stored in the
// CSLPWL then the CSM will become unsecure. If it does not
// match, then the zone will remain secure.
// An example password of:
// 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
```

### 3.13.7.4 C Code Example to Resecure C28x Zone1

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

### 3.13.7.5 Environments That Require ECSL Unlocking

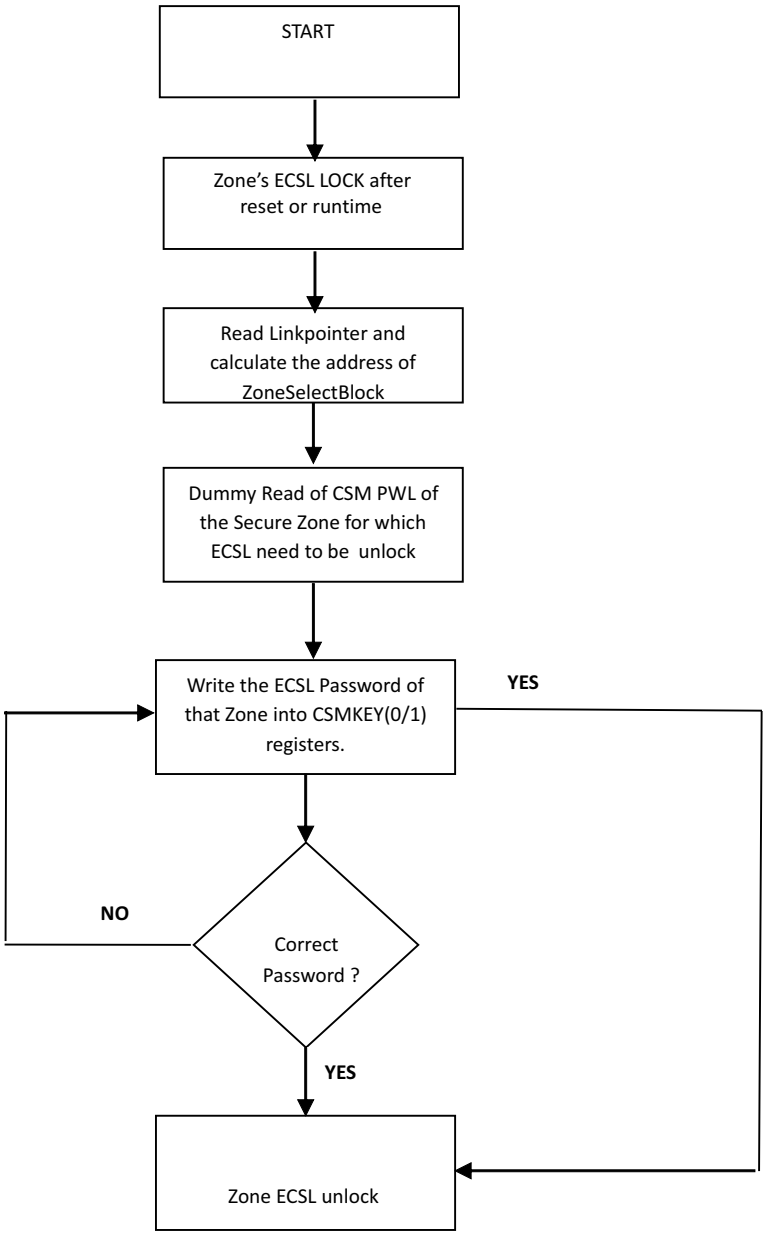
The following are the typical situations under which unsecuring can be required:

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

### 3.13.7.6 ECSL Password Match Flow

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

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




### 3.13.7.7 ECSL Disable Considerations for any Zone

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

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

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

```
volatile long int *ECSL = (volatile int *)0x5F000;    //ECSL register file
volatile long int *ECSLPWL = (volatile int *)0x78028; //ECSL Password location (assuming default
Zone sel block)
                                                        volatile int tmp;
int I;
// Read the 64-bits of the password locations (PWL)
.
for (I=0;I<2; I++) tmp = *ECSLPWL++;
// If the ECSL password locations (ECSLPWL) are all = ones (0xFFFF),
// then the ECSL will now be disable. If the password
// is not all ones (0xFFFF), then the code below is required
// to disable the ECSL.
// Write the 64-bit password to the CSMKEYx registers
// If this password matches that stored in the
// CSMPWL then ECSL will get disable. If it does not
// match, then the zone will remain secure.
// An example password of:
// 0x1111222233334444 is used.
*ECSL++ = 0x22221111; // Register Z1_CSMKEY0 at 0x5F010
*ECSL++ = 0x44443333; // Register Z1_CSMKEY1 at 0x5F012
```

### 3.13.7.8 Device Unique ID

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

## 3.14 Registers

### 3.14.1 System Control Base Addresses

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

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

### 3.14.1.1 ACCESS_PROTECTION_REGS Registers

Table 3-19 lists the memory-mapped registers for the ACCESS_PROTECTION_REGS. All register offset addresses not listed in Table 3-19 should be considered as reserved locations and the register contents should not be modified.

**Table 3-19. ACCESS_PROTECTION_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | NMAVFLG | Non-Master Access Violation Flag Register | | 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 3-20 shows the codes that are used for access types in this section.

**Table 3-20. ACCESS_PROTECTION_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| W=1 | W | Write |
| **Reset or Default Value** | | |
| -*n* | | Value after reset or the default value |
| **Register Array Variables** | | |

**Table 3-20. ACCESS_PROTECTION_REGS Access Type Codes (continued)**

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

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

NMAVFLG is shown in Figure 3-23 and described in Table 3-21.

Return to Summary Table.

Non-Master Access Violation Flag Register

**Figure 3-23. NMAVFLG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | RESERVED | RESERVED |
| R-0h | | | | | | 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 3-21. NMAVFLG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | CLA1FETCH | R | 0h | Non Master CLA1 Fetch Access Violation Flag:<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 |

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | CPUWRITE | R | 0h | Non Master CPU Write Access Violation Flag:<br>0: No violation.<br>1: Access violation occured.<br>Reset type: SYSRSn |
| 0 | CPUREAD | R | 0h | Non Master CPU Read Access Violation Flag:<br>0: No violation.<br>1: Access violation occured.<br>Reset type: SYSRSn |

#### 3.14.1.1.2 NMAVSET Register (Offset = 2h) [reset = 0h]

NMAVSET is shown in Figure 3-24 and described in Table 3-22.

Return to Summary Table.

Non-Master Access Violation Flag Set Register

**Figure 3-24. NMAVSET Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | RESERVED | RESERVED |
| R-0h | | | | | | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | CLA1FETCH | CLA1WRITE | CLA1READ | DMAWRITE | CPUFETCH | CPUWRITE | CPUREAD |
| R-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | CLA1FETCH | R=0/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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 |

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | CPUREAD | R=0/W=1 | 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 |

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

NMAVCLR is shown in Figure 3-25 and described in Table 3-23.

Return to Summary Table.

Non-Master Access Violation Flag Clear Register

**Figure 3-25. NMAVCLR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | RESERVED | RESERVED |
| R-0h | | | | | | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | CLA1FETCH | CLA1WRITE | CLA1READ | DMAWRITE | CPUFETCH | CPUWRITE | CPUREAD |
| R-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | CLA1FETCH | R=0/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 0h | 0: No action.<br>1: CPU Write Access Violation Flag in NMAVFLG register will be cleared.<br>Reset type: SYSRSn |

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

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

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

NMAVINTEN is shown in Figure 3-26 and described in Table 3-24.

Return to Summary Table.

Non-Master Access Violation Interrupt Enable Register

**Figure 3-26. NMAVINTEN Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | RESERVED | RESERVED |
| R-0h | | | | | | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | CLA1FETCH | CLA1WRITE | CLA1READ | RESERVED | CPUFETCH | CPUWRITE | CPUREAD |
| R-0h | R/W-0h | R/W-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h |

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | CLA1FETCH | R/W | 0h | 0: CLA1 Non Master Fetch Access Violation Interrupt is disabled.<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 | RESERVED | R | 0h | Reserved |
| 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 |

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

NMCPURDAVADDR is shown in Figure 3-27 and described in Table 3-25.

Return to Summary Table.

Non-Master CPU Read Access Violation Address

**Figure 3-27. NMCPURDAVADDR Register**

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

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

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

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

NMCPUWRAVADDR is shown in Figure 3-28 and described in Table 3-26.

Return to Summary Table.

Non-Master CPU Write Access Violation Address

**Figure 3-28. NMCPUWRAVADDR Register**

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

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

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

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

NMCPUFAVADDR is shown in Figure 3-29 and described in Table 3-27.

Return to Summary Table.

Non-Master CPU Fetch Access Violation Address

**Figure 3-29. NMCPUFAVADDR Register**

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

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

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

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

NMDMAWRAVADDR is shown in Figure 3-30 and described in Table 3-28.

Return to Summary Table.

Non-Master DMA Write Access Violation Address

**Figure 3-30. NMDMAWRAVADDR Register**

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

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

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

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

NMCLA1RDAVADDR is shown in Figure 3-31 and described in Table 3-29.

Return to Summary Table.

Non-Master CLA1 Read Access Violation Address

**Figure 3-31. NMCLA1RDAVADDR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NMCLA1RDAVADDR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

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

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

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

NMCLA1WRAVADDR is shown in Figure 3-32 and described in Table 3-30.

Return to Summary Table.

Non-Master CLA1 Write Access Violation Address

**Figure 3-32. NMCLA1WRAVADDR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NMCLA1WRAVADDR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

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

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

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

NMCLA1FAVADDR is shown in Figure 3-33 and described in Table 3-31.

Return to Summary Table.

Non-Master CLA1 Fetch Access Violation Address

**Figure 3-33. NMCLA1FAVADDR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NMCLA1FAVADDR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

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

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

#### 3.14.1.1.12 MAVFLG Register (Offset = 20h) [reset = 0h]

MAVFLG is shown in Figure 3-34 and described in Table 3-32.

Return to Summary Table.

Master Access Violation Flag Register

**Figure 3-34. MAVFLG Register**

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

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-3 | RESERVED | R | 0h | Reserved |
| 2 | DMAWRITE | R | 0h | Master DMA Write Access Violation Flag:<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 |

#### 3.14.1.1.13 MAVSET Register (Offset = 22h) [reset = 0h]

MAVSET is shown in Figure 3-35 and described in Table 3-33.

Return to Summary Table.

Master Access Violation Flag Set Register

**Figure 3-35. MAVSET Register**

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

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-3 | RESERVED | R | 0h | Reserved |
| 2 | DMAWRITE | R=0/W=1 | 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/W=1 | 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/W=1 | 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 |

#### 3.14.1.1.14 MAVCLR Register (Offset = 24h) [reset = 0h]

MAVCLR is shown in Figure 3-36 and described in Table 3-34.

Return to Summary Table.

Master Access Violation Flag Clear Register

**Figure 3-36. MAVCLR Register**

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

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-3 | RESERVED | R | 0h | Reserved |
| 2 | DMAWRITE | R=0/W=1 | 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/W=1 | 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/W=1 | 0h | 0: No action.<br>1: CPU Fetch Access Violation Flag in MAVFLG register will be cleared.<br>Reset type: SYSRSn |

#### 3.14.1.1.15 MAVINTEN Register (Offset = 26h) [reset = 0h]

MAVINTEN is shown in Figure 3-37 and described in Table 3-35.

Return to Summary Table.

Master Access Violation Interrupt Enable Register

**Figure 3-37. MAVINTEN Register**

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

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-3 | RESERVED | R | 0h | Reserved |
| 2 | DMAWRITE | R/W | 0h | 0: DMA Write Access Violation Interrupt is disabled.<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 |

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

MCPUFAVADDR is shown in Figure 3-38 and described in Table 3-36.

Return to Summary Table.

Master CPU Fetch Access Violation Address

**Figure 3-38. MCPUFAVADDR Register**

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

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

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

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

MCPUWRAVADDR is shown in Figure 3-39 and described in Table 3-37.

Return to Summary Table.

Master CPU Write Access Violation Address

**Figure 3-39. MCPUWRAVADDR Register**

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

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

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

#### 3.14.1.1.18 MDMAWRAVADDR Register (Offset = 2Ch) [reset = 0h]

MDMAWRAVADDR is shown in Figure 3-40 and described in Table 3-38.

Return to Summary Table.

Master DMA Write Access Violation Address

**Figure 3-40. 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 3-38. 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 |

### 3.14.1.2 CLK_CFG_REGS Registers

Table 3-39 lists the memory-mapped registers for the CLK_CFG_REGS. All register offset addresses not listed in Table 3-39 should be considered as reserved locations and the register contents should not be modified.

**Table 3-39. CLK_CFG_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 2h | CLKCFGLOCK1 | Lock bit for CLKCFG registers | EALLOW | 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 |
| 22h | SYSCLKDIVSEL | System Clock Divider Select 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 |
| 32h | XTALCR | XTAL Control Register | EALLOW | Go |

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

**Table 3-40. CLK_CFG_REGS Access Type Codes**

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

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

CLKCFGLOCK1 is shown in Figure 3-41 and described in Table 3-41.

Return to Summary Table.

Lock bit for CLKCFG registers
Notes:
[1] Any bit in this register, once set can only be cleared through a CPU1.SYSRSn. Write of 0 to any bit of this register has no effect
[2] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed

**Figure 3-41. CLKCFGLOCK1 Register**

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

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-17 | RESERVED | R=0 | 0h | Reserved |
| 16 | XTALCR | R/WSOnce | 0h | Lock bit for XTALCR register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: SYSRSn |
| 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: 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: SYSRSn |
| 12 | RESERVED | R | 0h | Reserved |
| 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: SYSRSn |
| 10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R=0 | 0h | Reserved |
| 8 | RESERVED | R=0 | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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: SYSRSn |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 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: 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: 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: 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: SYSRSn |

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

CLKSRCCTL1 is shown in Figure 3-42 and described in Table 3-42.

Return to Summary Table.

Clock Source Control register-1

**Figure 3-42. CLKSRCCTL1 Register**

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

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5 | WDHALTI | R/W | 0h | Watchdog HALT Mode Ignore Bit: This bit determines if WD is functional in the HALT mode or not.<br>0 = WD is not functional in the HALT mode. Clock to WD is gated when system enters HALT mode. Additionally, INTOSC1 and INTOSC2 are powered-down when system enters HALT mode<br>1 = WD is functional in the HALT mode. Clock to WD is not gated and INTOSC1/2 are not powered-down when system enters HALT mode<br>Reset type: XRSn |
| 4 | RESERVED | R | 0h | Reserved |
| 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 |

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 or disabling the previous clock source to allow the change to<br>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 |

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

CLKSRCCTL2 is shown in Figure 3-43 and described in Table 3-43.

Return to Summary Table.

Clock Source Control register-2

**Figure 3-43. CLKSRCCTL2 Register**

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

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

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

#### 3.14.1.2.4 CLKSRCCTL3 Register (Offset = Ch) [reset = 0h]

CLKSRCCTL3 is shown in Figure 3-44 and described in Table 3-44.

Return to Summary Table.

Clock Source Control register-3

**Figure 3-44. CLKSRCCTL3 Register**

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

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

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

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

SYSPLLCTL1 is shown in Figure 3-45 and described in Table 3-45.

Return to Summary Table.

SYSPLL Control register-1

**Figure 3-45. SYSPLLCTL1 Register**

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

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-2 | RESERVED | R=0 | 0h | Reserved |
| 1 | PLLCLKEN | R/W | 0h | SYSPLL bypassed or included in the PLLSYSCLK path: This bit decides if the SYSPLL is bypassed when PLLSYSCLK is generated<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 |

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

SYSPLLMULT is shown in Figure 3-46 and described in Table 3-46.

Return to Summary Table.

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

**Figure 3-46. SYSPLLMULT Register**

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

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | RESERVED | R=0 | 0h | Reserved |
| 29-24 | RESERVED | R | 0h | Reserved |
| 23-19 | RESERVED | R=0 | 0h | Reserved |
| 18-16 | ODIV | R/W | 0h | SYSPLL Output Clock Divider<br>PLL Output Divider = ODIV + 1<br>Reset type: XRSn |
| 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 |

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

SYSPLLSTS is shown in Figure 3-47 and described in Table 3-47.

Return to Summary Table.

SYSPLL Status register

**Figure 3-47. SYSPLLSTS Register**

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

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-2 | RESERVED | R=0 | 0h | Reserved |
| 1 | SLIPS | R | 0h | SYSPLL Slip Status Bit: This bit indicates whether the SYSPLL is out of lock range<br>0 = SYSPLL is not out of lock<br>1 = SYSPLL is out of lock<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 |

#### 3.14.1.2.8 SYSCLKDIVSEL Register (Offset = 22h) [reset = 2h]

SYSCLKDIVSEL is shown in Figure 3-48 and described in Table 3-48.

Return to Summary Table.

System Clock Divider Select register

**Figure 3-48. 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 3-48. SYSCLKDIVSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-0 | PLLSYSCLKDIV | R/W | 2h | PLLSYSCLK Divide Select: This bit selects the divider setting for the PLLSYSCLK.<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 |

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

XCLKOUTDIVSEL is shown in Figure 3-49 and described in Table 3-49.

Return to Summary Table.

XCLKOUT Divider Select register

**Figure 3-49. XCLKOUTDIVSEL Register**

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

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

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

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

LOSPCP is shown in Figure 3-50 and described in Table 3-50.

Return to Summary Table.

Low Speed Clock Source Prescalar

**Figure 3-50. LOSPCP Register**

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

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-3 | RESERVED | R=0 | 0h | Reserved |
| 2-0 | LSPCLKDIV | R/W | 2h | These bits configure the low-speed peripheral clock (LSPCLK) rate<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: SYSRSn |

#### 3.14.1.2.11 MCDCR Register (Offset = 2Eh) [reset = 0h]

MCDCR is shown in Figure 3-51 and described in Table 3-51.

Return to Summary Table.

Missing Clock Detect Control Register

**Figure 3-51. MCDCR Register**

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

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-4 | RESERVED | R=0 | 0h | Reserved |
| 3 | OSCOFF | R/W | 0h | Oscillator Clock Disconnect from MCD Bit:<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/W=1 | 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 |

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

X1CNT is shown in Figure 3-52 and described in Table 3-52.

Return to Summary Table.

10-bit Counter on X1 Clock

**Figure 3-52. X1CNT Register**

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

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-17 | RESERVED | R=0 | 0h | Reserved |
| 16 | CLR | R=0/W=1 | 0h | X1 Counter clear:<br>A write of '1' to this bit field clears the X1CNT and makes it count from 0x0 again (provided X1 clock is ticking).<br>Writes of '0' are ignore to this bit field<br>Reset type: N/A |
| 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 guarantee that the Crystal connected to X1/X2 is powered Up.<br>Reset type: PORn |

#### 3.14.1.2.13 XTALCR Register (Offset = 32h) [reset = 5h]

XTALCR is shown in Figure 3-53 and described in Table 3-53.

Return to Summary Table.

XTAL Control Register

**Figure 3-53. XTALCR Register**

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

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

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-3 | RESERVED | R=0 | 0h | Reserved |
| 2 | SWH | R/W | 1h | Configures XTAL oscillator to optimally work with (lower) 2 to 10 MHz or (higher) 10 to 25MHz Frequncy ranges.<br>0 : Oscillator tuned to 2 to 10 MHz operation.<br>1 : Oscillator tuned above 10 MHz to 25 MHz.<br>Reset type: XRSn |
| 1 | SE | R/W | 0h | Configures XTAL oscillator in single-ended or Crystal mode when XTAL oscillator is powered up(i.e. OSCOFF = 0)<br>0 XTAL oscillator in Crystal mode<br>1 XTAL oscilator in single-ended mode (through X1)<br>Reset type: XRSn |
| 0 | OSCOFF | R/W | 1h | This bit if '1', powers-down the XTAL oscillator macro and hence doesn't let X2 to be driven by the XTAL oscillator. If a crystal is connected to X1/X2, user needs to first clear this bit, wait for the oscillator to power up (using X1CNT) and then only switch the clock source to X1/X2<br>Reset type: XRSn |

### 3.14.1.3 CPU_SYS_REGS Registers

Table 3-54 lists the memory-mapped registers for the CPU_SYS_REGS. All register offset addresses not listed in Table 3-54 should be considered as reserved locations and the register contents should not be modified.

**Table 3-54. CPU_SYS_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | CPUSYSLOCK1 | Lock bit for CPUSYS registers | EALLOW | Go |
| Ah | PIEVERRADDR | PIE Vector Fetch Error Address register | EALLOW | Go |
| 22h | PCLKCR0 | 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 |
| 3Ch | PCLKCR13 | Peripheral Clock Gating Registers | EALLOW | Go |
| 3Eh | PCLKCR14 | Peripheral Clock Gating Registers | EALLOW | Go |
| 40h | PCLKCR15 | Peripheral Clock Gating Registers | EALLOW | Go |
| 42h | PCLKCR16 | Peripheral Clock Gating Registers | EALLOW | Go |
| 44h | PCLKCR17 | Peripheral Clock Gating Registers | EALLOW | Go |
| 46h | PCLKCR18 | Peripheral Clock Gating Registers | EALLOW | Go |
| 48h | PCLKCR19 | Peripheral Clock Gating Registers | EALLOW | Go |
| 4Ah | PCLKCR20 | Peripheral Clock Gating Registers | EALLOW | Go |
| 4Ch | PCLKCR21 | Peripheral Clock Gating Registers | 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 |
| 7Eh | RESCCLR | Reset Cause Clear Register | | Go |
| 80h | RESC | Reset Cause register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-55 shows the codes that are used for access types in this section.

**Table 3-55. CPU_SYS_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| W=1 | W | Write |
| WSOnce | SOnce<br>W | Set once<br>Write |
| **Reset or Default Value** | | |
| *-n* | | Value after reset or the default value |

**Table 3-55. CPU_SYS_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. |

#### 3.14.1.3.1 CPUSYSLOCK1 Register (Offset = 0h) [reset = 0h]

CPUSYSLOCK1 is shown in Figure 3-54 and described in Table 3-56.

Return to Summary Table.

Lock bit for CPUSYS registers
Notes:
[1] Any bit in this register, once set can only be cleared through a CPU1.SYSRSn. Write of 0 to any bit of this register has no effect
[2] The locking mechanism applies to only writes. Reads to the registers which have LOCK protection are always allowed

**Figure 3-54. CPUSYSLOCK1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | PCLKCR21 | PCLKCR20 | PCLKCR19 | PCLKCR18 | PCLKCR17 |
| R-0h | R-0h | R-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIOLPMSEL1 | GPIOLPMSEL0 | LPMCR | RESERVED | PCLKCR16 | PCLKCR15 | PCLKCR14 | PCLKCR13 |
| R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | PCLKCR10 | PCLKCR9 | PCLKCR8 | PCLKCR7 | PCLKCR6 | RESERVED |
| R-0h | R-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PCLKCR4 | PCLKCR3 | PCLKCR2 | RESERVED | PCLKCR0 | PIEVERRADDR | RESERVED | RESERVED |
| R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R-0h | R/WSOnce-0h | R/WSOnce-0h | R-0h | R-0h |

**Table 3-56. CPUSYSLOCK1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RESERVED | R | 0h | Reserved |
| 30 | RESERVED | R | 0h | Reserved |
| 29 | RESERVED | R | 0h | Reserved |
| 28 | PCLKCR21 | R/WSOnce | 0h | Lock bit for PCLKCR21 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br><br>Reset type: SYSRSn |
| 27 | PCLKCR20 | R/WSOnce | 0h | Lock bit for PCLKCR20 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br><br>Reset type: SYSRSn |
| 26 | PCLKCR19 | R/WSOnce | 0h | Lock bit for PCLKCR19 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br><br>Reset type: SYSRSn |
| 25 | PCLKCR18 | R/WSOnce | 0h | Lock bit for PCLKCR18 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br><br>Reset type: SYSRSn |

**Table 3-56. CPUSYSLOCK1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24 | PCLKCR17 | R/WSOnce | 0h | Lock bit for PCLKCR17 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br><br>Reset type: SYSRSn |
| 23 | GPIOLPMSEL1 | R/WSOnce | 0h | Lock bit for GPIOLPMSEL1 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br><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><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><br>Reset type: SYSRSn |
| 20 | RESERVED | R | 0h | Reserved |
| 19 | PCLKCR16 | R/WSOnce | 0h | Lock bit for PCLKCR16 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br><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><br>Reset type: SYSRSn |
| 17 | PCLKCR14 | R/WSOnce | 0h | Lock bit for PCLKCR14 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br><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><br>Reset type: SYSRSn |
| 15 | RESERVED | R | 0h | Reserved |
| 14 | RESERVED | R | 0h | Reserved |
| 13 | PCLKCR10 | R/WSOnce | 0h | Lock bit for PCLKCR10 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br><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><br>Reset type: SYSRSn |

**Table 3-56. CPUSYSLOCK1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | PCLKCR8 | R/WSOnce | 0h | Lock bit for PCLKCR8 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br><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><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><br>Reset type: SYSRSn |
| 8 | RESERVED | R | 0h | Reserved |
| 7 | PCLKCR4 | R/WSOnce | 0h | Lock bit for PCLKCR4 Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br><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><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><br>Reset type: SYSRSn |
| 4 | RESERVED | R | 0h | Reserved |
| 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 | RESERVED | R | 0h | Reserved |
| 0 | RESERVED | R | 0h | Reserved |

#### 3.14.1.3.2 PIEVERRADDR Register (Offset = Ah) [reset = 003FFFFFh]

PIEVERRADDR is shown in Figure 3-55 and described in Table 3-57.

Return to Summary Table.

PIE Vector Fetch Error Address register

**Figure 3-55. PIEVERRADDR Register**

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

**Table 3-57. 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 |

#### 3.14.1.3.3 PCLKCR0 Register (Offset = 22h) [reset = 38h]

PCLKCR0 is shown in Figure 3-56 and described in Table 3-58.

Return to Summary Table.

Peripheral Clock Gating Registers

**Figure 3-56. PCLKCR0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | RESERVED | TBCLKSYNC | RESERVED | HRPWM |
| R=0-0h | | | | R-0h | R/W-0h | R=0-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | CPUTIMER2 | CPUTIMER1 | CPUTIMER0 | DMA | RESERVED | CLA1 |
| R=0-0h | | R/W-1h | R/W-1h | R/W-1h | R/W-0h | R-0h | R/W-0h |

**Table 3-58. PCLKCR0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | RESERVED | R=0 | 0h | Reserved |
| 19 | RESERVED | R | 0h | Reserved |
| 18 | TBCLKSYNC | R/W | 0h | EPWM Time Base Clock sync: When set PWM time bases of all the PWM modules belonging to the same CPU-Subsystem (as partitioned using their CPUSEL bits) start counting<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>Reset type: SYSRSn |
| 15-6 | RESERVED | R=0 | 0h | Reserved |
| 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 |

**Table 3-58. PCLKCR0 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 | 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 |

#### 3.14.1.3.4 PCLKCR2 Register (Offset = 26h) [reset = 0h]

PCLKCR2 is shown in Figure 3-57 and described in Table 3-59.

Return to Summary Table.

Peripheral Clock Gating Registers

**Figure 3-57. PCLKCR2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-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 3-59. PCLKCR2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R=0 | 0h | Reserved |
| 15 | RESERVED | R | 0h | Reserved |
| 14 | RESERVED | R | 0h | Reserved |
| 13 | RESERVED | R | 0h | Reserved |
| 12 | RESERVED | R | 0h | Reserved |
| 11 | RESERVED | R | 0h | Reserved |
| 10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 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 |
| 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 |

**Table 3-59. PCLKCR2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | EPWM4 | R/W | 0h | EPWM4 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<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>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>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>Reset type: SYSRSn |

#### 3.14.1.3.5 PCLKCR3 Register (Offset = 28h) [reset = 0h]

PCLKCR3 is shown in Figure 3-58 and described in Table 3-60.

Return to Summary Table.

Peripheral Clock Gating Registers

**Figure 3-58. PCLKCR3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | ECAP7 | ECAP6 | ECAP5 | ECAP4 | ECAP3 | ECAP2 | ECAP1 |
| R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-60. PCLKCR3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-8 | RESERVED | R=0 | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | ECAP7 | R/W | 0h | ECAP7 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 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 |

**Table 3-60. PCLKCR3 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |

#### 3.14.1.3.6 PCLKCR4 Register (Offset = 2Ah) [reset = 0h]

PCLKCR4 is shown in Figure 3-59 and described in Table 3-61.

Return to Summary Table.

Peripheral Clock Gating Registers

**Figure 3-59. PCLKCR4 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | RESERVED | EQEP2 | EQEP1 |
| R=0-0h | | | | R-0h | R-0h | R/W-0h | R/W-0h |

**Table 3-61. PCLKCR4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-4 | RESERVED | R=0 | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 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 |

#### 3.14.1.3.7 PCLKCR6 Register (Offset = 2Eh) [reset = 0h]

PCLKCR6 is shown in Figure 3-60 and described in Table 3-62.

Return to Summary Table.

Peripheral Clock Gating Registers

**Figure 3-60. PCLKCR6 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | |
| R=0-0h | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | SD1 |
| R=0-0h | | | | | | | | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R/W-0h |

**Table 3-62. PCLKCR6 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-8 | RESERVED | R=0 | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1 | RESERVED | R | 0h | Reserved |
| 0 | 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 |

#### 3.14.1.3.8 PCLKCR7 Register (Offset = 30h) [reset = 0h]

PCLKCR7 is shown in Figure 3-61 and described in Table 3-63.

Return to Summary Table.

Peripheral Clock Gating Registers

**Figure 3-61. PCLKCR7 Register**

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

**Table 3-63. PCLKCR7 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-4 | RESERVED | R=0 | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 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 |

#### 3.14.1.3.9 PCLKCR8 Register (Offset = 32h) [reset = 0h]

PCLKCR8 is shown in Figure 3-62 and described in Table 3-64.

Return to Summary Table.

Peripheral Clock Gating Registers

**Figure 3-62. PCLKCR8 Register**

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

**Table 3-64. PCLKCR8 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-18 | RESERVED | R=0 | 0h | Reserved |
| 17 | RESERVED | R | 0h | Reserved |
| 16 | RESERVED | R | 0h | Reserved |
| 15-4 | RESERVED | R=0 | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 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 |

### 3.14.1.3.10 PCLKCR9 Register (Offset = 34h) [reset = 0h]

PCLKCR9 is shown in Figure 3-63 and described in Table 3-65.

Return to Summary Table.

Peripheral Clock Gating Registers

**Figure 3-63. PCLKCR9 Register**

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

**Table 3-65. PCLKCR9 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-2 | RESERVED | R=0 | 0h | Reserved |
| 1 | RESERVED | R | 0h | Reserved |
| 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 |

### 3.14.1.3.11 PCLKCR10 Register (Offset = 36h) [reset = 0h]

PCLKCR10 is shown in Figure 3-64 and described in Table 3-66.

Return to Summary Table.

Peripheral Clock Gating Registers

**Figure 3-64. PCLKCR10 Register**

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

**Table 3-66. PCLKCR10 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-4 | RESERVED | R=0 | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 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 |

### 3.14.1.3.12 PCLKCR13 Register (Offset = 3Ch) [reset = 0h]

PCLKCR13 is shown in Figure 3-65 and described in Table 3-67.

Return to Summary Table.

Peripheral Clock Gating Registers

**Figure 3-65. PCLKCR13 Register**

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

**Table 3-67. PCLKCR13 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-4 | RESERVED | R=0 | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 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 |

### 3.14.1.3.13 PCLKCR14 Register (Offset = 3Eh) [reset = 0h]

PCLKCR14 is shown in Figure 3-66 and described in Table 3-68.

Return to Summary Table.

Peripheral Clock Gating Registers

**Figure 3-66. PCLKCR14 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | CMPSS7 | CMPSS6 | CMPSS5 | CMPSS4 | CMPSS3 | CMPSS2 | CMPSS1 |
| R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-68. PCLKCR14 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-8 | RESERVED | R=0 | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 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 |

**Table 3-68. 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 |

#### 3.14.1.3.14 PCLKCR15 Register (Offset = 40h) [reset = 0h]

PCLKCR15 is shown in Figure 3-67 and described in Table 3-69.

Return to Summary Table.

Peripheral Clock Gating Registers

**Figure 3-67. PCLKCR15 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | PGA7 | PGA6 | PGA5 | PGA4 | PGA3 | PGA2 | PGA1 |
| R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-69. PCLKCR15 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-8 | RESERVED | R=0 | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | PGA7 | R/W | 0h | PGA7 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 5 | PGA6 | R/W | 0h | PGA6 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 4 | PGA5 | R/W | 0h | PGA5 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 3 | PGA4 | R/W | 0h | PGA4 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 2 | PGA3 | R/W | 0h | PGA3 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |
| 1 | PGA2 | R/W | 0h | PGA2 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |

**Table 3-69. PCLKCR15 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | PGA1 | R/W | 0h | PGA1 Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |

#### 3.14.1.3.15 PCLKCR16 Register (Offset = 42h) [reset = 0h]

PCLKCR16 is shown in Figure 3-68 and described in Table 3-70.

Return to Summary Table.

Peripheral Clock Gating Registers

**Figure 3-68. PCLKCR16 Register**

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

**Table 3-70. PCLKCR16 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | RESERVED | R=0 | 0h | Reserved |
| 19 | RESERVED | R | 0h | Reserved |
| 18 | RESERVED | R | 0h | Reserved |
| 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 | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1 | RESERVED | R | 0h | Reserved |
| 0 | RESERVED | R | 0h | Reserved |

### 3.14.1.3.16 PCLKCR17 Register (Offset = 44h) [reset = 0h]

PCLKCR17 is shown in Figure 3-69 and described in Table 3-71.

Return to Summary Table.

Peripheral Clock Gating Registers

**Figure 3-69. PCLKCR17 Register**

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

**Table 3-71. PCLKCR17 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-4 | RESERVED | R=0 | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1 | RESERVED | R | 0h | Reserved |
| 0 | RESERVED | R | 0h | Reserved |

#### 3.14.1.3.17 PCLKCR18 Register (Offset = 46h) [reset = 0h]

PCLKCR18 is shown in Figure 3-70 and described in Table 3-72.

Return to Summary Table.

Peripheral Clock Gating Registers

**Figure 3-70. PCLKCR18 Register**

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

**Table 3-72. PCLKCR18 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-4 | RESERVED | R=0 | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1 | RESERVED | R | 0h | Reserved |
| 0 | RESERVED | R | 0h | Reserved |

### 3.14.1.3.18 PCLKCR19 Register (Offset = 48h) [reset = 0h]

PCLKCR19 is shown in Figure 3-71 and described in Table 3-73.

Return to Summary Table.

Peripheral Clock Gating Registers

**Figure 3-71. PCLKCR19 Register**

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

**Table 3-73. PCLKCR19 Register Field Descriptions**

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

#### 3.14.1.3.19 PCLKCR20 Register (Offset = 4Ah) [reset = 0h]

PCLKCR20 is shown in Figure 3-72 and described in Table 3-74.

Return to Summary Table.

Peripheral Clock Gating Registers

**Figure 3-72. PCLKCR20 Register**

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

**Table 3-74. PCLKCR20 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-2 | RESERVED | R=0 | 0h | Reserved |
| 1 | RESERVED | R | 0h | Reserved |
| 0 | PMBUS_A | R/W | 0h | PMBUS_A Clock Enable bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |

### 3.14.1.3.20 PCLKCR21 Register (Offset = 4Ch) [reset = 0h]

PCLKCR21 is shown in Figure 3-73 and described in Table 3-75.

Return to Summary Table.

Peripheral Clock Gating Registers

**Figure 3-73. PCLKCR21 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | DCC_0 |
| R=0-0h | | | | | | | R/W-0h |

**Table 3-75. PCLKCR21 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-1 | RESERVED | R=0 | 0h | Reserved |
| 0 | DCC_0 | R/W | 0h | DCC Clock Enable Bit:<br>0: Module clock is gated-off<br>1: Module clock is turned-on<br>Reset type: SYSRSn |

#### 3.14.1.3.21 LPMCR Register (Offset = 76h) [reset = FCh]

LPMCR is shown in Figure 3-74 and described in Table 3-76.

Return to Summary Table.

LPM Control Register

**Figure 3-74. LPMCR Register**

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

**Table 3-76. LPMCR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RESERVED | R | 0h | Reserved |
| 30-18 | RESERVED | R=0 | 0h | Reserved |
| 17-16 | RESERVED | R | 0h | Reserved |
| 15 | RESERVED | R | 0h | Reserved |
| 14-8 | RESERVED | R=0 | 0h | Reserved |
| 7-2 | RESERVED | R | 0h | Reserved |
| 1-0 | LPM | R/W | 0h | These bits set the low power mode for the device. Takes effect when CPU executes the IDLE instruction (when IDLE instruction is out of EXE Phase of the Pipeline)<br>00: IDLE Mode<br>01: Reserved<br>1x: HALT Mode<br>Reset type: SYSRSn |

### 3.14.1.3.22 GPIOLPMSEL0 Register (Offset = 78h) [reset = 0h]

GPIOLPMSEL0 is shown in Figure 3-75 and described in Table 3-77.

Return to Summary Table.

GPIO LPM Wakeup select registers

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

**Figure 3-75. 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 3-77. 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 |

**Table 3-77. GPIOLPMSEL0 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 3-77. GPIOLPMSEL0 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 3.14.1.3.23 GPIOLPMSEL1 Register (Offset = 7Ah) [reset = 0h]

GPIOLPMSEL1 is shown in Figure 3-76 and described in Table 3-78.

Return to Summary Table.

GPIO LPM Wakeup select registers

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

**Figure 3-76. 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 3-78. 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 |

**Table 3-78. GPIOLPMSEL1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 3-78. GPIOLPMSEL1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 3.14.1.3.24 TMR2CLKCTL Register (Offset = 7Ch) [reset = 0h]

TMR2CLKCTL is shown in Figure 3-77 and described in Table 3-79.

Return to Summary Table.

Timer2 Clock Measurement functionality control register

**Figure 3-77. TMR2CLKCTL Register**

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

**Table 3-79. TMR2CLKCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-3 | TMR2CLKPRESCALE | R/W | 0h | 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>Reset type: SYSRSn |
| 2-0 | TMR2CLKSRCSEL | R/W | 0h | CPU Timer 2 Clock Source Select Bit: This bit selects the source for CPU Timer 2:<br>000 =SYSCLK Selected (default on reset, pre-scale is bypassed)<br>001 = INTOSC1<br>010 = INTOSC2<br>011 = XTAL<br>100 = FLPUMPOSC<br>101 = FOSCCLK<br>110 = AUXPLLCLK (Reserved)<br>111 = reserved<br>Reset type: SYSRSn |

### 3.14.1.3.25 RESCCLR Register (Offset = 7Eh) [reset = 0h]

RESCCLR is shown in Figure 3-78 and described in Table 3-80.

Return to Summary Table.

Reset Cause Clear Register

**Figure 3-78. RESCCLR Register**

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

**Table 3-80. RESCCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-9 | RESERVED | R=0 | 0h | Reserved |
| 8 | SCCRESETn | W=1 | 0h | Clear bit for corresponding status bit in RESC. Read of RESCCLR always gives 0.<br>Writing a 1 to this bit clears the status bit in RESC to 0<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 7 | RESERVED | R=0 | 0h | Reserved |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R=0 | 0h | Reserved |
| 3 | NMIWDRSn | W=1 | 0h | Clear bit for corresponding status bit in RESC. Read of RESCCLR always gives 0.<br>Writing a 1 to this bit clears the status bit in RESC to 0<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 2 | WDRSn | W=1 | 0h | Clear bit for corresponding status bit in RESC. Read of RESCCLR always gives 0.<br>Writing a 1 to this bit clears the status bit in RESC to 0<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 1 | XRSn | W=1 | 0h | Clear bit for corresponding status bit in RESC. Read of RESCCLR always gives 0.<br>Writing a 1 to this bit clears the status bit in RESC to 0<br>Writing 0 has no effect.<br>Reset type: SYSRSn |
| 0 | POR | W=1 | 0h | Clear bit for corresponding status bit in RESC. Read of RESCCLR always gives 0.<br>Writing a 1 to this bit clears the status bit in RESC to 0<br>Writing 0 has no effect.<br>Reset type: SYSRSn |

### 3.14.1.3.26 RESC Register (Offset = 80h) [reset = X]

RESC is shown in Figure 3-79 and described in Table 3-81.

Return to Summary Table.

Reset Cause register

**Figure 3-79. RESC Register**

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

**Table 3-81. RESC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | DCON | R | 0h | Reading this bit provides the status of debugger connection to the C28x CPU.<br>0 : Debugger is not connected to the C28x CPU<br>1 : Debugger is connected to the C28x CPU<br>Notes:<br>[1] This bit is connected to the DCON o/p signal of the C28x CPU<br>Reset type: N/A |
| 30 | XRSn_pin_status | R | X | Reading this bit provides the current status of the XRSn pin. Reset value is reflective of the pin status.<br>Reset type: N/A |
| 29-9 | RESERVED | R=0 | 0h | Reserved |
| 8 | SCCRESETn | R | 0h | If this bit is set, indicates that the device was reset by SCCRESETn (fired by DCSM).<br>Writing a 1 to this bit will force the bit to 0<br>Writing of 0 will have no effect.<br>Reset type: PORn |
| 7 | RESERVED | R=0 | 0h | Reserved |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R=0 | 0h | Reserved |
| 3 | NMIWDRSn | R | 0h | If this bit is set, indicates that the device was reset by NMIWDRSn.<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: PORn |

**Table 3-81. RESC Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | WDRSn | R | 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>Note:<br>[1] A bit inside WD module also provides the same information. This bit is present to keep things consistent. This register is a one-stop shop for the software to know the reset cause for the C28x core.<br>Reset type: PORn |
| 1 | XRSn | R | 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: PORn |
| 0 | POR | R | 1h | If this bit is set, indicates that the device was reset by PORn.<br>Writing a 1 to this bit will force the bit to 0<br>Writing of 0 will have no effect.<br>Reset type: PORn |

### 3.14.1.4 CPUTIMER_REGS Registers

Table 3-82 lists the memory-mapped registers for the CPUTIMER_REGS. All register offset addresses not listed in Table 3-82 should be considered as reserved locations and the register contents should not be modified.

**Table 3-82. CPUTIMER_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | TIM | CPU-Timer, Counter Register | | 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 3-83 shows the codes that are used for access types in this section.

**Table 3-83. CPUTIMER_REGS Access Type Codes**

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

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

TIM is shown in Figure 3-80 and described in Table 3-84.

Return to Summary Table.

CPU-Timer, Counter Register

**Figure 3-80. TIM Register**

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

**Table 3-84. TIM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | MSW | R/W | 0h | CPU-Timer Counter Registers<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 |

#### 3.14.1.4.2 PRD Register (Offset = 2h) [reset = 0001FFFFh]

PRD is shown in Figure 3-81 and described in Table 3-85.

Return to Summary Table.

CPU-Timer, Period Register

**Figure 3-81. PRD Register**

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

**Table 3-85. PRD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | MSW | R/W | 1h | CPU-Timer Period Registers<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 |

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

TCR is shown in Figure 3-82 and described in Table 3-86.

Return to Summary Table.

CPU-Timer, Control Register

**Figure 3-82. TCR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| TIF | TIE | RESERVED | | FREE | SOFT | RESERVED | |
| R/W1C-0h | R/W-0h | R-0h | | R/W-0h | R/W-0h | R-0h | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| RESERVED | | TRB | TSS | RESERVED | | | |
| R-0h | | R/W-0h | R/W-0h | R-1h | | | |

**Table 3-86. TCR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | TIF | R/W1C | 0h | 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>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>2h (R/W) = Free run<br>3h (R/W) = Free run |
| 10 | SOFT | R/W | 0h | If the FREE bit is set to 1, then, upon a software breakpoint, the timer continues to run (that is, free runs). In this case, SOFT is a don't care. But if FREE is 0, then SOFT takes effect. In this case, if SOFT = 0, the timer halts the next time the TIMH:TIM decrements. If the SOFT bit is 1, then the timer halts when the TIMH:TIM<br>has decremented to zero.<br>Reset type: SYSRSn |
| 9-6 | RESERVED | R | 0h | Reserved |

**Table 3-86. TCR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | TRB | R/W | 0h | Timer reload<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 |

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

TPR is shown in Figure 3-83 and described in Table 3-87.

Return to Summary Table.

CPU-Timer, Prescale Register

**Figure 3-83. TPR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| PSC | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TDDR | | | | | | | |
| R/W-0h | | | | | | | |

**Table 3-87. TPR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | PSC | R | 0h | 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<br>(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<br>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 |

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

TPRH is shown in Figure 3-84 and described in Table 3-88.

Return to Summary Table.

CPU-Timer, Prescale Register High

**Figure 3-84. TPRH Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| PSCH | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TDDRH | | | | | | | |
| R/W-0h | | | | | | | |

**Table 3-88. TPRH Register Field Descriptions**

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

### 3.14.1.5 DCC_REGS Registers

Table 6-2 lists the memory-mapped registers for the DCC_REGS. All register offset addresses not listed in Table 6-2 should be considered as reserved locations and the register contents should not be modified.

**Table 3-89. DCC_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | DCCGCTRL | Global Control Register | | Go |
| 4h | DCCREV | DCC Revision Register | | Go |
| 8h | DCCCNTSEED0 | Counter 0 Seed Value | | Go |
| Ch | DCCVALIDSEED0 | Valid 0 Seed Value | | Go |
| 10h | DCCCNTSEED1 | Counter 1 Seed Value | | Go |
| 14h | DCCSTATUS | DCC Status | | Go |
| 18h | DCCCNT0 | Counter 0 Value | | Go |
| 1Ch | DCCVALID0 | Valid Value 0 | | Go |
| 20h | DCCCNT1 | Counter 1 Value | | Go |
| 24h | DCCCLKSRC1 | Clock Source 1 | | Go |
| 28h | DCCCLKSRC0 | Clock Source 0 | | Go |

Complex bit access types are encoded to fit into small table cells. Table 6-3 shows the codes that are used for access types in this section.

**Table 3-90. DCC_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| R=1 | 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. |

#### 3.14.1.5.1 DCCGCTRL Register (Offset = 0h) [reset = 5555h]

DCCGCTRL is shown in Figure 6-7 and described in Table 6-4.

Return to Summary Table.

Global Control Register

**Figure 3-85. DCCGCTRL Register**

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

**Table 3-91. DCCGCTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R/W | 0h | Reserved |
| 15-12 | DONEENA | R/W | 5h | DONE Enable<br>Enables/disables the done interrupt signal, but has no effect on the done status flag in DCCSTAT register.<br>0101 The done signal is disabled<br>Others The done signal is enabled<br>Reset type: SYSRSn |
| 11-8 | SINGLESHOT | R/W | 5h | Single-Shot Enable<br>Enables/disables repetitive operation of the DCC.<br>1010 Stop counting when COUNTER0 and VALID0 both reach zero<br>Note: Configure this to 0xA before Enabling DCC<br>Note: All values other than 1010 are reserved<br>Reset type: SYSRSn |
| 7-4 | ERRENA | R/W | 5h | Error Enable<br>Enables/disables the error signal.<br>0101 The error signal is disabled<br>Others The error signal is enabled<br>Reset type: SYSRSn |
| 3-0 | DCCENA | R/W | 5h | DCC Enable<br>Starts and stops the operation of the DCC.<br>0101 Counters are stopped<br>Others Counters are running<br>Reset type: SYSRSn |

#### 3.14.1.5.2 DCCREV Register (Offset = 4h) [reset = 40041003h]

DCCREV is shown in Figure 6-8 and described in Table 6-5.

Return to Summary Table.

DCC Revision Register

**Figure 3-86. DCCREV Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | RESERVED | | 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 |
| RESERVED | | | | | MAJOR | | |
| R-0h | | | | | R-0h | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | MINOR | | | | | |
| R-0h | | R-3h | | | | | |

**Table 3-92. DCCREV Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | RESERVED | R | 0h | Reserved |
| 29-28 | RESERVED | R | 0h | Reserved |
| 27-16 | RESERVED | R | 0h | Reserved |
| 15-11 | RESERVED | R | 0h | Reserved |
| 10-8 | MAJOR | R | 0h | Major Revision Number<br>Represents major changes to the module (e.g. entirely new features are added/changed). The major revision number for this module.<br>Reset type: SYSRSn |
| 7-6 | RESERVED | R | 0h | Reserved |
| 5-0 | MINOR | R | 3h | Minor Revision Number<br>Represents minor changes to the module (e.g. enhancements to existing features). The minor revision number for this module.<br>Reset type: SYSRSn |

#### 3.14.1.5.3 DCCCNTSEED0 Register (Offset = 8h) [reset = 0h]

DCCCNTSEED0 is shown in Figure 6-9 and described in Table 6-6.

Return to Summary Table.

Counter 0 Seed Value

**Figure 3-87. DCCCNTSEED0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | COUNTSEED0 | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | R/W-0h | | | | | | | | | | | | | | | | | | | |

**Table 3-93. DCCCNTSEED0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | RESERVED | R | 0h | Reserved |
| 19-0 | COUNTSEED0 | R/W | 0h | Seed Value for Counter 0<br>Contains the seed value that gets loaded into Counter 0 (Clock Source 0).<br>NOTE: Operating the DCC with '0' in the COUNTSEED0 register will result in undefined operation.<br>Reset type: SYSRSn |

#### 3.14.1.5.4 DCCVALIDSEED0 Register (Offset = Ch) [reset = 0h]

DCCVALIDSEED0 is shown in Figure 6-10 and described in Table 6-7.

Return to Summary Table.

Valid 0 Seed Value

**Figure 3-88. DCCVALIDSEED0 Register**

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

**Table 3-94. DCCVALIDSEED0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-0 | VALIDSEED | R/W | 0h | Seed Value for Valid Duration Counter 0<br>Contains the seed value that gets loaded into the valid duration counter for Clock Source 0.<br>NOTE: Operating the DCC with '0' in the VALIDSEED0 register will result in undefined operation. VALID0 defines a window in which COUNT1 expires. This window is meant to be at least four cycles wide. Do not program a value less than '4' into the VALID0 register.<br>Reset type: SYSRSn |

#### 3.14.1.5.5 DCCCNTSEED1 Register (Offset = 10h) [reset = 0h]

DCCCNTSEED1 is shown in Figure 6-11 and described in Table 6-8.

Return to Summary Table.

Counter 1 Seed Value

**Figure 3-89. DCCCNTSEED1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | COUNTSEED1 | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | R/W-0h | | | | | | | | | | | | | | | | | | | |

**Table 3-95. DCCCNTSEED1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | RESERVED | R | 0h | Reserved |
| 19-0 | COUNTSEED1 | R/W | 0h | Seed Value for Counter 1<br>Contains the seed value that gets loaded into Counter 1 (Clock Source 1).<br>NOTE: Operating the DCC with '0' in the COUNTSEED1 register will result in undefined operation.<br>Reset type: SYSRSn |

#### 3.14.1.5.6 DCCSTATUS Register (Offset = 14h) [reset = 0h]

DCCSTATUS is shown in Figure 6-12 and described in Table 6-9.

Return to Summary Table.

DCC Status

**Figure 3-90. DCCSTATUS Register**

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

**Table 3-96. DCCSTATUS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-2 | RESERVED | R | 0h | Reserved |
| 1 | DONE | R/W | 0h | Single-Shot Done Flag<br>Indicates when single-shot mode is complete without error. Writing a '1' to this bit clears the flag.<br>0 Single-shot mode has not completed.<br>1 Single-shot mode has completed.<br>Reset type: SYSRSn |
| 0 | ERR | R/W | 0h | Error Flag<br>Indicates whether or not an error has occurred. Writing a '1' to this bit clears the flag.<br>0 No errors have occurred.<br>1 An error has occurred.<br>Reset type: SYSRSn |

#### 3.14.1.5.7 DCCCNT0 Register (Offset = 18h) [reset = 0h]

DCCCNT0 is shown in Figure 6-13 and described in Table 6-10.

Return to Summary Table.

Counter 0 Value

**Figure 3-91. DCCCNT0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | COUNT0 | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | | | | | |

**Table 3-97. DCCCNT0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | RESERVED | R | 0h | Reserved |
| 19-0 | COUNT0 | R | 0h | Current Value of Counter 0<br>NOTE: Reads of the counter value may not be exact since the read operation is synchronized to the register clock.<br>Reset type: SYSRSn |

#### 3.14.1.5.8 DCCVALID0 Register (Offset = 1Ch) [reset = 0h]

DCCVALID0 is shown in Figure 6-14 and described in Table 6-11.

Return to Summary Table.

Valid Value 0

**Figure 3-92. DCCVALID0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | VALID0 | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | |

**Table 3-98. DCCVALID0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-0 | VALID0 | R | 0h | Current Value of Valid 0<br>NOTE: Reads of the counter value may not be exact since the read operation is synchronized to the register clock.<br>Reset type: SYSRSn |

#### 3.14.1.5.9 DCCCNT1 Register (Offset = 20h) [reset = 0h]

DCCCNT1 is shown in Figure 6-15 and described in Table 6-12.

Return to Summary Table.

Counter 1 Value

**Figure 3-93. DCCCNT1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | COUNT1 | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | | | | | |

**Table 3-99. DCCCNT1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | RESERVED | R | 0h | Reserved |
| 19-0 | COUNT1 | R | 0h | Current Value of Counter 1<br>NOTE: Reads of the counter value may not be exact since the read operation is synchronized to the register clock.<br>Reset type: SYSRSn |

#### 3.14.1.5.10 DCCCLKSRC1 Register (Offset = 24h) [reset = 5002h]

DCCCLKSRC1 is shown in Figure 6-16 and described in Table 6-13.

Return to Summary Table.

Clock Source 1

**Figure 3-94. DCCCLKSRC1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| KEY | | | | RESERVED | | | | | | | | CLKSRC1 | | | |
| R/W-5h | | | | R-0h | | | | | | | | R/W-2h | | | |

**Table 3-100. DCCCLKSRC1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-12 | KEY | R/W | 5h | Enables or Disables Clock Source Selection for COUNT1<br>1010 The CLKSRC field selects the clock source for COUNT1.<br>Others Clock source selection is disabled. The secondary oscillator is selected for COUNT1.<br>Reset type: SYSRSn |
| 11-4 | RESERVED | R | 0h | Reserved |
| 3-0 | CLKSRC1 | R/W | 2h | Clock Source Select for Counter 1<br>Specifies the clock source for COUNT1, when the KEY field enables this feature.<br>0000 Clock Source 0 (PLLRAWCLK) is selected for COUNT1.<br>Note: Configure this value to 0x0 before enabling DCC<br>Note: All values other than 0000 are reserved.<br>Reset type: SYSRSn |

#### 3.14.1.5.11 DCCCLKSRC0 Register (Offset = 28h) [reset = 1h]

DCCCLKSRC0 is shown in Figure 6-17 and described in Table 6-14.

Return to Summary Table.

Clock Source 0

**Figure 3-95. DCCCLKSRC0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | | | | | | CLKSRC0 | | | |
| R-0h | | | | | | | | | | | | R/W-1h | | | |

**Table 3-101. DCCCLKSRC0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-4 | RESERVED | R | 0h | Reserved |
| 3-0 | CLKSRC0 | R/W | 1h | Clock Source Select for Counter 0<br>0000 Clock Source 0(XTAL) is selected for COUNT0.<br>0001 Clock Source 1(INTOSC1) is selected for COUNT0.<br>0010 Clock Source 2(INTOSC2) is selected for COUNT0.<br>Note: Reference Clock for the PLL (OSCCLKSRCSEL) and the DCCCLKSRC0 selected should be the same, meaning if OSCCLKSRCSEL=0x1 (XTAL) then DCCCLKSRC0.CLKRSC0=0x0 (XTAL).<br>Note: All other values are reserved.<br>Reset type: SYSRSn |

### 3.14.1.6 DCSM_BANK0_Z1_REGS Registers

Table 3-102 lists the memory-mapped registers for the DCSM_BANK0_Z1_REGS. All register offset addresses not listed in Table 3-102 should be considered as reserved locations and the register contents should not be modified.

**Table 3-102. DCSM_BANK0_Z1_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | B0_Z1_LINKPOINTER | Zone 1 Link Pointer for flash BANK0 | | Go |
| 2h | Z1_OTPSECLOCK | Zone 1 OTP Secure JTAG lock | | Go |
| 4h | Z1_BOOTDEF_HIGH | Boot definition (high 32bit) | | Go |
| 6h | B0_Z1_LINKPOINTERERR | Link Pointer Error for flash BANK0 | | Go |
| 8h | Z1_BOOTPIN_CONFIG | Boot Pin Configuration | | Go |
| Ah | Z1_GPREG2 | Zone1 General Purpose Register-2 | | Go |
| Ch | Z1_BOOTDEF_LOW | Boot definition (low 32bit) | | 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 | B0_Z1_GRABSECTR | Zone 1 Grab Flash BANK0 Sectors Register | | Go |
| 1Ch | Z1_GRABRAMR | Zone 1 Grab RAM Blocks Register | | Go |
| 1Eh | B0_Z1_EXEONLYSECTR | Zone 1 Flash BANK0 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 3-103 shows the codes that are used for access types in this section.

**Table 3-103. DCSM_BANK0_Z1_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | 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. |

#### 3.14.1.6.1 B0_Z1_LINKPOINTER Register (Offset = 0h) [reset = E0000000h]

B0_Z1_LINKPOINTER is shown in Figure 3-96 and described in Table 3-104.

Return to Summary Table.

Zone 1 Link Pointer for flash BANK0

**Figure 3-96. B0_Z1_LINKPOINTER Register**

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

**Table 3-104. B0_Z1_LINKPOINTER Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-29 | RESERVED | R | 7h | Reserved |
| 28-0 | LINKPOINTER | R | 0h | This is resolved Link-Pointer for Zone1 zone select block USER OTP of Flash BANK0. This is generated by using three physical Link-Pointer values loaded from OTP in Flash BANK0.<br>Reset type: SYSRSn |

#### 3.14.1.6.2 Z1_OTPSECLOCK Register (Offset = 2h) [reset = Fh]

Z1_OTPSECLOCK is shown in Figure 3-97 and described in Table 3-105.

Return to Summary Table.

Zone 1 OTP Secure JTAG lock

**Figure 3-97. 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-0h | | | | R-0h | | | | R-Fh | | | |

**Table 3-105. Z1_OTPSECLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-8 | CRCLOCK | R | 0h | Value in this field gets loaded from Z1OTP_CRCLOCK[3:0] when a read is issued to address location of Z1OTP_CRCLOCK in OTP.<br>1111 : VCU has ability to calculate CRC on secure memories.<br>Other Value : VCU does not have ability to calculate CRC on secure memories.<br>Reset type: XRSn |
| 7-4 | PSWDLOCK | R | 0h | Value in this field gets loaded from Z1OTP_PSWDLOCK[3:0] when a read is issued to address location of Z1OTP_PSWDLOCK in OTP.<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 not be read without unlocking Zone1.<br>Reset type: XRSn |
| 3-0 | JTAGLOCK | R | Fh | Value in this field gets loaded from Z1OTP_JTAGLOCK[3:0] when a read is issued to address location of Z1OTP_JTAGLOCK in OTP.<br>1111 : JTAG/Emulation access is allowed.<br>Other Value : JTAG/Emulation access not allowed.<br>Reset type: XRSn |

#### 3.14.1.6.3 Z1_BOOTDEF_HIGH Register (Offset = 4h) [reset = 0h]

Z1_BOOTDEF_HIGH is shown in Figure 3-98 and described in Table 3-106.

Return to Summary Table.

Boot definition (high 32bit)

**Figure 3-98. Z1_BOOTDEF_HIGH Register**

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

**Table 3-106. Z1_BOOTDEF_HIGH Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | BOOTDEF_HIGH | R | 0h | Refer ROM Code and Peripheral Booting section of TRM.<br>Reset type: SYSRSn |

#### 3.14.1.6.4 B0_Z1_LINKPOINTERERR Register (Offset = 6h) [reset = 0h]

B0_Z1_LINKPOINTERERR is shown in Figure 3-99 and described in Table 3-107.

Return to Summary Table.

Link Pointer Error for flash BANK0

**Figure 3-99. B0_Z1_LINKPOINTERERR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | Z1_LINKPOINTERERR | | | | | | | | | | | | |
| R=0-0h | | | R-0h | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Z1_LINKPOINTERERR | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 3-107. B0_Z1_LINKPOINTERERR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-29 | RESERVED | R=0 | 0h | Reserved |
| 28-0 | Z1_LINKPOINTERERR | R | 0h | These bits indicate errors during formation of the resolved Link-Pointer value after the three physical Link-Pointer values loaded from USER OTP in Flash BANK0<br>0 : No Error.<br>Other : Error on bit positions which is set to 1.<br>Reset type: SYSRSn |

#### 3.14.1.6.5 Z1_BOOTPIN_CONFIG Register (Offset = 8h) [reset = 0h]

Z1_BOOTPIN_CONFIG is shown in Figure 3-100 and described in Table 3-108.

Return to Summary Table.

Boot Pin Configuration

**Figure 3-100. Z1_BOOTPIN_CONFIG Register**

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

**Table 3-108. Z1_BOOTPIN_CONFIG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | BOOTPIN_CONIFG | R | 0h | Refer ROM Code and Peripheral Booting section of TRM.<br>Reset type: SYSRSn |

#### 3.14.1.6.6 Z1_GPREG2 Register (Offset = Ah) [reset = 0h]

Z1_GPREG2 is shown in Figure 3-101 and described in Table 3-109.

Return to Summary Table.

Zone1 General Purpose Register-2

**Figure 3-101. Z1_GPREG2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPREG2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-109. Z1_GPREG2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | GPREG2 | R | 0h | Refer ROM Code and Peripheral Booting section of TRM.<br>Reset type: SYSRSn |

#### 3.14.1.6.7 Z1_BOOTDEF_LOW Register (Offset = Ch) [reset = 0h]

Z1_BOOTDEF_LOW is shown in Figure 3-102 and described in Table 3-110.

Return to Summary Table.

Boot definition (low 32bit)

**Figure 3-102. Z1_BOOTDEF_LOW Register**

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

**Table 3-110. Z1_BOOTDEF_LOW Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | BOOTDEF_LOW | R | 0h | Refer ROM Code and Peripheral Booting section of TRM.<br>Reset type: SYSRSn |

#### 3.14.1.6.8 Z1_CSMKEY0 Register (Offset = 10h) [reset = 0h]

Z1_CSMKEY0 is shown in Figure 3-103 and described in Table 3-111.

Return to Summary Table.

Zone 1 CSM Key 0

**Figure 3-103. 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 3-111. Z1_CSMKEY0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z1_CSMKEY0 | R | 0h | To unlock Zone1, user needs to write this regsiter with exact value as Z1_CSMPSWD0, programmed in USER OTP (zone gets unlock only when 128 bit password in USER OTP match with value written in four CSMKEY registers.)<br>Reset type: SYSRSn |

#### 3.14.1.6.9 Z1_CSMKEY1 Register (Offset = 12h) [reset = 0h]

Z1_CSMKEY1 is shown in Figure 3-104 and described in Table 3-112.

Return to Summary Table.

Zone 1 CSM Key 1

**Figure 3-104. 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 3-112. Z1_CSMKEY1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z1_CSMKEY1 | R | 0h | To unlock Zone1, user needs to write this regsiter with exact value as Z1_CSMPSWD1, programmed in USER OTP (zone gets unlock only when 128 bit password in USER OTP match with value written in four CSMKEY registers.)<br>Reset type: SYSRSn |

#### 3.14.1.6.10 Z1_CSMKEY2 Register (Offset = 14h) [reset = 0h]

Z1_CSMKEY2 is shown in Figure 3-105 and described in Table 3-113.

Return to Summary Table.

Zone 1 CSM Key 2

**Figure 3-105. 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 3-113. Z1_CSMKEY2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z1_CSMKEY2 | R | 0h | To unlock Zone1, user needs to write this regsiter with exact value as Z1_CSMPSWD2, programmed in USER OTP (zone gets unlock only when 128 bit password in USER OTP match with value written in four CSMKEY registers.)<br>Reset type: SYSRSn |

#### 3.14.1.6.11 Z1_CSMKEY3 Register (Offset = 16h) [reset = 0h]

Z1_CSMKEY3 is shown in Figure 3-106 and described in Table 3-114.

Return to Summary Table.

Zone 1 CSM Key 3

**Figure 3-106. 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 3-114. Z1_CSMKEY3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z1_CSMKEY3 | R | 0h | To unlock Zone1, user needs to write this regsiter with exact value as Z1_CSMPSWD3, programmed in USER OTP (zone gets unlock only when 128 bit password in USER OTP match with value written in four CSMKEY registers.)<br>Reset type: SYSRSn |

#### 3.14.1.6.12 Z1_CR Register (Offset = 19h) [reset = 8h]

Z1_CR is shown in Figure 3-107 and described in Table 3-115.

Return to Summary Table.

Zone 1 CSM Control Register

**Figure 3-107. 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 3-115. 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 USER OTP has not been performed.<br>1 : Dummy read to CSM Password locations in USER OTP has been performed.<br>Reset type: SYSRSn |
| 5 | UNSECURE | R | 0h | Indiacates the state of Zone1.<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 passwords.<br>0 : Zone CSM Passwords are not all ones.<br>1 : Zone CSM Passwords are all ones and device is permanently blocked.<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 |

#### 3.14.1.6.13 B0_Z1_GRABSECTR Register (Offset = 1Ah) [reset = 0h]

B0_Z1_GRABSECTR is shown in Figure 3-108 and described in Table 3-116.

Return to Summary Table.

Zone 1 Grab Flash BANK0 Sectors Register

**Figure 3-108. B0_Z1_GRABSECTR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GRAB_SECT15 | | GRAB_SECT14 | | GRAB_SECT13 | | GRAB_SECT12 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GRAB_SECT11 | | GRAB_SECT10 | | GRAB_SECT9 | | GRAB_SECT8 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GRAB_SECT7 | | GRAB_SECT6 | | GRAB_SECT5 | | GRAB_SECT4 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GRAB_SECT3 | | GRAB_SECT2 | | GRAB_SECT1 | | GRAB_SECT0 | |
| R-0h | | R-0h | | R-0h | | R-0h | |

**Table 3-116. B0_Z1_GRABSECTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GRAB_SECT15 | R | 0h | Value in this field gets loaded from B0_Z1OTP_GRABSECT[31:30] when a read is issued to address location of B0_Z1OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 15 is inaccessible.<br>01 : Request to allocate Flash Sector 15 to Zone1.<br>10 : No request for Flash Sector 15<br>11 : No request for Flash Sector 15 when this zone is UNLOCKED. Else Flash Sector 15 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 29-28 | GRAB_SECT14 | R | 0h | Value in this field gets loaded from B0_Z1OTP_GRABSECT[29:28] when a read is issued to address location of B0_Z1OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 14 is inaccessible.<br>01 : Request to allocate Flash Sector 14 to Zone1.<br>10 : No request for Flash Sector 14<br>11 : No request for Flash Sector 14 when this zone is UNLOCKED. Else Flash Sector 14 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 27-26 | GRAB_SECT13 | R | 0h | Value in this field gets loaded from B0_Z1OTP_GRABSECT[27:26] when a read is issued to address location of B0_Z1OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 13 is inaccessible.<br>01 : Request to allocate Flash Sector 13 to Zone1.<br>10 : No request for Flash Sector 13<br>11 : No request for Flash Sector 13 when this zone is UNLOCKED. Else Flash Sector 13 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |

**Table 3-116. B0_Z1_GRABSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 25-24 | GRAB_SECT12 | R | 0h | Value in this field gets loaded from B0_Z1OTP_GRABSECT[25:24] when a read is issued to address location of B0_Z1OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 12 is inaccessible.<br>01 : Request to allocate Flash Sector 12 to Zone1.<br>10 : No request for Flash Sector 12<br>11 : No request for Flash Sector 12 when this zone is UNLOCKED. Else Flash Sector 12 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 23-22 | GRAB_SECT11 | R | 0h | Value in this field gets loaded from B0_Z1OTP_GRABSECT[23:22] when a read is issued to address location of B0_Z1OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 11 is inaccessible.<br>01 : Request to allocate Flash Sector 11 to Zone1.<br>10 : No request for Flash Sector 11<br>11 : No request for Flash Sector 11 when this zone is UNLOCKED. Else Flash Sector 11 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 21-20 | GRAB_SECT10 | R | 0h | Value in this field gets loaded from B0_Z1OTP_GRABSECT[21:20] when a read is issued to address location of B0_Z1OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 10 is inaccessible.<br>01 : Request to allocate Flash Sector 10 to Zone1.<br>10 : No request for Flash Sector 10<br>11 : No request for Flash Sector 10 when this zone is UNLOCKED. Else Flash Sector 10 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 19-18 | GRAB_SECT9 | R | 0h | Value in this field gets loaded from B0_Z1OTP_GRABSECT[19:18] when a read is issued to address location of B0_Z1OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 9 is inaccessible.<br>01 : Request to allocate Flash Sector 9 to Zone1.<br>10 : No request for Flash Sector 9<br>11 : No request for Flash Sector 9 when this zone is UNLOCKED. Else Flash Sector 9 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 17-16 | GRAB_SECT8 | R | 0h | Value in this field gets loaded from B0_Z1OTP_GRABSECT[17:16] when a read is issued to address location of B0_Z1OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 8 is inaccessible.<br>01 : Request to allocate Flash Sector 8 to Zone1.<br>10 : No request for Flash Sector 8<br>11 : No request for Flash sector 8 when this zone is UNLOCKED. Else Flash sector 8 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |

**Table 3-116. B0_Z1_GRABSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | GRAB_SECT7 | R | 0h | Value in this field gets loaded from B0_Z1OTP_GRABSECT[15:14] when a read is issued to address location of B0_Z1OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 7 is inaccessible.<br>01 : Request to allocate Flash Sector 7 to Zone1.<br>10 : No request for Flash Sector 7<br>11 : No request for Flash Sector 7 when this zone is UNLOCKED. Else Flash Sector 7 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 13-12 | GRAB_SECT6 | R | 0h | Value in this field gets loaded from B0_Z1OTP_GRABSECT[13:12] when a read is issued to address location of B0_Z1OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 6 is inaccessible.<br>01 : Request to allocate Flash Sector 6 to Zone1.<br>10 : No request for Flash Sector 6<br>11 : No request for Flash Sector 6 when this zone is UNLOCKED. Else Flash Sector 6 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 11-10 | GRAB_SECT5 | R | 0h | Value in this field gets loaded from B0_Z1OTP_GRABSECT[11:10] when a read is issued to address location of B0_Z1OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 5 is inaccessible.<br>01 : Request to allocate Flash Sector 5 to Zone1.<br>10 : No request for Flash Sector 5<br>11 : No request for Flash Sector 5 when this zone is UNLOCKED. Else Flash Sector 5 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 9-8 | GRAB_SECT4 | R | 0h | Value in this field gets loaded from B0_Z1OTP_GRABSECT[9:8] when a read is issued to address location of B0_Z1OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 4 is inaccessible.<br>01 : Request to allocate Flash Sector 4 to Zone1.<br>10 : No request for Flash Sector 4<br>11 : No request for Flash Sector 4 when this zone is UNLOCKED. Else Flash Sector 4 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 7-6 | GRAB_SECT3 | R | 0h | Value in this field gets loaded from B0_Z1OTP_GRABSECT[7:6] when a read is issued to address location of B0_Z1OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 3 is inaccessible.<br>01 : Request to allocate Flash Sector 3 to Zone1.<br>10 : No request for Flash Sector 3<br>11 : No request for Flash Sector 3 when this zone is UNLOCKED. Else Flash Sector 3 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |

**Table 3-116. B0_Z1_GRABSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5-4 | GRAB_SECT2 | R | 0h | Value in this field gets loaded from B0_Z1OTP_GRABSECT[5:4] when a read is issued to address location of B0_Z1OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 2 is inaccessible.<br>01 : Request to allocate Flash Sector 2 to Zone1.<br>10 : No request for Flash Sector 2<br>11 : No request for Flash Sector 2 when this zone is UNLOCKED. Else Flash Sector 2 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 3-2 | GRAB_SECT1 | R | 0h | Value in this field gets loaded from B0_Z1OTP_GRABSECT[3:2] when a read is issued to address location of B0_Z1OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 1 is inaccessible.<br>01 : Request to allocate Flash Sector 1 to Zone1.<br>10 : No request for Flash Sector 1<br>11 : No request for Flash sector 1 when this zone is UNLOCKED. Else Flash sector 1 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 1-0 | GRAB_SECT0 | R | 0h | Value in this field gets loaded from B0_Z1OTP_GRABSECT[1:0] when a read is issued to address location of B0_Z1OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 0 is inaccessible.<br>01 : Request to allocate Flash Sector 0 to Zone1.<br>10 : No request for Flash Sector 0<br>11 : No request for Flash Sector 0 when this zone is UNLOCKED. Else Flash Sector 0 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |

#### 3.14.1.6.14 Z1_GRABRAMR Register (Offset = 1Ch) [reset = 0h]

Z1_GRABRAMR is shown in Figure 3-109 and described in Table 3-117.

Return to Summary Table.

Zone 1 Grab RAM Blocks Register

**Figure 3-109. Z1_GRABRAMR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GRAB_RAM7 | | GRAB_RAM6 | | GRAB_RAM5 | | GRAB_RAM4 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GRAB_RAM3 | | GRAB_RAM2 | | GRAB_RAM1 | | GRAB_RAM0 | |
| R-0h | | R-0h | | R-0h | | R-0h | |

**Table 3-117. Z1_GRABRAMR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-14 | GRAB_RAM7 | R | 0h | Value in this field gets loaded from Z1OTP_GRABRAM[15:14] when a read is issued to address location of Z1OTP_GRABRAM in USER OTP.<br>00 : Invalid. LS7 RAM is inaccessible.<br>01 : Request to allocate LS7 RAM to Zone1.<br>10 : No request for LS7 RAM<br>11 : No request for LS7 RAM when this zone is UNLOCKED. Else LS7 RAM is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 13-12 | GRAB_RAM6 | R | 0h | Value in this field gets loaded from Z1OTP_GRABRAM[13:12] when a read is issued to address location of Z1OTP_GRABRAM in USER OTP.<br>00 : Invalid. LS6 RAM is inaccessible.<br>01 : Request to allocate LS6 RAM to Zone1.<br>10 : No request for LS6 RAM<br>11 : No request for LS6 RAM when this zone is UNLOCKED. Else LS6 RAM is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 11-10 | GRAB_RAM5 | R | 0h | Value in this field gets loaded from Z1OTP_GRABRAM[11:10] when a read is issued to address location of Z1OTP_GRABRAM in USER OTP.<br>00 : Invalid. LS5 RAM is inaccessible.<br>01 : Request to allocate LS5 RAM to Zone1.<br>10 : No request for LS5 RAM<br>11 : No request for LS5 RAM when this zone is UNLOCKED. Else LS5 RAM is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |

**Table 3-117. Z1_GRABRAMR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9-8 | GRAB_RAM4 | R | 0h | Value in this field gets loaded from Z1OTP_GRABRAM[9:8] when a read is issued to address location of Z1OTP_GRABRAM in USER OTP.<br>00 : Invalid. LS4 RAM is inaccessible.<br>01 : Request to allocate LS4 RAM to Zone1.<br>10 : No request for LS4 RAM<br>11 : No request for LS4 RAM when this zone is UNLOCKED. Else LS4 RAM is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 7-6 | GRAB_RAM3 | R | 0h | Value in this field gets loaded from Z1OTP_GRABRAM[7:6] when a read is issued to address location of Z1OTP_GRABRAM in USER OTP.<br>00 : Invalid. LS3 RAM is inaccessible.<br>01 : Request to allocate LS3 RAM to Zone1.<br>10 : No request for LS3 RAM<br>11 : No request for LS3 RAM when this zone is UNLOCKED. Else LS3 RAM is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 5-4 | GRAB_RAM2 | R | 0h | Value in this field gets loaded from Z1OTP_GRABRAM[5:4] when a read is issued to address location of Z1OTP_GRABRAM in USER OTP.<br>00 : Invalid. LS2 RAM is inaccessible.<br>01 : Request to allocate LS2 RAM to Zone1.<br>10 : No request for LS2 RAM<br>11 : No request for LS2 RAM when this zone is UNLOCKED. Else LS2 RAM is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 3-2 | GRAB_RAM1 | R | 0h | Value in this field gets loaded from Z1OTP_GRABRAM[3:2] when a read is issued to address location of Z1OTP_GRABRAM in USER OTP.<br>00 : Invalid. LS1 RAM is inaccessible.<br>01 : Request to allocate LS1 RAM to Zone1.<br>10 : No request for LS1 RAM<br>11 : No request for LS1 RAM when this zone is UNLOCKED. Else LS1 RAM is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 1-0 | GRAB_RAM0 | R | 0h | Value in this field gets loaded from Z1OTP_GRABRAM[1:0] when a read is issued to address location of Z1OTP_GRABRAM in USER OTP.<br>00 : Invalid. LS0 RAM is inaccessible.<br>01 : Request to allocate LS0 RAM to Zone1.<br>10 : No request for LS0 RAM<br>11 : No request for LS0 RAM when this zone is UNLOCKED. Else LS0 RAM is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |

#### 3.14.1.6.15 B0_Z1_EXEONLYSECTR Register (Offset = 1Eh) [reset = 0h]

B0_Z1_EXEONLYSECTR is shown in Figure 3-110 and described in Table 3-118.

Return to Summary Table.

Zone 1 Flash BANK0 Execute_Only Sector Register

**Figure 3-110. B0_Z1_EXEONLYSECTR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| EXEONLY_SECT15 | EXEONLY_SECT14 | EXEONLY_SECT13 | EXEONLY_SECT12 | EXEONLY_SECT11 | EXEONLY_SECT10 | EXEONLY_SECT9 | EXEONLY_SECT8 |
| 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_SECT7 | EXEONLY_SECT6 | EXEONLY_SECT5 | EXEONLY_SECT4 | EXEONLY_SECT3 | EXEONLY_SECT2 | EXEONLY_SECT1 | EXEONLY_SECT0 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 3-118. B0_Z1_EXEONLYSECTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15 | EXEONLY_SECT15 | R | 0h | Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[15:15] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 15 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 15 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 14 | EXEONLY_SECT14 | R | 0h | Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[14:14] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 14 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 14 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 13 | EXEONLY_SECT13 | R | 0h | Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[13:13] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 13 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 13 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |

**Table 3-118. B0_Z1_EXEONLYSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 12 | EXEONLY_SECT12 | R | 0h | Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[12:12] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 12 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 12 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 11 | EXEONLY_SECT11 | R | 0h | Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[11:11] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 11 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 11 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 10 | EXEONLY_SECT10 | R | 0h | Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[10:10] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 10 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 10 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 9 | EXEONLY_SECT9 | R | 0h | Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[9:9] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 9 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 9 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 8 | EXEONLY_SECT8 | R | 0h | Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[8:8] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 8 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 8 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 7 | EXEONLY_SECT7 | R | 0h | Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[7:7] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 7 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 7 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |

**Table 3-118. B0_Z1_EXEONLYSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | EXEONLY_SECT6 | R | 0h | Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[6:6] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 6 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 6 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 5 | EXEONLY_SECT5 | R | 0h | Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[5:5] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 5 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 5 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 4 | EXEONLY_SECT4 | R | 0h | Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[4:4] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 4 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 4 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 3 | EXEONLY_SECT3 | R | 0h | Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[3:3] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 3 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 3 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 2 | EXEONLY_SECT2 | R | 0h | Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[2:2] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 2 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 2 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 1 | EXEONLY_SECT1 | R | 0h | Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[1:1] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 1 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 1 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |

**Table 3-118. B0_Z1_EXEONLYSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | EXEONLY_SECT0 | R | 0h | Value in this field gets loaded from B0_Z1OTP_EXEONLYSECT[0:0] when a read is issued to B0_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 0 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 0 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |

#### 3.14.1.6.16 Z1_EXEONLYRAMR Register (Offset = 20h) [reset = 0h]

Z1_EXEONLYRAMR is shown in Figure 3-111 and described in Table 3-119.

Return to Summary Table.

Zone 1 RAM Execute_Only Block Register

**Figure 3-111. 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 3-119. Z1_EXEONLYRAMR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | EXEONLY_RAM7 | R | 0h | Value in this field gets loaded from Z1OTP_EXEONLYRAM[7:7] when a read is issued to Z1OTP_EXEONLYRAM address location in USER OTP.<br>0 : Execute-Only protection is enabled for LS7 RAM (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for LS7 RAM (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 6 | EXEONLY_RAM6 | R | 0h | Value in this field gets loaded from Z1OTP_EXEONLYRAM[6:6] when a read is issued to Z1OTP_EXEONLYRAM address location in USER OTP.<br>0 : Execute-Only protection is enabled for LS6 RAM (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for LS6 RAM (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 5 | EXEONLY_RAM5 | R | 0h | Value in this field gets loaded from Z1OTP_EXEONLYRAM[5:5] when a read is issued to Z1OTP_EXEONLYRAM address location in USER OTP.<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 |

**Table 3-119. Z1_EXEONLYRAMR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | EXEONLY_RAM4 | R | 0h | Value in this field gets loaded from Z1OTP_EXEONLYRAM[4:4] when a read is issued to Z1OTP_EXEONLYRAM address location in USER 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 |
| 3 | EXEONLY_RAM3 | R | 0h | Value in this field gets loaded from Z1OTP_EXEONLYRAM[3:3] when a read is issued to Z1OTP_EXEONLYRAM address location in USER 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 Z1OTP_EXEONLYRAM[2:2] when a read is issued to Z1OTP_EXEONLYRAM address location in USER 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 Z1OTP_EXEONLYRAM[1:1] when a read is issued to Z1OTP_EXEONLYRAM address location in USER 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 Z1OTP_EXEONLYRAM[0:0] when a read is issued to Z1OTP_EXEONLYRAM address location in USER 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 |

### 3.14.1.7 DCSM_BANK0_Z2_REGS Registers

Table 3-120 lists the memory-mapped registers for the DCSM_BANK0_Z2_REGS. All register offset addresses not listed in Table 3-120 should be considered as reserved locations and the register contents should not be modified.

**Table 3-120. DCSM_BANK0_Z2_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | B0_Z2_LINKPOINTER | Zone 2 Link Pointer for flash BANK0 | | Go |
| 2h | Z2_OTPSECLOCK | Zone 2 OTP Secure JTAG lock | | Go |
| 6h | B0_Z2_LINKPOINTERERR | Link Pointer Error for flash BANK0 | | 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 | B0_Z2_GRABSECTR | Zone 2 Grab Flash BANK0 Sectors Register | | Go |
| 1Ch | Z2_GRABRAMR | Zone 2 Grab RAM Blocks Register | | Go |
| 1Eh | B0_Z2_EXEONLYSECTR | Zone 2 Flash BANK0 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 3-121 shows the codes that are used for access types in this section.

**Table 3-121. DCSM_BANK0_Z2_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | 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. |

#### 3.14.1.7.1 B0_Z2_LINKPOINTER Register (Offset = 0h) [reset = E0000000h]

B0_Z2_LINKPOINTER is shown in Figure 3-112 and described in Table 3-122.

Return to Summary Table.

Zone 2 Link Pointer for flash BANK0

**Figure 3-112. B0_Z2_LINKPOINTER Register**

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

**Table 3-122. B0_Z2_LINKPOINTER Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-29 | RESERVED | R | 7h | Reserved |
| 28-0 | LINKPOINTER | R | 0h | This is resolved Link-Pointer for Zone2 zone select block USER OTP of Flash BANK0. This is generated by using three physical Link-Pointer values loaded from OTP in Flash BANK0.<br>Reset type: SYSRSn |

#### 3.14.1.7.2 Z2_OTPSECLOCK Register (Offset = 2h) [reset = Fh]

Z2_OTPSECLOCK is shown in Figure 3-113 and described in Table 3-123.

Return to Summary Table.

Zone 2 OTP Secure JTAG lock

**Figure 3-113. 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-0h | | | | R-0h | | | | R-Fh | | | |

**Table 3-123. Z2_OTPSECLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-8 | CRCLOCK | R | 0h | Value in this field gets loaded from Z2_CRCLOCK[3:0] when a read is issued to address location of Z2OTP_CRCLOCK in OTP.<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: XRSn |
| 7-4 | PSWDLOCK | R | 0h | Value in this field gets loaded from Z2_PSWDLOCK[3:0] when a read is issued to address location of Z2OTP_PSWDLOCK in OTP.<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: XRSn |
| 3-0 | JTAGLOCK | R | Fh | Value in this field gets loaded from Z2OTP_JATGLOCK[3:0] when a read is issued to address location of Z2_JATGLOCK in OTP.<br>1111 : JTAG/Emulation access is allowed.<br>Other Value : JTAG/Emulation access not allowed.<br>Reset type: XRSn |

#### 3.14.1.7.3 B0_Z2_LINKPOINTERERR Register (Offset = 6h) [reset = 0h]

B0_Z2_LINKPOINTERERR is shown in Figure 3-114 and described in Table 3-124.

Return to Summary Table.

Link Pointer Error for flash BANK0

**Figure 3-114. B0_Z2_LINKPOINTERERR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | Z2_LINKPOINTERERR | | | | | | | | | | | | |
| R=0-0h | | | R-0h | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Z2_LINKPOINTERERR | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 3-124. B0_Z2_LINKPOINTERERR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-29 | RESERVED | R=0 | 0h | Reserved |
| 28-0 | Z2_LINKPOINTERERR | R | 0h | These bits indicate errors during formation of the resolved Link-Pointer value after the three physical Link-Pointer values loaded from USER OTP in Flash BANK0<br>0 : No Error.<br>Other : Error on bit positions which is set to 1.<br>Reset type: SYSRSn |

#### 3.14.1.7.4 Z2_CSMKEY0 Register (Offset = 10h) [reset = 0h]

Z2_CSMKEY0 is shown in Figure 3-115 and described in Table 3-125.

Return to Summary Table.

Zone 2 CSM Key 0

**Figure 3-115. 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 3-125. Z2_CSMKEY0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z2_CSMKEY0 | R | 0h | To unlock Zone2 user needs to write this regsiter with exact value as Z2_CSMPSWD0, programmed in USER OTP (zone gets unlock only when 128 bit password in USER OTP match with value written in four CSMKEY registers.)<br>Reset type: SYSRSn |

#### 3.14.1.7.5 Z2_CSMKEY1 Register (Offset = 12h) [reset = 0h]

Z2_CSMKEY1 is shown in Figure 3-116 and described in Table 3-126.

Return to Summary Table.

Zone 2 CSM Key 1

**Figure 3-116. 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 3-126. Z2_CSMKEY1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z2_CSMKEY1 | R | 0h | To unlock Zone2 user needs to write this regsiter with exact value as Z2_CSMPSWD1, programmed in USER OTP (zone gets unlock only when 128 bit password in USER OTP match with value written in four CSMKEY registers.)<br>Reset type: SYSRSn |

#### 3.14.1.7.6 Z2_CSMKEY2 Register (Offset = 14h) [reset = 0h]

Z2_CSMKEY2 is shown in Figure 3-117 and described in Table 3-127.

Return to Summary Table.

Zone 2 CSM Key 2

**Figure 3-117. 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 3-127. Z2_CSMKEY2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z2_CSMKEY2 | R | 0h | To unlock Zone2 user needs to write this regsiter with exact value as Z2_CSMPSWD2, programmed in USER OTP (zone gets unlock only when 128 bit password in USER OTP match with value written in four CSMKEY registers.)<br>Reset type: SYSRSn |

#### 3.14.1.7.7 Z2_CSMKEY3 Register (Offset = 16h) [reset = 0h]

Z2_CSMKEY3 is shown in Figure 3-118 and described in Table 3-128.

Return to Summary Table.

Zone 2 CSM Key 3

**Figure 3-118. 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 3-128. Z2_CSMKEY3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Z2_CSMKEY3 | R | 0h | To unlock Zone2 user needs to write this regsiter with exact value as Z2_CSMPSWD3, programmed in USER OTP (zone gets unlock only when 128 bit password in USER OTP match with value written in four CSMKEY registers.)<br>Reset type: SYSRSn |

### 3.14.1.7.8 Z2_CR Register (Offset = 19h) [reset = 8h]

Z2_CR is shown in Figure 3-119 and described in Table 3-129.

Return to Summary Table.

Zone 2 CSM Control Register

**Figure 3-119. 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 3-129. 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 USER OTP has not been performed.<br>1 : Dummy read to CSM Password locations in USER 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 passwords.<br>0 : Zone CSM Passwords are not all ones.<br>1 : Zone CSM Passwords are all ones and device is permanently blocked.<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 |

#### 3.14.1.7.9 B0_Z2_GRABSECTR Register (Offset = 1Ah) [reset = 0h]

B0_Z2_GRABSECTR is shown in Figure 3-120 and described in Table 3-130.

Return to Summary Table.

Zone 2 Grab Flash BANK0 Sectors Register

**Figure 3-120. B0_Z2_GRABSECTR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GRAB_SECT15 | | GRAB_SECT14 | | GRAB_SECT13 | | GRAB_SECT12 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GRAB_SECT11 | | GRAB_SECT10 | | GRAB_SECT9 | | GRAB_SECT8 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GRAB_SECT7 | | GRAB_SECT6 | | GRAB_SECT5 | | GRAB_SECT4 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GRAB_SECT3 | | GRAB_SECT2 | | GRAB_SECT1 | | GRAB_SECT0 | |
| R-0h | | R-0h | | R-0h | | R-0h | |

**Table 3-130. B0_Z2_GRABSECTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GRAB_SECT15 | R | 0h | Value in this field gets loaded from B0_Z2OTP_GRABSECT[31:30] when a read is issued to address location of B0_Z2OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 15 is inaccessible.<br>01 : Request to allocate Flash Sector 15 to Zone2.<br>10 : No request for Flash Sector 15<br>11 : No request for Flash Sector 15 when this zone is UNLOCKED. Else Flash Sector 15 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 29-28 | GRAB_SECT14 | R | 0h | Value in this field gets loaded from B0_Z2OTP_GRABSECT[29:28] when a read is issued to address location of B0_Z2OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 14 is inaccessible.<br>01 : Request to allocate Flash Sector 14 to Zone2.<br>10 : No request for Flash Sector 14<br>11 : No request for Flash Sector 14 when this zone is UNLOCKED. Else Flash Sector 14 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 27-26 | GRAB_SECT13 | R | 0h | Value in this field gets loaded from B0_Z2OTP_GRABSECT[27:26] when a read is issued to address location of B0_Z2OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 13 is inaccessible.<br>01 : Request to allocate Flash Sector 13 to Zone2.<br>10 : No request for Flash Sector 13<br>11 : No request for Flash Sector 13 when this zone is UNLOCKED. Else Flash Sector 13 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |

**Table 3-130. B0_Z2_GRABSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 25-24 | GRAB_SECT12 | R | 0h | Value in this field gets loaded from B0_Z2OTP_GRABSECT[25:24] when a read is issued to address location of B0_Z2OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 12 is inaccessible.<br>01 : Request to allocate Flash Sector 12 to Zone2.<br>10 : No request for Flash Sector 12<br>11 : No request for Flash Sector 12 when this zone is UNLOCKED. Else Flash Sector 12 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 23-22 | GRAB_SECT11 | R | 0h | Value in this field gets loaded from B0_Z2OTP_GRABSECT[23:22] when a read is issued to address location of B0_Z2OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 11 is inaccessible.<br>01 : Request to allocate Flash Sector 11 to Zone2.<br>10 : No request for Flash Sector 11<br>11 : No request for Flash Sector 11 when this zone is UNLOCKED. Else Flash Sector 11 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 21-20 | GRAB_SECT10 | R | 0h | Value in this field gets loaded from B0_Z2OTP_GRABSECT[21:20] when a read is issued to address location of B0_Z2OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 10 is inaccessible.<br>01 : Request to allocate Flash Sector 10 to Zone2.<br>10 : No request for Flash Sector 10<br>11 : No request for Flash Sector 10 when this zone is UNLOCKED. Else Flash Sector 10 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 19-18 | GRAB_SECT9 | R | 0h | Value in this field gets loaded from B0_Z2OTP_GRABSECT[19:18] when a read is issued to address location of B0_Z2OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 9 is inaccessible.<br>01 : Request to allocate Flash Sector 9 to Zone2.<br>10 : No request for Flash Sector 9<br>11 : No request for Flash Sector 9 when this zone is UNLOCKED. Else Flash Sector 9 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 17-16 | GRAB_SECT8 | R | 0h | Value in this field gets loaded from B0_Z2OTP_GRABSECT[17:16] when a read is issued to address location of B0_Z2OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 8 is inaccessible.<br>01 : Request to allocate Flash Sector 8 to Zone2.<br>10 : No request for Flash Sector 8<br>11 : No request for Flash sector 8 when this zone is UNLOCKED. Else Flash sector 8 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |

**Table 3-130. B0_Z2_GRABSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | GRAB_SECT7 | R | 0h | Value in this field gets loaded from B0_Z2OTP_GRABSECT[15:14] when a read is issued to address location of B0_Z2OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 7 is inaccessible.<br>01 : Request to allocate Flash Sector 7 to Zone2.<br>10 : No request for Flash Sector 7<br>11 : No request for Flash Sector 7 when this zone is UNLOCKED. Else Flash Sector 7 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 13-12 | GRAB_SECT6 | R | 0h | Value in this field gets loaded from B0_Z2OTP_GRABSECT[13:12] when a read is issued to address location of B0_Z2OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 6 is inaccessible.<br>01 : Request to allocate Flash Sector 6 to Zone2.<br>10 : No request for Flash Sector 6<br>11 : No request for Flash Sector 6 when this zone is UNLOCKED. Else Flash Sector 6 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 11-10 | GRAB_SECT5 | R | 0h | Value in this field gets loaded from B0_Z2OTP_GRABSECT[11:10] when a read is issued to address location of B0_Z2OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 5 is inaccessible.<br>01 : Request to allocate Flash Sector 5 to Zone2.<br>10 : No request for Flash Sector 5<br>11 : No request for Flash Sector 5 when this zone is UNLOCKED. Else Flash Sector 5 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 9-8 | GRAB_SECT4 | R | 0h | Value in this field gets loaded from B0_Z2OTP_GRABSECT[9:8] when a read is issued to address location of B0_Z2OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 4 is inaccessible.<br>01 : Request to allocate Flash Sector 4 to Zone2.<br>10 : No request for Flash Sector 4<br>11 : No request for Flash Sector 4 when this zone is UNLOCKED. Else Flash Sector 4 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 7-6 | GRAB_SECT3 | R | 0h | Value in this field gets loaded from B0_Z2OTP_GRABSECT[7:6] when a read is issued to address location of B0_Z2OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 3 is inaccessible.<br>01 : Request to allocate Flash Sector 3 to Zone2.<br>10 : No request for Flash Sector 3<br>11 : No request for Flash Sector 3 when this zone is UNLOCKED. Else Flash Sector 3 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |

**Table 3-130. B0_Z2_GRABSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5-4 | GRAB_SECT2 | R | 0h | Value in this field gets loaded from B0_Z2OTP_GRABSECT[5:4] when a read is issued to address location of B0_Z2OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 2 is inaccessible.<br>01 : Request to allocate Flash Sector 2 to Zone2.<br>10 : No request for Flash Sector 2<br>11 : No request for Flash Sector 2 when this zone is UNLOCKED. Else Flash Sector 2 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 3-2 | GRAB_SECT1 | R | 0h | Value in this field gets loaded from B0_Z2OTP_GRABSECT[3:2] when a read is issued to address location of B0_Z2OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 1 is inaccessible.<br>01 : Request to allocate Flash Sector 1 to Zone2.<br>10 : No request for Flash Sector 1<br>11 : No request for Flash sector 1 when this zone is UNLOCKED. Else Flash sector 1 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 1-0 | GRAB_SECT0 | R | 0h | Value in this field gets loaded from B0_Z2OTP_GRABSECT[1:0] when a read is issued to address location of B0_Z2OTP_GRABSECT in USER OTP of Flash BANK0.<br>00 : Invalid. Flash Sector 0 is inaccessible.<br>01 : Request to allocate Flash Sector 0 to Zone2.<br>10 : No request for Flash Sector 0<br>11 : No request for Flash Sector 0 when this zone is UNLOCKED. Else Flash Sector 0 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |

#### 3.14.1.7.10 Z2_GRABRAMR Register (Offset = 1Ch) [reset = 0h]

Z2_GRABRAMR is shown in Figure 3-121 and described in Table 3-131.

Return to Summary Table.

Zone 2 Grab RAM Blocks Register

**Figure 3-121. Z2_GRABRAMR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GRAB_RAM7 | | GRAB_RAM6 | | GRAB_RAM5 | | GRAB_RAM4 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GRAB_RAM3 | | GRAB_RAM2 | | GRAB_RAM1 | | GRAB_RAM0 | |
| R-0h | | R-0h | | R-0h | | R-0h | |

**Table 3-131. Z2_GRABRAMR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-14 | GRAB_RAM7 | R | 0h | Value in this field gets loaded from Z2OTP_GRABRAM[15:14] when a read is issued to address location of Z2OTP_GRABRAM in USER OTP.<br>00 : Invalid. LS7 RAM is inaccessible.<br>01 : Request to allocate LS7 RAM to Zone2.<br>10 : No request for LS7 RAM<br>11 : No request for LS7 RAM when this zone is UNLOCKED. Else LS7 RAM is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 13-12 | GRAB_RAM6 | R | 0h | Value in this field gets loaded from Z2OTP_GRABRAM[13:12] when a read is issued to address location of Z2OTP_GRABRAM in USER OTP.<br>00 : Invalid. LS6 RAM is inaccessible.<br>01 : Request to allocate LS6 RAM to Zone2.<br>10 : No request for LS6 RAM<br>11 : No request for LS6 RAM when this zone is UNLOCKED. Else LS6 RAM is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 11-10 | GRAB_RAM5 | R | 0h | Value in this field gets loaded from Z2OTP_GRABRAM[11:10] when a read is issued to address location of Z2OTP_GRABRAM in USER OTP.<br>00 : Invalid. LS5 RAM is inaccessible.<br>01 : Request to allocate LS5 RAM to Zone2.<br>10 : No request for LS5 RAM<br>11 : No request for LS5 RAM when this zone is UNLOCKED. Else LS5 RAM is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |

**Table 3-131. Z2_GRABRAMR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9-8 | GRAB_RAM4 | R | 0h | Value in this field gets loaded from Z2OTP_GRABRAM[9:8] when a read is issued to address location of Z2OTP_GRABRAM in USER OTP.<br>00 : Invalid. LS4 RAM is inaccessible.<br>01 : Request to allocate LS4 RAM to Zone2.<br>10 : No request for LS4 RAM<br>11 : No request for LS4 RAM when this zone is UNLOCKED. Else LS4 RAM is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 7-6 | GRAB_RAM3 | R | 0h | Value in this field gets loaded from Z2OTP_GRABRAM[7:6] when a read is issued to address location of Z2OTP_GRABRAM in USER OTP.<br>00 : Invalid. LS3 RAM is inaccessible.<br>01 : Request to allocate LS3 RAM to Zone2.<br>10 : No request for LS3 RAM<br>11 : No request for LS3 RAM when this zone is UNLOCKED. Else LS3 RAM is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 5-4 | GRAB_RAM2 | R | 0h | Value in this field gets loaded from Z2OTP_GRABRAM[5:4] when a read is issued to address location of Z2OTP_GRABRAM in USER OTP.<br>00 : Invalid. LS2 RAM is inaccessible.<br>01 : Request to allocate LS2 RAM to Zone2.<br>10 : No request for LS2 RAM<br>11 : No request for LS2 RAM when this zone is UNLOCKED. Else LS2 RAM is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 3-2 | GRAB_RAM1 | R | 0h | Value in this field gets loaded from Z2OTP_GRABRAM[3:2] when a read is issued to address location of Z2OTP_GRABRAM in USER OTP.<br>00 : Invalid. LS1 RAM is inaccessible.<br>01 : Request to allocate LS1 RAM to Zone2.<br>10 : No request for LS1 RAM<br>11 : No request for LS1 RAM when this zone is UNLOCKED. Else LS1 RAM is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 1-0 | GRAB_RAM0 | R | 0h | Value in this field gets loaded from Z2OTP_GRABRAM[1:0] when a read is issued to address location of Z2OTP_GRABRAM in USER OTP.<br>00 : Invalid. LS0 RAM is inaccessible.<br>01 : Request to allocate LS0 RAM to Zone2.<br>10 : No request for LS0 RAM<br>11 : No request for LS0 RAM when this zone is UNLOCKED. Else LS0 RAM is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |

#### 3.14.1.7.11 B0_Z2_EXEONLYSECTR Register (Offset = 1Eh) [reset = 0h]

B0_Z2_EXEONLYSECTR is shown in Figure 3-122 and described in Table 3-132.

Return to Summary Table.

Zone 2 Flash BANK0 Execute_Only Sector Register

**Figure 3-122. B0_Z2_EXEONLYSECTR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| EXEONLY_SECT15 | EXEONLY_SECT14 | EXEONLY_SECT13 | EXEONLY_SECT12 | EXEONLY_SECT11 | EXEONLY_SECT10 | EXEONLY_SECT9 | EXEONLY_SECT8 |
| 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_SECT7 | EXEONLY_SECT6 | EXEONLY_SECT5 | EXEONLY_SECT4 | EXEONLY_SECT3 | EXEONLY_SECT2 | EXEONLY_SECT1 | EXEONLY_SECT0 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 3-132. B0_Z2_EXEONLYSECTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15 | EXEONLY_SECT15 | R | 0h | Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[15:15] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 15 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 15 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 14 | EXEONLY_SECT14 | R | 0h | Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[14:14] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 14 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 14 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 13 | EXEONLY_SECT13 | R | 0h | Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[13:13] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 13 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 13 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |

**Table 3-132. B0_Z2_EXEONLYSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 12 | EXEONLY_SECT12 | R | 0h | Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[12:12] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 12 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 12 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 11 | EXEONLY_SECT11 | R | 0h | Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[11:11] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 11 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 11 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 10 | EXEONLY_SECT10 | R | 0h | Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[10:10] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 10 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 10 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 9 | EXEONLY_SECT9 | R | 0h | Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[9:9] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 9 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 9 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 8 | EXEONLY_SECT8 | R | 0h | Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[8:8] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 8 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 8 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 7 | EXEONLY_SECT7 | R | 0h | Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[7:7] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 7 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 7 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |

**Table 3-132. B0_Z2_EXEONLYSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | EXEONLY_SECT6 | R | 0h | Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[6:6] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 6 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 6 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 5 | EXEONLY_SECT5 | R | 0h | Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[5:5] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 5 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 5 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 4 | EXEONLY_SECT4 | R | 0h | Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[4:4] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 4 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 4 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 3 | EXEONLY_SECT3 | R | 0h | Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[3:3] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 3 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 3 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 2 | EXEONLY_SECT2 | R | 0h | Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[2:2] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 2 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 2 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 1 | EXEONLY_SECT1 | R | 0h | Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[1:1] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 1 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 1 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |

**Table 3-132. B0_Z2_EXEONLYSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | EXEONLY_SECT0 | R | 0h | Value in this field gets loaded from B0_Z2OTP_EXEONLYSECT[0:0] when a read is issued to B0_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK0.<br>0 : Execute-Only protection is enabled for Flash Sector 0 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 0 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |

#### 3.14.1.7.12 Z2_EXEONLYRAMR Register (Offset = 20h) [reset = 0h]

Z2_EXEONLYRAMR is shown in Figure 3-123 and described in Table 3-133.

Return to Summary Table.

Zone 2 RAM Execute_Only Block Register

**Figure 3-123. 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 3-133. Z2_EXEONLYRAMR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R=0 | 0h | Reserved |
| 15-8 | RESERVED | R=0 | 0h | Reserved |
| 7 | EXEONLY_RAM7 | R | 0h | Value in this field gets loaded from Z2OTP_EXEONLYRAM[7:7] when a read is issued to Z2OTP_EXEONLYRAM address location in USER OTP.<br>0 : Execute-Only protection is enabled for LS7 RAM (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for LS7 RAM (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 6 | EXEONLY_RAM6 | R | 0h | Value in this field gets loaded from Z2OTP_EXEONLYRAM[6:6] when a read is issued to Z2OTP_EXEONLYRAM address location in USER OTP.<br>0 : Execute-Only protection is enabled for LS6 RAM (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for LS6 RAM (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 5 | EXEONLY_RAM5 | R | 0h | Value in this field gets loaded from Z2OTP_EXEONLYRAM[5:5] when a read is issued to Z2OTP_EXEONLYRAM address location in USER OTP.<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 |

**Table 3-133. Z2_EXEONLYRAMR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | EXEONLY_RAM4 | R | 0h | Value in this field gets loaded from Z2OTP_EXEONLYRAM[4:4] when a read is issued to Z2OTP_EXEONLYRAM address location in USER OTP.<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 |
| 3 | EXEONLY_RAM3 | R | 0h | Value in this field gets loaded from Z2OTP_EXEONLYRAM[3:3] when a read is issued to Z2OTP_EXEONLYRAM address location in USER OTP.<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 Z2OTP_EXEONLYRAM[2:2] when a read is issued to Z2OTP_EXEONLYRAM address location in USER 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 Z2OTP_EXEONLYRAM[1:1] when a read is issued to Z2OTP_EXEONLYRAM address location in USER 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 Z2OTP_EXEONLYRAM[0:0] when a read is issued to Z2OTP_EXEONLYRAM address location in USER 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 |

### 3.14.1.8 DCSM_BANK1_Z1_REGS Registers

Table 3-134 lists the memory-mapped registers for the DCSM_BANK1_Z1_REGS. All register offset addresses not listed in Table 3-134 should be considered as reserved locations and the register contents should not be modified.

**Table 3-134. DCSM_BANK1_Z1_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | B1_Z1_LINKPOINTER | Zone 1 Link Pointer for flash BANK1 | | Go |
| 6h | B1_Z1_LINKPOINTERERR | Link Pointer Error for flash BANK1 | | Go |
| 1Ah | B1_Z1_GRABSECTR | Zone 1 Grab Flash BANK1 Sectors Register | | Go |
| 1Eh | B1_Z1_EXEONLYSECTR | Zone 1 Flash BANK1 Execute_Only Sector Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-135 shows the codes that are used for access types in this section.

**Table 3-135. DCSM_BANK1_Z1_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | 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. |

#### 3.14.1.8.1 B1_Z1_LINKPOINTER Register (Offset = 0h) [reset = E0000000h]

B1_Z1_LINKPOINTER is shown in Figure 3-124 and described in Table 3-136.

Return to Summary Table.

Zone 1 Link Pointer for flash BANK1

**Figure 3-124. B1_Z1_LINKPOINTER Register**

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

**Table 3-136. B1_Z1_LINKPOINTER Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-29 | RESERVED | R | 7h | Reserved |
| 28-0 | LINKPOINTER | R | 0h | This is resolved Link-Pointer for Zone1 zone select block USER OTP of Flash BANK1. This is generated by using three physical Link-Pointer values loaded from OTP in Flash BANK1.<br>Reset type: SYSRSn |

#### 3.14.1.8.2 B1_Z1_LINKPOINTERERR Register (Offset = 6h) [reset = 0h]

B1_Z1_LINKPOINTERERR is shown in Figure 3-125 and described in Table 3-137.

Return to Summary Table.

Link Pointer Error for flash BANK1

**Figure 3-125. B1_Z1_LINKPOINTERERR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | Z1_LINKPOINTERERR | | | | | | | | | | | | |
| R=0-0h | | | R-0h | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Z1_LINKPOINTERERR | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 3-137. B1_Z1_LINKPOINTERERR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-29 | RESERVED | R=0 | 0h | Reserved |
| 28-0 | Z1_LINKPOINTERERR | R | 0h | These bits indicate errors during formation of the resolved Link-Pointer value after the three physical Link-Pointer values loaded from USER OTP in Flash BANK1<br>0 : No Error.<br>Other : Error on bit positions which is set to 1.<br>Reset type: SYSRSn |

#### 3.14.1.8.3 B1_Z1_GRABSECTR Register (Offset = 1Ah) [reset = 0h]

B1_Z1_GRABSECTR is shown in Figure 3-126 and described in Table 3-138.

Return to Summary Table.

Zone 1 Grab Flash BANK1 Sectors Register

**Figure 3-126. B1_Z1_GRABSECTR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GRAB_SECT15 | | GRAB_SECT14 | | GRAB_SECT13 | | GRAB_SECT12 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GRAB_SECT11 | | GRAB_SECT10 | | GRAB_SECT9 | | GRAB_SECT8 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GRAB_SECT7 | | GRAB_SECT6 | | GRAB_SECT5 | | GRAB_SECT4 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GRAB_SECT3 | | GRAB_SECT2 | | GRAB_SECT1 | | GRAB_SECT0 | |
| R-0h | | R-0h | | R-0h | | R-0h | |

**Table 3-138. B1_Z1_GRABSECTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GRAB_SECT15 | R | 0h | Value in this field gets loaded from B1_Z1OTP_GRABSECT[31:30] when a read is issued to address location of B1_Z1OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 15 is inaccessible.<br>01 : Request to allocate Flash Sector 15 to Zone1.<br>10 : No request for Flash Sector 15<br>11 : No request for Flash Sector 15 when this zone is UNLOCKED. Else Flash Sector 15 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 29-28 | GRAB_SECT14 | R | 0h | Value in this field gets loaded from B1_Z1OTP_GRABSECT[29:28] when a read is issued to address location of B1_Z1OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 14 is inaccessible.<br>01 : Request to allocate Flash Sector 14 to Zone1.<br>10 : No request for Flash Sector 14<br>11 : No request for Flash Sector 14 when this zone is UNLOCKED. Else Flash Sector 14 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 27-26 | GRAB_SECT13 | R | 0h | Value in this field gets loaded from B1_Z1OTP_GRABSECT[27:26] when a read is issued to address location of B1_Z1OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 13 is inaccessible.<br>01 : Request to allocate Flash Sector 13 to Zone1.<br>10 : No request for Flash Sector 13<br>11 : No request for Flash Sector 13 when this zone is UNLOCKED. Else Flash Sector 13 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |

**Table 3-138. B1_Z1_GRABSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 25-24 | GRAB_SECT12 | R | 0h | Value in this field gets loaded from B1_Z1OTP_GRABSECT[25:24] when a read is issued to address location of B1_Z1OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 12 is inaccessible.<br>01 : Request to allocate Flash Sector 12 to Zone1.<br>10 : No request for Flash Sector 12<br>11 : No request for Flash Sector 12 when this zone is UNLOCKED. Else Flash Sector 12 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 23-22 | GRAB_SECT11 | R | 0h | Value in this field gets loaded from B1_Z1OTP_GRABSECT[23:22] when a read is issued to address location of B1_Z1OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 11 is inaccessible.<br>01 : Request to allocate Flash Sector 11 to Zone1.<br>10 : No request for Flash Sector 11<br>11 : No request for Flash Sector 11 when this zone is UNLOCKED. Else Flash Sector 11 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 21-20 | GRAB_SECT10 | R | 0h | Value in this field gets loaded from B1_Z1OTP_GRABSECT[21:20] when a read is issued to address location of B1_Z1OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 10 is inaccessible.<br>01 : Request to allocate Flash Sector 10 to Zone1.<br>10 : No request for Flash Sector 10<br>11 : No request for Flash Sector 10 when this zone is UNLOCKED. Else Flash Sector 10 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 19-18 | GRAB_SECT9 | R | 0h | Value in this field gets loaded from B1_Z1OTP_GRABSECT[19:18] when a read is issued to address location of B1_Z1OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 9 is inaccessible.<br>01 : Request to allocate Flash Sector 9 to Zone1.<br>10 : No request for Flash Sector 9<br>11 : No request for Flash Sector 9 when this zone is UNLOCKED. Else Flash Sector 9 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 17-16 | GRAB_SECT8 | R | 0h | Value in this field gets loaded from B1_Z1OTP_GRABSECT[17:16] when a read is issued to address location of B1_Z1OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 8 is inaccessible.<br>01 : Request to allocate Flash Sector 8 to Zone1.<br>10 : No request for Flash Sector 8<br>11 : No request for Flash sector 8 when this zone is UNLOCKED. Else Flash sector 8 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |

**Table 3-138. B1_Z1_GRABSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | GRAB_SECT7 | R | 0h | Value in this field gets loaded from B1_Z1OTP_GRABSECT[15:14] when a read is issued to address location of B1_Z1OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 7 is inaccessible.<br>01 : Request to allocate Flash Sector 7 to Zone1.<br>10 : No request for Flash Sector 7<br>11 : No request for Flash Sector 7 when this zone is UNLOCKED. Else Flash Sector 7 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 13-12 | GRAB_SECT6 | R | 0h | Value in this field gets loaded from B1_Z1OTP_GRABSECT[13:12] when a read is issued to address location of B1_Z1OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 6 is inaccessible.<br>01 : Request to allocate Flash Sector 6 to Zone1.<br>10 : No request for Flash Sector 6<br>11 : No request for Flash Sector 6 when this zone is UNLOCKED. Else Flash Sector 6 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 11-10 | GRAB_SECT5 | R | 0h | Value in this field gets loaded from B1_Z1OTP_GRABSECT[11:10] when a read is issued to address location of B1_Z1OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 5 is inaccessible.<br>01 : Request to allocate Flash Sector 5 to Zone1.<br>10 : No request for Flash Sector 5<br>11 : No request for Flash Sector 5 when this zone is UNLOCKED. Else Flash Sector 5 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 9-8 | GRAB_SECT4 | R | 0h | Value in this field gets loaded from B1_Z1OTP_GRABSECT[9:8] when a read is issued to address location of B1_Z1OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 4 is inaccessible.<br>01 : Request to allocate Flash Sector 4 to Zone1.<br>10 : No request for Flash Sector 4<br>11 : No request for Flash Sector 4 when this zone is UNLOCKED. Else Flash Sector 4 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 7-6 | GRAB_SECT3 | R | 0h | Value in this field gets loaded from B1_Z1OTP_GRABSECT[7:6] when a read is issued to address location of B1_Z1OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 3 is inaccessible.<br>01 : Request to allocate Flash Sector 3 to Zone1.<br>10 : No request for Flash Sector 3<br>11 : No request for Flash Sector 3 when this zone is UNLOCKED. Else Flash Sector 3 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |

**Table 3-138. B1_Z1_GRABSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5-4 | GRAB_SECT2 | R | 0h | Value in this field gets loaded from B1_Z1OTP_GRABSECT[5:4] when a read is issued to address location of B1_Z1OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 2 is inaccessible.<br>01 : Request to allocate Flash Sector 2 to Zone1.<br>10 : No request for Flash Sector 2<br>11 : No request for Flash Sector 2 when this zone is UNLOCKED. Else Flash Sector 2 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 3-2 | GRAB_SECT1 | R | 0h | Value in this field gets loaded from B1_Z1OTP_GRABSECT[3:2] when a read is issued to address location of B1_Z1OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 1 is inaccessible.<br>01 : Request to allocate Flash Sector 1 to Zone1.<br>10 : No request for Flash Sector 1<br>11 : No request for Flash sector 1 when this zone is UNLOCKED. Else Flash sector 1 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 1-0 | GRAB_SECT0 | R | 0h | Value in this field gets loaded from B1_Z1OTP_GRABSECT[1:0] when a read is issued to address location of B1_Z1OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 0 is inaccessible.<br>01 : Request to allocate Flash Sector 0 to Zone1.<br>10 : No request for Flash Sector 0<br>11 : No request for Flash Sector 0 when this zone is UNLOCKED. Else Flash Sector 0 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |

#### 3.14.1.8.4 B1_Z1_EXEONLYSECTR Register (Offset = 1Eh) [reset = 0h]

B1_Z1_EXEONLYSECTR is shown in Figure 3-127 and described in Table 3-139.

Return to Summary Table.

Zone 1 Flash BANK1 Execute_Only Sector Register

**Figure 3-127. B1_Z1_EXEONLYSECTR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| EXEONLY_SECT15 | EXEONLY_SECT14 | EXEONLY_SECT13 | EXEONLY_SECT12 | EXEONLY_SECT11 | EXEONLY_SECT10 | EXEONLY_SECT9 | EXEONLY_SECT8 |
| 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_SECT7 | EXEONLY_SECT6 | EXEONLY_SECT5 | EXEONLY_SECT4 | EXEONLY_SECT3 | EXEONLY_SECT2 | EXEONLY_SECT1 | EXEONLY_SECT0 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 3-139. B1_Z1_EXEONLYSECTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15 | EXEONLY_SECT15 | R | 0h | Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[15:15] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 15 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 15 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 14 | EXEONLY_SECT14 | R | 0h | Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[14:14] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 14 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 14 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 13 | EXEONLY_SECT13 | R | 0h | Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[13:13] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 13 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 13 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |

**Table 3-139. B1_Z1_EXEONLYSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 12 | EXEONLY_SECT12 | R | 0h | Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[12:12] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 12 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 12 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 11 | EXEONLY_SECT11 | R | 0h | Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[11:11] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 11 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 11 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 10 | EXEONLY_SECT10 | R | 0h | Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[10:10] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 10 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 10 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 9 | EXEONLY_SECT9 | R | 0h | Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[9:9] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 9 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 9 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 8 | EXEONLY_SECT8 | R | 0h | Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[8:8] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 8 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 8 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 7 | EXEONLY_SECT7 | R | 0h | Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[7:7] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 7 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 7 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |

**Table 3-139. B1_Z1_EXEONLYSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | EXEONLY_SECT6 | R | 0h | Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[6:6] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 6 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 6 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 5 | EXEONLY_SECT5 | R | 0h | Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[5:5] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 5 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 5 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 4 | EXEONLY_SECT4 | R | 0h | Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[4:4] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 4 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 4 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 3 | EXEONLY_SECT3 | R | 0h | Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[3:3] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 3 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 3 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 2 | EXEONLY_SECT2 | R | 0h | Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[2:2] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 2 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 2 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |
| 1 | EXEONLY_SECT1 | R | 0h | Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[1:1] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 1 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 1 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |

**Table 3-139. B1_Z1_EXEONLYSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | EXEONLY_SECT0 | R | 0h | Value in this field gets loaded from B1_Z1OTP_EXEONLYSECT[0:0] when a read is issued to B1_Z1OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 0 (only if it's allocated to Zone1)<br>1 : Execute-Only protection is disabled for Flash Sector 0 (only if it's allocated to Zone1)<br>Reset type: SYSRSn |

### 3.14.1.9 DCSM_BANK1_Z2_REGS Registers

Table 3-140 lists the memory-mapped registers for the DCSM_BANK1_Z2_REGS. All register offset addresses not listed in Table 3-140 should be considered as reserved locations and the register contents should not be modified.

**Table 3-140. DCSM_BANK1_Z2_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | B1_Z2_LINKPOINTER | Zone 2 Link Pointer for flash BANK1 | | Go |
| 6h | B1_Z2_LINKPOINTERERR | Link Pointer Error for flash BANK1 | | Go |
| 1Ah | B1_Z2_GRABSECTR | Zone 2 Grab Flash BANK1 Sectors Register | | Go |
| 1Eh | B1_Z2_EXEONLYSECTR | Zone 2 Flash BANK1 Execute_Only Sector Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-141 shows the codes that are used for access types in this section.

**Table 3-141. DCSM_BANK1_Z2_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | 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. |

#### 3.14.1.9.1 B1_Z2_LINKPOINTER Register (Offset = 0h) [reset = E0000000h]

B1_Z2_LINKPOINTER is shown in Figure 3-128 and described in Table 3-142.

Return to Summary Table.

Zone 2 Link Pointer for flash BANK1

**Figure 3-128. B1_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 3-142. B1_Z2_LINKPOINTER Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-29 | RESERVED | R | 7h | Reserved |
| 28-0 | LINKPOINTER | R | 0h | This is resolved Link-Pointer for Zone2 zone select block USER OTP of Flash BANK1. This is generated by using three physical Link-Pointer values loaded from OTP in Flash BANK1.<br>Reset type: SYSRSn |

#### 3.14.1.9.2 B1_Z2_LINKPOINTERERR Register (Offset = 6h) [reset = 0h]

B1_Z2_LINKPOINTERERR is shown in Figure 3-129 and described in Table 3-143.

Return to Summary Table.

Link Pointer Error for flash BANK1

**Figure 3-129. B1_Z2_LINKPOINTERERR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | Z2_LINKPOINTERERR | | | | | | | | | | | | |
| R=0-0h | | | R-0h | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Z2_LINKPOINTERERR | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 3-143. B1_Z2_LINKPOINTERERR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-29 | RESERVED | R=0 | 0h | Reserved |
| 28-0 | Z2_LINKPOINTERERR | R | 0h | These bits indicate errors during formation of the resolved Link-Pointer value after the three physical Link-Pointer values loaded from USER OTP in Flash BANK1<br>0 : No Error.<br>Other : Error on bit positions which is set to 1.<br>Reset type: SYSRSn |

#### 3.14.1.9.3 B1_Z2_GRABSECTR Register (Offset = 1Ah) [reset = 0h]

B1_Z2_GRABSECTR is shown in Figure 3-130 and described in Table 3-144.

Return to Summary Table.

Zone 2 Grab Flash BANK1 Sectors Register

**Figure 3-130. B1_Z2_GRABSECTR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GRAB_SECT15 | | GRAB_SECT14 | | GRAB_SECT13 | | GRAB_SECT12 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GRAB_SECT11 | | GRAB_SECT10 | | GRAB_SECT9 | | GRAB_SECT8 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GRAB_SECT7 | | GRAB_SECT6 | | GRAB_SECT5 | | GRAB_SECT4 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GRAB_SECT3 | | GRAB_SECT2 | | GRAB_SECT1 | | GRAB_SECT0 | |
| R-0h | | R-0h | | R-0h | | R-0h | |

**Table 3-144. B1_Z2_GRABSECTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GRAB_SECT15 | R | 0h | Value in this field gets loaded from B1_Z2OTP_GRABSECT[31:30] when a read is issued to address location of B1_Z2OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 15 is inaccessible.<br>01 : Request to allocate Flash Sector 15 to Zone2.<br>10 : No request for Flash Sector 15<br>11 : No request for Flash Sector 15 when this zone is UNLOCKED. Else Flash Sector 15 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 29-28 | GRAB_SECT14 | R | 0h | Value in this field gets loaded from B1_Z2OTP_GRABSECT[29:28] when a read is issued to address location of B1_Z2OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 14 is inaccessible.<br>01 : Request to allocate Flash Sector 14 to Zone2.<br>10 : No request for Flash Sector 14<br>11 : No request for Flash Sector 14 when this zone is UNLOCKED. Else Flash Sector 14 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 27-26 | GRAB_SECT13 | R | 0h | Value in this field gets loaded from B1_Z2OTP_GRABSECT[27:26] when a read is issued to address location of B1_Z2OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 13 is inaccessible.<br>01 : Request to allocate Flash Sector 13 to Zone2.<br>10 : No request for Flash Sector 13<br>11 : No request for Flash Sector 13 when this zone is UNLOCKED. Else Flash Sector 13 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |

**Table 3-144. B1_Z2_GRABSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 25-24 | GRAB_SECT12 | R | 0h | Value in this field gets loaded from B1_Z2OTP_GRABSECT[25:24] when a read is issued to address location of B1_Z2OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 12 is inaccessible.<br>01 : Request to allocate Flash Sector 12 to Zone2.<br>10 : No request for Flash Sector 12<br>11 : No request for Flash Sector 12 when this zone is UNLOCKED. Else Flash Sector 12 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 23-22 | GRAB_SECT11 | R | 0h | Value in this field gets loaded from B1_Z2OTP_GRABSECT[23:22] when a read is issued to address location of B1_Z2OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 11 is inaccessible.<br>01 : Request to allocate Flash Sector 11 to Zone2.<br>10 : No request for Flash Sector 11<br>11 : No request for Flash Sector 11 when this zone is UNLOCKED. Else Flash Sector 11 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 21-20 | GRAB_SECT10 | R | 0h | Value in this field gets loaded from B1_Z2OTP_GRABSECT[21:20] when a read is issued to address location of B1_Z2OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 10 is inaccessible.<br>01 : Request to allocate Flash Sector 10 to Zone2.<br>10 : No request for Flash Sector 10<br>11 : No request for Flash Sector 10 when this zone is UNLOCKED. Else Flash Sector 10 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 19-18 | GRAB_SECT9 | R | 0h | Value in this field gets loaded from B1_Z2OTP_GRABSECT[19:18] when a read is issued to address location of B1_Z2OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 9 is inaccessible.<br>01 : Request to allocate Flash Sector 9 to Zone2.<br>10 : No request for Flash Sector 9<br>11 : No request for Flash Sector 9 when this zone is UNLOCKED. Else Flash Sector 9 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 17-16 | GRAB_SECT8 | R | 0h | Value in this field gets loaded from B1_Z2OTP_GRABSECT[17:16] when a read is issued to address location of B1_Z2OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 8 is inaccessible.<br>01 : Request to allocate Flash Sector 8 to Zone2.<br>10 : No request for Flash Sector 8<br>11 : No request for Flash sector 8 when this zone is UNLOCKED. Else Flash sector 8 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |

**Table 3-144. B1_Z2_GRABSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | GRAB_SECT7 | R | 0h | Value in this field gets loaded from B1_Z2OTP_GRABSECT[15:14] when a read is issued to address location of B1_Z2OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 7 is inaccessible.<br>01 : Request to allocate Flash Sector 7 to Zone2.<br>10 : No request for Flash Sector 7<br>11 : No request for Flash Sector 7 when this zone is UNLOCKED. Else Flash Sector 7 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 13-12 | GRAB_SECT6 | R | 0h | Value in this field gets loaded from B1_Z2OTP_GRABSECT[13:12] when a read is issued to address location of B1_Z2OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 6 is inaccessible.<br>01 : Request to allocate Flash Sector 6 to Zone2.<br>10 : No request for Flash Sector 6<br>11 : No request for Flash Sector 6 when this zone is UNLOCKED. Else Flash Sector 6 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 11-10 | GRAB_SECT5 | R | 0h | Value in this field gets loaded from B1_Z2OTP_GRABSECT[11:10] when a read is issued to address location of B1_Z2OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 5 is inaccessible.<br>01 : Request to allocate Flash Sector 5 to Zone2.<br>10 : No request for Flash Sector 5<br>11 : No request for Flash Sector 5 when this zone is UNLOCKED. Else Flash Sector 5 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 9-8 | GRAB_SECT4 | R | 0h | Value in this field gets loaded from B1_Z2OTP_GRABSECT[9:8] when a read is issued to address location of B1_Z2OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 4 is inaccessible.<br>01 : Request to allocate Flash Sector 4 to Zone2.<br>10 : No request for Flash Sector 4<br>11 : No request for Flash Sector 4 when this zone is UNLOCKED. Else Flash Sector 4 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 7-6 | GRAB_SECT3 | R | 0h | Value in this field gets loaded from B1_Z2OTP_GRABSECT[7:6] when a read is issued to address location of B1_Z2OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 3 is inaccessible.<br>01 : Request to allocate Flash Sector 3 to Zone2.<br>10 : No request for Flash Sector 3<br>11 : No request for Flash Sector 3 when this zone is UNLOCKED. Else Flash Sector 3 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |

**Table 3-144. B1_Z2_GRABSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5-4 | GRAB_SECT2 | R | 0h | Value in this field gets loaded from B1_Z2OTP_GRABSECT[5:4] when a read is issued to address location of B1_Z2OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 2 is inaccessible.<br>01 : Request to allocate Flash Sector 2 to Zone2.<br>10 : No request for Flash Sector 2<br>11 : No request for Flash Sector 2 when this zone is UNLOCKED. Else Flash Sector 2 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 3-2 | GRAB_SECT1 | R | 0h | Value in this field gets loaded from B1_Z2OTP_GRABSECT[1:0] when a read is issued to address location of B1_Z2OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 1 is inaccessible.<br>01 : Request to allocate Flash Sector 1 to Zone2.<br>10 : No request for Flash Sector 1<br>11 : No request for Flash sector 1 when this zone is UNLOCKED. Else Flash sector 1 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |
| 1-0 | GRAB_SECT0 | R | 0h | Value in this field gets loaded from B1_Z2OTP_GRABSECT[1:0] when a read is issued to address location of B1_Z2OTP_GRABSECT in USER OTP of Flash BANK1.<br>00 : Invalid. Flash Sector 0 is inaccessible.<br>01 : Request to allocate Flash Sector 0 to Zone2.<br>10 : No request for Flash Sector 0<br>11 : No request for Flash Sector 0 when this zone is UNLOCKED. Else Flash Sector 0 is inaccessible if this zone is LOCKED.<br>Reset type: SYSRSn |

#### 3.14.1.9.4 B1_Z2_EXEONLYSECTR Register (Offset = 1Eh) [reset = 0h]

B1_Z2_EXEONLYSECTR is shown in Figure 3-131 and described in Table 3-145.

Return to Summary Table.

Zone 2 Flash BANK1 Execute_Only Sector Register

**Figure 3-131. B1_Z2_EXEONLYSECTR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| EXEONLY_SECT15 | EXEONLY_SECT14 | EXEONLY_SECT13 | EXEONLY_SECT12 | EXEONLY_SECT11 | EXEONLY_SECT10 | EXEONLY_SECT9 | EXEONLY_SECT8 |
| 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_SECT7 | EXEONLY_SECT6 | EXEONLY_SECT5 | EXEONLY_SECT4 | EXEONLY_SECT3 | EXEONLY_SECT2 | EXEONLY_SECT1 | EXEONLY_SECT0 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 3-145. B1_Z2_EXEONLYSECTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15 | EXEONLY_SECT15 | R | 0h | Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[15:15] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 15 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 15 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 14 | EXEONLY_SECT14 | R | 0h | Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[14:14] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 14 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 14 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 13 | EXEONLY_SECT13 | R | 0h | Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[13:13] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 13 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 13 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |

**Table 3-145. B1_Z2_EXEONLYSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 12 | EXEONLY_SECT12 | R | 0h | Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[12:12] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 12 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 12 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 11 | EXEONLY_SECT11 | R | 0h | Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[11:11] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 11 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 11 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 10 | EXEONLY_SECT10 | R | 0h | Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[10:10] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 10 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 10 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 9 | EXEONLY_SECT9 | R | 0h | Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[9:9] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 9 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 9 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 8 | EXEONLY_SECT8 | R | 0h | Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[8:8] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 8 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 8 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 7 | EXEONLY_SECT7 | R | 0h | Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[7:7] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 7 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 7 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |

**Table 3-145. B1_Z2_EXEONLYSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | EXEONLY_SECT6 | R | 0h | Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[6:6] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 6 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 6 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 5 | EXEONLY_SECT5 | R | 0h | Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[5:5] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 5 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 5 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 4 | EXEONLY_SECT4 | R | 0h | Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[4:4] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 4 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 4 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 3 | EXEONLY_SECT3 | R | 0h | Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[3:3] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 3 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 3 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 2 | EXEONLY_SECT2 | R | 0h | Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[2:2] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 2 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 2 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |
| 1 | EXEONLY_SECT1 | R | 0h | Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[1:1] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 1 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 1 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |

**Table 3-145. B1_Z2_EXEONLYSECTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | EXEONLY_SECT0 | R | 0h | Value in this field gets loaded from B1_Z2OTP_EXEONLYSECT[0:0] when a read is issued to B1_Z2OTP_EXEONLYSECT address location in USER OTP of Flash BANK1.<br>0 : Execute-Only protection is enabled for Flash Sector 0 (only if it's allocated to Zone2)<br>1 : Execute-Only protection is disabled for Flash Sector 0 (only if it's allocated to Zone2)<br>Reset type: SYSRSn |

### 3.14.1.10 DCSM_COMMON_REGS Registers

Table 3-146 lists the memory-mapped registers for the DCSM_COMMON_REGS. All register offset addresses not listed in Table 3-146 should be considered as reserved locations and the register contents should not be modified.

**Table 3-146. DCSM_COMMON_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | FLSEM | Flash Wrapper Semaphore Register | EALLOW | Go |
| 2h | B0_SECTSTAT | Flash BANK0 Sectors Status Register | | Go |
| 4h | RAMSTAT | RAM Status Register | | Go |
| 8h | B1_SECTSTAT | Flash BANK1 Sectors Status Register | | Go |
| Ah | SECERRSTAT | Security Error Status Register | | Go |
| Ch | SECERRCLR | Security Error Clear Register | | Go |
| Eh | SECERRFRC | Security Error Force Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-147 shows the codes that are used for access types in this section.

**Table 3-147. DCSM_COMMON_REGS Access Type Codes**

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

#### 3.14.1.10.1 FLSEM Register (Offset = 0h) [reset = 0h]

FLSEM is shown in Figure 3-132 and described in Table 3-148.

Return to Summary Table.

Flash Wrapper Semaphore Register

**Figure 3-132. 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 3-148. 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 |

#### 3.14.1.10.2 B0_SECTSTAT Register (Offset = 2h) [reset = 0h]

B0_SECTSTAT is shown in Figure 3-133 and described in Table 3-149.

Return to Summary Table.

Flash BANK0 Sectors Status Register

**Figure 3-133. B0_SECTSTAT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| STATUS_SECT15 | | STATUS_SECT14 | | STATUS_SECT13 | | STATUS_SECT12 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| STATUS_SECT11 | | STATUS_SECT10 | | STATUS_SECT9 | | STATUS_SECT8 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| STATUS_SECT7 | | STATUS_SECT6 | | STATUS_SECT5 | | STATUS_SECT4 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| STATUS_SECT3 | | STATUS_SECT2 | | STATUS_SECT1 | | STATUS_SECT0 | |
| R-0h | | R-0h | | R-0h | | R-0h | |

**Table 3-149. B0_SECTSTAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | STATUS_SECT15 | R | 0h | Reflects the status of Flash BANK0 Sector 15.<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 |
| 29-28 | STATUS_SECT14 | R | 0h | Reflects the status of Flash BANK0 Sector 14.<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 |
| 27-26 | STATUS_SECT13 | R | 0h | Reflects the status of Flash BANK0 Sector 13.<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_SECT12 | R | 0h | Reflects the status of Flash BANK0 Sector 12.<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 |

**Table 3-149. B0_SECTSTAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 23-22 | STATUS_SECT11 | R | 0h | Reflects the status of Flash BANK0 Sector 11.<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_SECT10 | R | 0h | Reflects the status of Flash BANK0 Sector 10.<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 |
| 19-18 | STATUS_SECT9 | R | 0h | Reflects the status of Flash BANK0 Sector 9.<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_SECT8 | R | 0h | Reflects the status of Flash BANK0 sector 8.<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_SECT7 | R | 0h | Reflects the status of Flash BANK0 Sector 7.<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_SECT6 | R | 0h | Reflects the status of Flash BANK0 Sector 6.<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_SECT5 | R | 0h | Reflects the status of Flash BANK0 Sector 5.<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 |

**Table 3-149. B0_SECTSTAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9-8 | STATUS_SECT4 | R | 0h | Reflects the status of Flash BANK0 Sector 4.<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_SECT3 | R | 0h | Reflects the status of Flash BANK0 Sector 3.<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_SECT2 | R | 0h | Reflects the status of Flash BANK0 Sector 2.<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 |
| 3-2 | STATUS_SECT1 | R | 0h | Reflects the status of Flash BANK0 sector 1.<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_SECT0 | R | 0h | Reflects the status of Flash BANK0 Sector 0.<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 |

### 3.14.1.10.3 RAMSTAT Register (Offset = 4h) [reset = 0h]

RAMSTAT is shown in Figure 3-134 and described in Table 3-150.

Return to Summary Table.

RAM Status Register

**Figure 3-134. RAMSTAT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| STATUS_RAM7 | | STATUS_RAM6 | | STATUS_RAM5 | | STATUS_RAM4 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| STATUS_RAM3 | | STATUS_RAM2 | | STATUS_RAM1 | | STATUS_RAM0 | |
| R-0h | | R-0h | | R-0h | | R-0h | |

**Table 3-150. RAMSTAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-14 | STATUS_RAM7 | R | 0h | Reflects the status of LS7 RAM.<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 LS6 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 |
| 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 |

**Table 3-150. RAMSTAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |

#### 3.14.1.10.4 B1_SECTSTAT Register (Offset = 8h) [reset = 0h]

B1_SECTSTAT is shown in Figure 3-135 and described in Table 3-151.

Return to Summary Table.

Flash BANK1 Sectors Status Register

**Figure 3-135. B1_SECTSTAT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| STATUS_SECT15 | | STATUS_SECT14 | | STATUS_SECT13 | | STATUS_SECT12 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| STATUS_SECT11 | | STATUS_SECT10 | | STATUS_SECT9 | | STATUS_SECT8 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| STATUS_SECT7 | | STATUS_SECT6 | | STATUS_SECT5 | | STATUS_SECT4 | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| STATUS_SECT3 | | STATUS_SECT2 | | STATUS_SECT1 | | STATUS_SECT0 | |
| R-0h | | R-0h | | R-0h | | R-0h | |

**Table 3-151. B1_SECTSTAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | STATUS_SECT15 | R | 0h | Reflects the status of Flash BANK1 Sector 15.<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 |
| 29-28 | STATUS_SECT14 | R | 0h | Reflects the status of Flash BANK1 Sector 14.<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 |
| 27-26 | STATUS_SECT13 | R | 0h | Reflects the status of Flash BANK1 Sector 13.<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_SECT12 | R | 0h | Reflects the status of Flash BANK1 Sector 12.<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 |

**Table 3-151. B1_SECTSTAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 23-22 | STATUS_SECT11 | R | 0h | Reflects the status of Flash BANK1 Sector 11.<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_SECT10 | R | 0h | Reflects the status of Flash BANK1 Sector 10.<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 |
| 19-18 | STATUS_SECT9 | R | 0h | Reflects the status of Flash BANK1 Sector 9.<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_SECT8 | R | 0h | Reflects the status of Flash BANK1 sector 8.<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_SECT7 | R | 0h | Reflects the status of Flash BANK1 Sector 7.<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_SECT6 | R | 0h | Reflects the status of Flash BANK1 Sector 6.<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_SECT5 | R | 0h | Reflects the status of Flash BANK1 Sector 5.<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 |

**Table 3-151. B1_SECTSTAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9-8 | STATUS_SECT4 | R | 0h | Reflects the status of Flash BANK1 Sector 4.<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_SECT3 | R | 0h | Reflects the status of Flash BANK1 Sector 3.<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_SECT2 | R | 0h | Reflects the status of Flash BANK1 Sector 2.<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 |
| 3-2 | STATUS_SECT1 | R | 0h | Reflects the status of Flash BANK1 sector 1.<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_SECT0 | R | 0h | Reflects the status of Flash BANK1 Sector 0.<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 |

#### 3.14.1.10.5 SECERRSTAT Register (Offset = Ah) [reset = 0h]

SECERRSTAT is shown in Figure 3-136 and described in Table 3-152.

Return to Summary Table.

Security Error Status Register

**Figure 3-136. SECERRSTAT Register**

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

**Table 3-152. SECERRSTAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-1 | RESERVED | R=0 | 0h | Reserved |
| 0 | ERR | R | 0h | This bit indicates if any error has occurred in the load of any security configuration from USER-OTP.<br>0: No error has occurred in the load of security information from USER-OTP<br>1: Error has occurred in the load of security information from USER-OTP<br>Reset type: PORn |

### 3.14.1.10.6 SECERRCLR Register (Offset = Ch) [reset = 0h]

SECERRCLR is shown in Figure 3-137 and described in Table 3-153.

Return to Summary Table.

Security Error Clear Register

**Figure 3-137. SECERRCLR Register**

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

**Table 3-153. SECERRCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-1 | RESERVED | R=0 | 0h | Reserved |
| 0 | ERR | R=0/W=1 | 0h | A write of '1' clears the SECERRSTST.ERR bit. Write of '0' is ignored. This bit always reads back '0'.<br>Reset type: N/A |

#### 3.14.1.10.7 SECERRFRC Register (Offset = Eh) [reset = 0h]

SECERRFRC is shown in Figure 3-138 and described in Table 3-154.

Return to Summary Table.

Security Error Force Register

**Figure 3-138. SECERRFRC Register**

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

**Table 3-154. SECERRFRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-1 | RESERVED | R=0 | 0h | Reserved |
| 0 | ERR | R=0/W=1 | 0h | A write of '1' sets the SECERRSTST.ERR bit. Write of '0' is ignored. This bit always reads back '0'.<br>Reset type: N/A |

### 3.14.1.11 DCSM_COMMON2_REGS Registers

Table 3-155 lists the memory-mapped registers for the DCSM_COMMON2_REGS. All register offset addresses not listed in Table 3-155 should be considered as reserved locations and the register contents should not be modified.

**Table 3-155. DCSM_COMMON2_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|

Complex bit access types are encoded to fit into small table cells. Table 3-156 shows the codes that are used for access types in this section.

**Table 3-156. DCSM_COMMON2_REGS Access Type Codes**

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

### 3.14.1.12 DEV_CFG_REGS Registers

Table 3-157 lists the memory-mapped registers for the DEV_CFG_REGS. All register offset addresses not listed in Table 3-157 should be considered as reserved locations and the register contents should not be modified.

**Table 3-157. DEV_CFG_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 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 |
| 74h | FUSEERR | e-Fuse error Status register | | Go |
| 82h | SOFTPRES0 | Processing Block 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 |
| 96h | SOFTPRES10 | Peripheral Software Reset register | EALLOW | Go |
| 9Ch | SOFTPRES13 | Peripheral Software Reset register | EALLOW | Go |
| 9Eh | SOFTPRES14 | Peripheral Software Reset register | EALLOW | Go |
| A0h | SOFTPRES15 | Peripheral Software Reset register | EALLOW | Go |
| A2h | SOFTPRES16 | Peripheral Software Reset register | EALLOW | Go |
| A4h | SOFTPRES17 | Peripheral Software Reset register | EALLOW | Go |
| A6h | SOFTPRES18 | Peripheral Software Reset register | EALLOW | Go |
| A8h | SOFTPRES19 | Peripheral Software Reset register | EALLOW | Go |
| AAh | SOFTPRES20 | Peripheral Software Reset register | EALLOW | Go |
| ACh | SOFTPRES21 | Peripheral Software Reset register | EALLOW | Go |
| 130h | TAP_STATUS | Status of JTAG State machine & Debugger Connect | | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-158 shows the codes that are used for access types in this section.

**Table 3-158. DEV_CFG_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| W=1 | W | Write |
| WOnce | W | Write |
| WSOnce | SOnce<br>W | Set once<br>Write |
| **Reset or Default Value** | | |
| *-n* | | Value after reset or the default value |
| **Register Array Variables** | | |

**Table 3-158. DEV_CFG_REGS Access Type Codes (continued)**

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

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

PARTIDL is shown in Figure 3-139 and described in Table 3-159.

Return to Summary Table.

Lower 32-bit of Device PART Identification Number

**Figure 3-139. PARTIDL Register**

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

**Table 3-159. PARTIDL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | RESERVED | R | X | Reserved |
| 27-24 | RESERVED | R | 0h | Reserved |
| 23-16 | FLASH_SIZE | R | X | 6 = 256KB<br>5 = 128KB<br>Reset type: XRSn |
| 15 | RESERVED | R | 0h | Reserved |
| 14-13 | INSTASPIN | R | X | 0 = InstaSPIN-MOTION + InstaSPIN-FOC<br>1 = InstaSPIN-FOC<br>2 = NONE<br>3 = NONE<br>Reset type: XRSn |
| 12 | RESERVED | R | 0h | Reserved |
| 11 | RESERVED | R | 0h | Reserved |
| 10-8 | PIN_COUNT | R | X | 0 = 56 pin<br>1 = 64 pin (Q100)<br>2 = 64 pin<br>3 = 80 pin<br>4 = 144 pin<br>5 = Reserved<br>6 = Reserved<br>7 = Reserved<br>Reset type: XRSn |
| 7-6 | QUAL | R | X | 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 |

**Table 3-159. PARTIDL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2-0 | RESERVED | R | 0h | Reserved |

### 3.14.1.12.2 PARTIDH Register (Offset = Ah) [reset = X]

PARTIDH is shown in Figure 3-140 and described in Table 3-160.

Return to Summary Table.

Upper 32-bit of Device PART Identification Number

**Figure 3-140. PARTIDH Register**

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

**Table 3-160. PARTIDH Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | DEVICE_CLASS_ID | R | 1h | Device class ID<br>Reset type: XRSn |
| 23-16 | PARTNO | R | X | Refer to Datasheet for Device Part Number<br>Reset type: XRSn |
| 15-8 | FAMILY | R | X | Device Family<br>0x3 - DELFINO DUAL CORE<br>0x4 - DELFINO SINGLE CORE<br>0x5 - PICCOLO SINGLE CORE<br>Other values Reserved<br>Reset type: XRSn |
| 7-4 | RESERVED | R | 0h | Reserved |
| 3-0 | RESERVED | R | 0h | Reserved |

#### 3.14.1.12.3 REVID Register (Offset = Ch) [reset = 0h]

REVID is shown in Figure 3-141 and described in Table 3-161.

Return to Summary Table.

Device Revision Number

**Figure 3-141. REVID Register**

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

**Table 3-161. REVID Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R=0 | 0h | Reserved |
| 15-0 | REVID | R | 0h | Device Revision<br>Reset type: N/A |

#### 3.14.1.12.4 FUSEERR Register (Offset = 74h) [reset = 0h]

FUSEERR is shown in Figure 3-142 and described in Table 3-162.

Return to Summary Table.

e-Fuse error Status register

**Figure 3-142. 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 3-162. FUSEERR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5 | ERR | R | 0h | Efuse Self Test Error Status set by hardware after fuse self test completes, in case of self test error<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>Note:<br>[1] 10101 means a single-bit error during autoload. Since this gets corrected by the ECC mechanism, this value shouldn't be treated as an error condition.<br>Reset type: XRSn |

### 3.14.1.12.5 SOFTPRES0 Register (Offset = 82h) [reset = 0h]

SOFTPRES0 is shown in Figure 3-143 and described in Table 3-163.

Return to 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 3-143. SOFTPRES0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | RESERVED | RESERVED | CPU1_CLA1 |
| R=0-0h | | | | R-0h | R-0h | R-0h | R/W-0h |

**Table 3-163. SOFTPRES0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-4 | RESERVED | R=0 | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1 | RESERVED | R | 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: SYSRSn |

### 3.14.1.12.6 SOFTPRES2 Register (Offset = 86h) [reset = 0h]

SOFTPRES2 is shown in Figure 3-144 and described in Table 3-164.

Return to Summary Table.

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-144. SOFTPRES2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-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 3-164. SOFTPRES2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R=0 | 0h | Reserved |
| 15 | RESERVED | R | 0h | Reserved |
| 14 | RESERVED | R | 0h | Reserved |
| 13 | RESERVED | R | 0h | Reserved |
| 12 | RESERVED | R | 0h | Reserved |
| 11 | RESERVED | R | 0h | Reserved |
| 10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 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: 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: 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: SYSRSn |
| 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: 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: SYSRSn |

**Table 3-164. SOFTPRES2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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: 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: 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: SYSRSn |

### 3.14.1.12.7 SOFTPRES3 Register (Offset = 88h) [reset = 0h]

SOFTPRES3 is shown in Figure 3-145 and described in Table 3-165.

Return to Summary Table.

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-145. SOFTPRES3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | ECAP7 | ECAP6 | ECAP5 | ECAP4 | ECAP3 | ECAP2 | ECAP1 |
| R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-165. SOFTPRES3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-8 | RESERVED | R=0 | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | ECAP7 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: SYSRSn |
| 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: 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: 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: 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: 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: 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: SYSRSn |

### 3.14.1.12.8 SOFTPRES4 Register (Offset = 8Ah) [reset = 0h]

SOFTPRES4 is shown in Figure 3-146 and described in Table 3-166.

Return to Summary Table.

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-146. SOFTPRES4 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | RESERVED | EQEP2 | EQEP1 |
| R=0-0h | | | | R-0h | R-0h | R/W-0h | R/W-0h |

**Table 3-166. SOFTPRES4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-4 | RESERVED | R=0 | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 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: 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: SYSRSn |

### 3.14.1.12.9 SOFTPRES6 Register (Offset = 8Eh) [reset = 0h]

SOFTPRES6 is shown in Figure 3-147 and described in Table 3-167.

Return to Summary Table.

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-147. 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 | RESERVED | SD1 |
| R=0-0h | | | | | | | | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R/W-0h |

**Table 3-167. SOFTPRES6 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-8 | RESERVED | R=0 | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1 | RESERVED | R | 0h | Reserved |
| 0 | SD1 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: SYSRSn |

### 3.14.1.12.10 SOFTPRES7 Register (Offset = 90h) [reset = 0h]

SOFTPRES7 is shown in Figure 3-148 and described in Table 3-168.

Return to Summary Table.

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-148. SOFTPRES7 Register**

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

**Table 3-168. SOFTPRES7 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-4 | RESERVED | R=0 | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 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: 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: SYSRSn |

### 3.14.1.12.11 SOFTPRES8 Register (Offset = 92h) [reset = 0h]

SOFTPRES8 is shown in Figure 3-149 and described in Table 3-169.

Return to Summary Table.

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-149. SOFTPRES8 Register**

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

**Table 3-169. SOFTPRES8 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-18 | RESERVED | R=0 | 0h | Reserved |
| 17 | RESERVED | R | 0h | Reserved |
| 16 | RESERVED | R | 0h | Reserved |
| 15-4 | RESERVED | R=0 | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 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: 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: SYSRSn |

### 3.14.1.12.12 SOFTPRES9 Register (Offset = 94h) [reset = 0h]

SOFTPRES9 is shown in Figure 3-150 and described in Table 3-170.

Return to Summary Table.

Peripheral Software Reset register

When bits in this register are set, the respective peripheral is in reset. All data is lost and the peripheral registers are returned to their reset states. Bits must be manually cleared after being set.

**Figure 3-150. SOFTPRES9 Register**

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

**Table 3-170. SOFTPRES9 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-2 | RESERVED | R=0 | 0h | Reserved |
| 1 | RESERVED | R | 0h | Reserved |
| 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: SYSRSn |

### 3.14.1.12.13 SOFTPRES10 Register (Offset = 96h) [reset = 0h]

SOFTPRES10 is shown in Figure 3-151 and described in Table 3-171.

Return to 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 3-151. SOFTPRES10 Register**

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

**Table 3-171. SOFTPRES10 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-4 | RESERVED | R=0 | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1 | CAN_B | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: SYSRSn |
| 0 | CAN_A | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: SYSRSn |

### 3.14.1.12.14 SOFTPRES13 Register (Offset = 9Ch) [reset = 0h]

SOFTPRES13 is shown in Figure 3-152 and described in Table 3-172.

Return to 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 3-152. SOFTPRES13 Register**

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

**Table 3-172. SOFTPRES13 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-4 | RESERVED | R=0 | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 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: 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: 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: SYSRSn |

### 3.14.1.12.15 SOFTPRES14 Register (Offset = 9Eh) [reset = 0h]

SOFTPRES14 is shown in Figure 3-153 and described in Table 3-173.

Return to 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 3-153. SOFTPRES14 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | CMPSS7 | CMPSS6 | CMPSS5 | CMPSS4 | CMPSS3 | CMPSS2 | CMPSS1 |
| R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-173. SOFTPRES14 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-8 | RESERVED | R=0 | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 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: 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: 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: 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: 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: 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: 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: SYSRSn |

### 3.14.1.12.16 SOFTPRES15 Register (Offset = A0h) [reset = 0h]

SOFTPRES15 is shown in Figure 3-154 and described in Table 3-174.

Return to 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 3-154. SOFTPRES15 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | PGA7 | PGA6 | PGA5 | PGA4 | PGA3 | PGA2 | PGA1 |
| R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-174. SOFTPRES15 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-8 | RESERVED | R=0 | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | PGA7 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: SYSRSn |
| 5 | PGA6 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: SYSRSn |
| 4 | PGA5 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: SYSRSn |
| 3 | PGA4 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: SYSRSn |
| 2 | PGA3 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: SYSRSn |
| 1 | PGA2 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: SYSRSn |
| 0 | PGA1 | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: SYSRSn |

### 3.14.1.12.17 SOFTPRES16 Register (Offset = A2h) [reset = 0h]

SOFTPRES16 is shown in Figure 3-155 and described in Table 3-175.

Return to 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 3-155. SOFTPRES16 Register**

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

**Table 3-175. SOFTPRES16 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | RESERVED | R=0 | 0h | Reserved |
| 19 | RESERVED | R | 0h | Reserved |
| 18 | RESERVED | R | 0h | Reserved |
| 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: 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: SYSRSn |
| 15-4 | RESERVED | R=0 | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1 | RESERVED | R | 0h | Reserved |
| 0 | RESERVED | R | 0h | Reserved |

### 3.14.1.12.18 SOFTPRES17 Register (Offset = A4h) [reset = X]

SOFTPRES17 is shown in Figure 3-156 and described in Table 3-176.

Return to 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 3-156. SOFTPRES17 Register**

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

**Table 3-176. SOFTPRES17 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-4 | RESERVED | R | X | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1 | RESERVED | R | 0h | Reserved |
| 0 | RESERVED | R | 0h | Reserved |

### 3.14.1.12.19 SOFTPRES18 Register (Offset = A6h) [reset = X]

SOFTPRES18 is shown in Figure 3-157 and described in Table 3-177.

Return to 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 3-157. SOFTPRES18 Register**

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

**Table 3-177. SOFTPRES18 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-4 | RESERVED | R | X | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1 | RESERVED | R | 0h | Reserved |
| 0 | RESERVED | R | 0h | Reserved |

### 3.14.1.12.20 SOFTPRES19 Register (Offset = A8h) [reset = 0h]

SOFTPRES19 is shown in Figure 3-158 and described in Table 3-178.

Return to 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 3-158. SOFTPRES19 Register**

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

**Table 3-178. SOFTPRES19 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-4 | RESERVED | R=0 | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1 | RESERVED | R | 0h | Reserved |
| 0 | LIN_A | R/W | 0h | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: SYSRSn |

### 3.14.1.12.21 SOFTPRES20 Register (Offset = AAh) [reset = X]

SOFTPRES20 is shown in Figure 3-159 and described in Table 3-179.

Return to 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 3-159. SOFTPRES20 Register**

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

**Table 3-179. SOFTPRES20 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-2 | RESERVED | R | X | Reserved |
| 1 | RESERVED | R | 0h | Reserved |
| 0 | PMBUS_A | R | X | 1: Module is under reset<br>0: Module reset is determined by the normal device reset structure<br>Reset type: SYSRSn |

### 3.14.1.12.22 SOFTPRES21 Register (Offset = ACh) [reset = 0h]

SOFTPRES21 is shown in Figure 3-160 and described in Table 3-180.

Return to 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 3-160. SOFTPRES21 Register**

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

**Table 3-180. SOFTPRES21 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-2 | RESERVED | R=0 | 0h | Reserved |
| 1 | RESERVED | R | 0h | Reserved |
| 0 | RESERVED | R | 0h | Reserved |

#### 3.14.1.12.23 TAP_STATUS Register (Offset = 130h) [reset = 0h]

TAP_STATUS is shown in Figure 3-161 and described in Table 3-181.

Return to Summary Table.

Status of JTAG State machine & Debugger Connect

**Figure 3-161. TAP_STATUS Register**

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

**Table 3-181. TAP_STATUS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | DCON | R | 0h | DebugConnect indication from IcePick.<br>Reset type: PORn |
| 30-16 | RESERVED | R=0 | 0h | Reserved |
| 15-0 | TAP_STATE | R | 0h | TAP State Vector. With bits representing, Connect coresponding POTAP* output to the<br>0:TLR,<br>1:IDLE,<br>2:SELECTDR,<br>3:CAPDR,<br>4:SHIFTDR,<br>5:EXIT1DR,<br>6:PAUSEDR,<br>7:EXIT2DR,<br>8:UPDTDR,<br>9:SLECTIR,<br>10:CAPIR,<br>11:SHIFTIR,<br>12:EXIT1IR,<br>13:PAUSEIR,<br>14:EXIT2IR,<br>15:UPDTIR,<br>Reset type: PORn |

### 3.14.1.13 DMA_CLA_SRC_SEL_REGS Registers

Table 3-182 lists the memory-mapped registers for the DMA_CLA_SRC_SEL_REGS. All register offset addresses not listed in Table 3-182 should be considered as reserved locations and the register contents should not be modified.

**Table 3-182. 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 3-183 shows the codes that are used for access types in this section.

**Table 3-183. DMA_CLA_SRC_SEL_REGS Access Type Codes**

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

#### 3.14.1.13.1 CLA1TASKSRCSELLOCK Register (Offset = 0h) [reset = 0h]

CLA1TASKSRCSELLOCK is shown in Figure 3-162 and described in Table 3-184.

Return to Summary Table.

CLA1 Task Trigger Source Select Lock Register

**Figure 3-162. 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 3-184. CLA1TASKSRCSELLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-2 | 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 |

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

DMACHSRCSELLOCK is shown in Figure 3-163 and described in Table 3-185.

Return to Summary Table.

DMA Channel Triger Source Select Lock Register

**Figure 3-163. 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 3-185. DMACHSRCSELLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-2 | RESERVED | R=0 | 0h | Reserved |
| 1 | DMACHSRCSEL2 | R/WSOnce | 0h | DMACHSRCSEL2 Register Lock bit:<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 |

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

CLA1TASKSRCSEL1 is shown in Figure 3-164 and described in Table 3-186.

Return to Summary Table.

CLA1 Task Trigger Source Select Register-1

**Figure 3-164. 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 3-186. 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 |

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

CLA1TASKSRCSEL2 is shown in Figure 3-165 and described in Table 3-187.

Return to Summary Table.

CLA1 Task Trigger Source Select Register-2

**Figure 3-165. 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 3-187. 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 |

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

DMACHSRCSEL1 is shown in Figure 3-166 and described in Table 3-188.

Return to Summary Table.

DMA Channel Trigger Source Select Register-1

**Figure 3-166. 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 3-188. 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 |

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

DMACHSRCSEL2 is shown in Figure 3-167 and described in Table 3-189.

Return to Summary Table.

DMA Channel Trigger Source Select Register-2

**Figure 3-167. 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 3-189. 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 |

### 3.14.1.14 FLASH_CTRL_REGS Registers

Table 3-190 lists the memory-mapped registers for the FLASH_CTRL_REGS. All register offset addresses not listed in Table 3-190 should be considered as reserved locations and the register contents should not be modified.

**Table 3-190. 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 |
| 26h | FPAC2 | Flash Pump Access Control Register 2 | 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 3-191 shows the codes that are used for access types in this section.

**Table 3-191. 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. |

#### 3.14.1.14.1 FRDCNTL Register (Offset = 0h) [reset = F00h]

FRDCNTL is shown in Figure 3-168 and described in Table 3-192.

Return to Summary Table.

Flash Read Control Register

**Figure 3-168. 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 3-192. 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 |

#### 3.14.1.14.2 FBAC Register (Offset = 1Eh) [reset = Fh]

FBAC is shown in Figure 3-169 and described in Table 3-193.

Return to Summary Table.

Flash Bank Access Control Register

**Figure 3-169. FBAC Register**

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

**Table 3-193. FBAC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-8 | BAGP | R/W | 0h | Bank Active Grace Period. These bits contain the starting count value for the BAGP down counter. Any access to a given bank causes its BAGP counter to reload the BAGP value for that bank. After the last access to this flash bank, the down counter delays from 0 to 255 prescaled SYSCLK clock cycles before putting the bank into one of the fallback power modes as determined by the FBFALLBACK register. This value must be greater than 1 when the fallback mode is not ACTIVE.<br>Note: The prescaled clock used for the BAGP down counter is a clock divided by 16 from input SYSCLK.<br>Reset type: SYSRSn |
| 7-0 | RESERVED | R | 0h | Reserved |

### 3.14.1.14.3 FBFALLBACK Register (Offset = 20h) [reset = 0h]

FBFALLBACK is shown in Figure 3-170 and described in Table 3-194.

Return to Summary Table.

Flash Bank Fallback Power Register

**Figure 3-170. FBFALLBACK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | BNKPWR1 | | BNKPWR0 | |
| R-0h | | | | R/W-0h | | R/W-0h | |

**Table 3-194. FBFALLBACK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-4 | RESERVED | R | 0h | Reserved |
| 3-2 | BNKPWR1 | R/W | 0h | Fall Back power mode<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>Reset type: SYSRSn |
| 1-0 | BNKPWR0 | R/W | 0h | Fall Back power mode<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>Reset type: SYSRSn |

### 3.14.1.14.4 FBPRDY Register (Offset = 22h) [reset = 0h]

FBPRDY is shown in Figure 3-171 and described in Table 3-195.

Return to Summary Table.

Flash Bank Pump Ready Register

**Figure 3-171. FBPRDY Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PUMPRDY | RESERVED | | | | | | |
| R-0h | R-0h | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | BANK1RDY | BANK0RDY |
| R-0h | | | | | | R-0h | R-0h |

**Table 3-195. 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 |
| 14-2 | RESERVED | R | 0h | Reserved |
| 1 | BANK1RDY | R | 0h | Bank 1 Ready. This is a read-only register which allows software to determine if the Bank 1 is ready for Flash access before the access is attempted.<br>Note: The user should wait for both the pump and the bank to be ready before attempting an access.<br>0 Bank 1 is not ready.<br>1 Bank 1 is in active power mode and is ready for access.<br>Reset type: SYSRSn |
| 0 | BANK0RDY | R | 0h | Bank 0 Ready. This is a read-only register which allows software to determine if the Bank 0 is ready for Flash access before the access is attempted.<br>Note: The user should wait for both the pump and the bank to be ready before attempting an access.<br>0 Bank 0 is not ready.<br>1 Bank 0 is in active power mode and is ready for access.<br>Reset type: SYSRSn |

### 3.14.1.14.5 FPAC1 Register (Offset = 24h) [reset = 00A00000h]

FPAC1 is shown in Figure 3-172 and described in Table 3-196.

Return to Summary Table.

Flash Pump Access Control Register 1

**Figure 3-172. FPAC1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | PSLEEP | | | |
| R-0h | | | | R/W-A0h | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PSLEEP | | | | | | | |
| R/W-A0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | PMPPWR |
| R-0h | | | | | | | R/W-0h |

**Table 3-196. FPAC1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | RESERVED | R | 0h | Reserved |
| 27-16 | PSLEEP | R/W | A0h | Pump sleep. These bits contain the starting count value for the charge pump sleep down counter. While the charge pump is in sleep mode, the power mode management logic holds the charge pump sleep counter at this value. When the charge pump exits sleep power mode, the down counter delays from 0 to PSLEEP prescaled SYSCLK clock cycles before putting the charge pump into active power mode.<br>Note: The pump sleep down counter uses the same prescaled clock as Bank sleep down counter which is divided by 2 of input SYSCLK.<br>Note: BootROM configures the PSLEEP value as 0x3E8 for 100 MHz operation. Users can modify the PSLEEP value based on their application requirements if needed.<br>Reset type: SYSRSn |
| 15-1 | RESERVED | R | 0h | Reserved |
| 0 | PMPPWR | R/W | 0h | Flash Charge Pump Fallback Power Mode. This bit selects what power mode the charge pump enters after the pump active grace period (PAGP) counter has timed out.<br>0 Sleep (all pump circuits disabled)<br>1 Active (all pump circuits active)<br>Note for devices with multiple flash banks: As the pump is shared between flash banks, if an access is made either bank, the value of this bit changes to 1 (active).<br>Reset type: SYSRSn |

#### 3.14.1.14.6 FPAC2 Register (Offset = 26h) [reset = 0h]

FPAC2 is shown in Figure 3-173 and described in Table 3-197.

Return to Summary Table.

Flash Pump Access Control Register 2

**Figure 3-173. FPAC2 Register**

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

**Table 3-197. FPAC2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-0 | PAGP | R/W | 0h | Pump Active Grace Period. This register contains the starting count value for the PAGP mode down counter. Any access to flash memory causes the counter to reload with the PAGP value. After the last access to flash memory, the down counter delays from 0 to 65535 prescaled SYSCLK clock cycles before entering one of the charge pump fallback power modes as determined by PUMPPWR in the FPAC1 register.<br>Note: The PAGP down counter is clocked by the same prescaled clock as the BAGP down counter which is divided by 16 of input SYSCLK.<br>Reset type: SYSRSn |

### 3.14.1.14.7 FMSTAT Register (Offset = 2Ah) [reset = 0h]

FMSTAT is shown in Figure 3-174 and described in Table 3-198.

Return to Summary Table.

Flash Module Status Register

**Figure 3-174. FMSTAT Register**

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

**Table 3-198. 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 |

**Table 3-198. 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 | RESERVED | R | 0h | Reserved |
| 1 | RESERVED | R | 0h | Reserved |
| 0 | RESERVED | R | 0h | Reserved |

### 3.14.1.14.8 FRD_INTF_CTRL Register (Offset = 180h) [reset = 0h]

FRD_INTF_CTRL is shown in Figure 3-175 and described in Table 3-199.

Return to Summary Table.

Flash Read Interface Control Register

**Figure 3-175. 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 3-199. 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 the prefetch mechanism.<br>1 A value of 1 enables the pre-fetch mechanism.<br>Reset type: SYSRSn |

### 3.14.1.15 FLASH_ECC_REGS Registers

Table 3-200 lists the memory-mapped registers for the FLASH_ECC_REGS. All register offset addresses not listed in Table 3-200 should be considered as reserved locations and the register contents should not be modified.

**Table 3-200. 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 3-201 shows the codes that are used for access types in this section.

**Table 3-201. FLASH_ECC_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| W=1 | 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. |

**Table 3-201. FLASH_ECC_REGS Access Type Codes (continued)**

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

#### 3.14.1.15.1 ECC_ENABLE Register (Offset = 0h) [reset = Ah]

ECC_ENABLE is shown in Figure 3-176 and described in Table 3-202.

Return to Summary Table.

ECC Enable

**Figure 3-176. 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 3-202. 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 |

#### 3.14.1.15.2 SINGLE_ERR_ADDR_LOW Register (Offset = 2h) [reset = 0h]

SINGLE_ERR_ADDR_LOW is shown in Figure 3-177 and described in Table 3-203.

Return to Summary Table.

Single Error Address Low

**Figure 3-177. 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 3-203. 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 |

#### 3.14.1.15.3 SINGLE_ERR_ADDR_HIGH Register (Offset = 4h) [reset = 0h]

SINGLE_ERR_ADDR_HIGH is shown in Figure 3-178 and described in Table 3-204.

Return to Summary Table.

Single Error Address High

**Figure 3-178. 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 3-204. 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 |

#### 3.14.1.15.4 UNC_ERR_ADDR_LOW Register (Offset = 6h) [reset = 0h]

UNC_ERR_ADDR_LOW is shown in Figure 3-179 and described in Table 3-205.

Return to Summary Table.

Uncorrectable Error Address Low

**Figure 3-179. 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 3-205. 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 |

#### 3.14.1.15.5 UNC_ERR_ADDR_HIGH Register (Offset = 8h) [reset = 0h]

UNC_ERR_ADDR_HIGH is shown in Figure 3-180 and described in Table 3-206.

Return to Summary Table.

Uncorrectable Error Address High

**Figure 3-180. 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 3-206. 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 |

### 3.14.1.15.6 ERR_STATUS Register (Offset = Ah) [reset = 0h]

ERR_STATUS is shown in Figure 3-181 and described in Table 3-207.

Return to Summary Table.

Error Status

**Figure 3-181. 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 3-207. 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 |
| 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 |

**Table 3-207. ERR_STATUS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |

### 3.14.1.15.7 ERR_POS Register (Offset = Ch) [reset = 0h]

ERR_POS is shown in Figure 3-182 and described in Table 3-208.

Return to Summary Table.

Error Position

**Figure 3-182. ERR_POS Register**

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

**Table 3-208. ERR_POS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |
| 24 | ERR_TYPE_H | R/W | 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/W | 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/W | 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/W | 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 |

### 3.14.1.15.8 ERR_STATUS_CLR Register (Offset = Eh) [reset = 0h]

ERR_STATUS_CLR is shown in Figure 3-183 and described in Table 3-209.

Return to Summary Table.

Error Status Clear

**Figure 3-183. ERR_STATUS_CLR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | UNC_ERR_H_CLR | FAIL_1_H_CLR | FAIL_0_H_CLR |
| R-0h | | | | | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | UNC_ERR_L_CLR | FAIL_1_L_CLR | FAIL_0_L_CLR |
| R-0h | | | | | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 3-209. ERR_STATUS_CLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-19 | RESERVED | R | 0h | Reserved |
| 18 | UNC_ERR_H_CLR | R=0/W=1 | 0h | Uncorrectable error clear. Writing a 1 to this bit will clear the UNC_ERR_H bit of ERR_STATUS register. Writes of 0 have no effect. Read returns 0.<br>Reset type: SYSRSn |
| 17 | FAIL_1_H_CLR | R=0/W=1 | 0h | Fail on 1 clear. Writing a 1 to this bit will clear the FAIL_1_H bit of ERR_STATUS register. Writes of 0 have no effect. Read returns 0.<br>Reset type: SYSRSn |
| 16 | FAIL_0_H_CLR | R=0/W=1 | 0h | Fail on 0 clear. Writing a 1 to this bit will clear the FAIL_0_H bit of ERR_STATUS register. Writes of 0 have no effect. Read returns 0.<br>Reset type: SYSRSn |
| 15-3 | RESERVED | R | 0h | Reserved |
| 2 | UNC_ERR_L_CLR | R=0/W=1 | 0h | Uncorrectable error clear. Writing a 1 to this bit will clear the UNC_ERR_L bit of ERR_STATUS register. Writes of 0 have no effect. Read returns 0.<br>Reset type: SYSRSn |
| 1 | FAIL_1_L_CLR | R=0/W=1 | 0h | Fail on 1 clear. Writing a 1 to this bit will clear the FAIL_1_L bit of ERR_STATUS register. Writes of 0 have no effect. Read returns 0.<br>Reset type: SYSRSn |
| 0 | FAIL_0_L_CLR | R=0/W=1 | 0h | Fail on 0 clear. Writing a 1 to this bit will clear the FAIL_0_L bit of ERR_STATUS register. Writes of 0 have no effect. Read returns 0.<br>Reset type: SYSRSn |

### 3.14.1.15.9 ERR_CNT Register (Offset = 10h) [reset = 0h]

ERR_CNT is shown in Figure 3-184 and described in Table 3-210.

Return to Summary Table.

Error Control

**Figure 3-184. 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 3-210. 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 |

### 3.14.1.15.10 ERR_THRESHOLD Register (Offset = 12h) [reset = 0h]

ERR_THRESHOLD is shown in Figure 3-185 and described in Table 3-211.

Return to Summary Table.

Error Threshold

**Figure 3-185. 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 3-211. 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 |

### 3.14.1.15.11 ERR_INTFLG Register (Offset = 14h) [reset = 0h]

ERR_INTFLG is shown in Figure 3-186 and described in Table 3-212.

Return to Summary Table.

Error Interrupt Flag

**Figure 3-186. 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 3-212. 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 |

### 3.14.1.15.12 ERR_INTCLR Register (Offset = 16h) [reset = 0h]

ERR_INTCLR is shown in Figure 3-187 and described in Table 3-213.

Return to Summary Table.

Error Interrupt Flag Clear

**Figure 3-187. ERR_INTCLR Register**

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

**Table 3-213. ERR_INTCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-2 | RESERVED | R | 0h | Reserved |
| 1 | UNC_ERR_INTCLR | R=0/W=1 | 0h | Uncorrectable bit error interrupt flag clear. Writing a 1 to this bit will clear UNC_ERR_INT_FLG. Writes of 0 have no effect.<br>Reset type: SYSRSn |
| 0 | SINGLE_ERR_INTCLR | R=0/W=1 | 0h | Single bit error interrupt flag clear. Writing a 1 to this bit will clear SINGLE_ERR_INT_FLG. Writes of 0 have no effect.<br>Reset type: SYSRSn |

#### 3.14.1.15.13 FDATAH_TEST Register (Offset = 18h) [reset = 0h]

FDATAH_TEST is shown in Figure 3-188 and described in Table 3-214.

Return to Summary Table.

Data High Test

**Figure 3-188. 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 3-214. 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 |

#### 3.14.1.15.14 FDATAL_TEST Register (Offset = 1Ah) [reset = 0h]

FDATAL_TEST is shown in Figure 3-189 and described in Table 3-215.

Return to Summary Table.

Data Low Test

**Figure 3-189. 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 3-215. 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 |

### 3.14.1.15.15 FADDR_TEST Register (Offset = 1Ch) [reset = 0h]

FADDR_TEST is shown in Figure 3-190 and described in Table 3-216.

Return to Summary Table.

ECC Test Address

**Figure 3-190. 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 3-216. 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 |

### 3.14.1.15.16 FECC_TEST Register (Offset = 1Eh) [reset = 0h]

FECC_TEST is shown in Figure 3-191 and described in Table 3-217.

Return to Summary Table.

ECC Test Address

**Figure 3-191. 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 3-217. 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 |

### 3.14.1.15.17 FECC_CTRL Register (Offset = 20h) [reset = 0h]

FECC_CTRL is shown in Figure 3-192 and described in Table 3-218.

Return to Summary Table.

ECC Control

**Figure 3-192. FECC_CTRL Register**

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

**Table 3-218. FECC_CTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-3 | RESERVED | R | 0h | Reserved |
| 2 | DO_ECC_CALC | R=0/W=1 | 0h | Enable ECC calculation. ECC logic will calculate ECC in one cycle for the data and address written in ECC test registers when ECC test logic is enabled by setting ECC_TEST_EN.<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 |

### 3.14.1.15.18 FOUTH_TEST Register (Offset = 22h) [reset = 0h]

FOUTH_TEST is shown in Figure 3-193 and described in Table 3-219.

Return to Summary Table.

Test Data Out High

**Figure 3-193. 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 3-219. 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 |

#### 3.14.1.15.19 FOUTL_TEST Register (Offset = 24h) [reset = 0h]

FOUTL_TEST is shown in Figure 3-194 and described in Table 3-220.

Return to Summary Table.

Test Data Out Low

**Figure 3-194. 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 3-220. 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 |

### 3.14.1.15.20 FECC_STATUS Register (Offset = 26h) [reset = 0h]

FECC_STATUS is shown in Figure 3-195 and described in Table 3-221.

Return to Summary Table.

ECC Status

**Figure 3-195. 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 3-221. 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 |

### 3.14.1.16 MEM_CFG_REGS Registers

Table 3-222 lists the memory-mapped registers for the MEM_CFG_REGS. All register offset addresses not listed in Table 3-222 should be considered as reserved locations and the register contents should not be modified.

**Table 3-222. 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 |
| 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 |
| 60h | MSGxLOCK | Message RAM Config Lock Register | | Go |
| 62h | MSGxCOMMIT | Message RAM Config Lock Commit 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 3-223 shows the codes that are used for access types in this section.

**Table 3-223. MEM_CFG_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| W=1 | W | Write |
| WSOnce | SOnce<br>W | Set once<br>Write |

**Table 3-223. MEM_CFG_REGS Access Type Codes (continued)**

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

#### 3.14.1.16.1 DxLOCK Register (Offset = 0h) [reset = 0h]

DxLOCK is shown in Figure 3-196 and described in Table 3-224.

Return to Summary Table.

Dedicated RAM Config Lock Register

**Figure 3-196. DxLOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | RESERVED | LOCK_M1 | LOCK_M0 |
| R-0h | | | | R-0h | R-0h | R/W-0h | R/W-0h |

**Table 3-224. DxLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1 | LOCK_M1 | R/W | 0h | Locks the write to access protection, master select, initialization control and test register fields for M1 RAM:<br>0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed.<br>1: Write to ACCPROT, TEST, INIT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 0 | LOCK_M0 | R/W | 0h | Locks the write to access protection, master select, initialization control and test register fields for M0 RAM:<br>0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed.<br>1: Write to ACCPROT, TEST, INIT and Mselect fields are blocked.<br>Reset type: SYSRSn |

#### 3.14.1.16.2 DxCOMMIT Register (Offset = 2h) [reset = 0h]

DxCOMMIT is shown in Figure 3-197 and described in Table 3-225.

Return to Summary Table.

Dedicated RAM Config Lock Commit Register

**Figure 3-197. DxCOMMIT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | RESERVED | COMMIT_M1 | COMMIT_M0 |
| R-0h | | | | R-0h | R-0h | R/WSOnce-0h | R/WSOnce-0h |

**Table 3-225. DxCOMMIT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1 | COMMIT_M1 | R/WSOnce | 0h | Permanently Locks the write to access protection, master select, initialization control and test register fields for M1 RAM:<br>0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed based on value of lock field in DxLOCK register.<br>1: Write to ACCPROT, TEST, INIT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 0 | COMMIT_M0 | R/WSOnce | 0h | Permanently Locks the write to access protection, master select, initialization control and test register fields for M0 RAM:<br>0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed based on value of lock field in DxLOCK register.<br>1: Write to ACCPROT, TEST, INIT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |

### 3.14.1.16.3 DxACCPROT0 Register (Offset = 8h) [reset = 0h]

DxACCPROT0 is shown in Figure 3-198 and described in Table 3-226.

Return to Summary Table.

Dedicated RAM Config Register

**Figure 3-198. DxACCPROT0 Register**

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

**Table 3-226. DxACCPROT0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-26 | RESERVED | R | 0h | Reserved |
| 25 | RESERVED | R | 0h | Reserved |
| 24 | RESERVED | R | 0h | Reserved |
| 23-18 | RESERVED | R | 0h | Reserved |
| 17 | RESERVED | R | 0h | Reserved |
| 16 | RESERVED | R | 0h | Reserved |
| 15-0 | RESERVED | R | 0h | Reserved |

#### 3.14.1.16.4 DxTEST Register (Offset = 10h) [reset = 0h]

DxTEST is shown in Figure 3-199 and described in Table 3-227.

Return to Summary Table.

Dedicated RAM TEST Register

**Figure 3-199. DxTEST Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | RESERVED | | TEST_M1 | | TEST_M0 | |
| R-0h | | R-0h | | R/W-0h | | R/W-0h | |

**Table 3-227. DxTEST Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-6 | RESERVED | R | 0h | Reserved |
| 5-4 | RESERVED | R | 0h | Reserved |
| 3-2 | TEST_M1 | R/W | 0h | Selects the defferent 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 defferent 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 |

### 3.14.1.16.5 DxINIT Register (Offset = 12h) [reset = 0h]

DxINIT is shown in Figure 3-200 and described in Table 3-228.

Return to Summary Table.

Dedicated RAM Init Register

**Figure 3-200. DxINIT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | RESERVED | INIT_M1 | INIT_M0 |
| R-0h | | | | R-0h | R-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 3-228. DxINIT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1 | INIT_M1 | R=0/W=1 | 0h | RAM Initialization control for M1 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 0 | INIT_M0 | R=0/W=1 | 0h | RAM Initialization control for M0 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |

#### 3.14.1.16.6 DxINITDONE Register (Offset = 14h) [reset = 0h]

DxINITDONE is shown in Figure 3-201 and described in Table 3-229.

Return to Summary Table.

Dedicated RAM InitDone Status Register

**Figure 3-201. DxINITDONE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | RESERVED | INITDONE_M1 | INITDONE_M0 |
| R-0h | | | | R-0h | R-0h | R-0h | R-0h |

**Table 3-229. DxINITDONE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1 | INITDONE_M1 | R | 0h | RAM Initialization status for M1 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization has completed.<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 |

### 3.14.1.16.7 LSxLOCK Register (Offset = 20h) [reset = 0h]

LSxLOCK is shown in Figure 3-202 and described in Table 3-230.

Return to Summary Table.

Local Shared RAM Config Lock Register

**Figure 3-202. LSxLOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LOCK_LS7 | LOCK_LS6 | LOCK_LS5 | LOCK_LS4 | LOCK_LS3 | LOCK_LS2 | LOCK_LS1 | LOCK_LS0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-230. LSxLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | LOCK_LS7 | R/W | 0h | Locks the write to access protection, master select, program or data memory select, initialization control and register fields test for LS7 RAM:<br>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed.<br>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 6 | LOCK_LS6 | R/W | 0h | Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS6 RAM:<br>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed.<br>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 5 | LOCK_LS5 | R/W | 0h | Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS5 RAM:<br>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed.<br>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are blocked.<br>Reset type: SYSRSn |

**Table 3-230. LSxLOCK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | LOCK_LS4 | R/W | 0h | Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS4 RAM:<br>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed.<br>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 3 | LOCK_LS3 | R/W | 0h | Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS3 RAM:<br>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed.<br>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 2 | LOCK_LS2 | R/W | 0h | Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS2 RAM:<br>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed.<br>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 1 | LOCK_LS1 | R/W | 0h | Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS1 RAM:<br>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed.<br>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 0 | LOCK_LS0 | R/W | 0h | Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS0 RAM:<br>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed.<br>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are blocked.<br>Reset type: SYSRSn |

### 3.14.1.16.8 LSxCOMMIT Register (Offset = 22h) [reset = 0h]

LSxCOMMIT is shown in Figure 3-203 and described in Table 3-231.

Return to Summary Table.

Local Shared RAM Config Lock Commit Register

**Figure 3-203. LSxCOMMIT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| COMMIT_LS7 | COMMIT_LS6 | COMMIT_LS5 | COMMIT_LS4 | COMMIT_LS3 | COMMIT_LS2 | COMMIT_LS1 | COMMIT_LS0 |
| R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h |

**Table 3-231. LSxCOMMIT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | COMMIT_LS7 | R/WSOnce | 0h | Permanently Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS7 RAM:<br>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed based on value of lock field in LSxLOCK register.<br>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 6 | COMMIT_LS6 | R/WSOnce | 0h | Permanently Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS6 RAM:<br>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed based on value of lock field in LSxLOCK register.<br>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 5 | COMMIT_LS5 | R/WSOnce | 0h | Permanently Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS5 RAM:<br>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed based on value of lock field in LSxLOCK register.<br>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |

**Table 3-231. LSxCOMMIT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | COMMIT_LS4 | R/WSOnce | 0h | Permanently Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS4 RAM:<br>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed based on value of lock field in LSxLOCK register.<br>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 3 | COMMIT_LS3 | R/WSOnce | 0h | Permanently Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS3 RAM:<br>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed based on value of lock field in LSxLOCK register.<br>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 2 | COMMIT_LS2 | R/WSOnce | 0h | Permanently Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS2 RAM:<br>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed based on value of lock field in LSxLOCK register.<br>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 1 | COMMIT_LS1 | R/WSOnce | 0h | Permanently Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS1 RAM:<br>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed based on value of lock field in LSxLOCK register.<br>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 0 | COMMIT_LS0 | R/WSOnce | 0h | Permanently Locks the write to access protection, master select, program or data memory select, initialization control and test register fields for LS0 RAM:<br>0: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are allowed based on value of lock field in LSxLOCK register.<br>1: Write to ACCPROT, TEST, INIT, CLAPGM and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |

### 3.14.1.16.9 LSxMSEL Register (Offset = 24h) [reset = 0h]

LSxMSEL is shown in Figure 3-204 and described in Table 3-232.

Return to Summary Table.

Local Shared RAM Master Sel Register

**Figure 3-204. LSxMSEL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MSEL_LS7 | | MSEL_LS6 | | MSEL_LS5 | | MSEL_LS4 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MSEL_LS3 | | MSEL_LS2 | | MSEL_LS1 | | MSEL_LS0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 3-232. LSxMSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-14 | MSEL_LS7 | R/W | 0h | Master Select for LS7 RAM:<br>00: Memory is dedicated to CPU.<br>01: Memory is shared between CPU and CLA1 if CLAPGM_LSx bit in LSxCLAPGM register is programmed as '0'.<br>10: Reserved.<br>11: Reserved.<br>Reset type: SYSRSn |
| 13-12 | MSEL_LS6 | R/W | 0h | Master Select for LS6 RAM:<br>00: Memory is dedicated to CPU.<br>01: Memory is shared between CPU and CLA1 if CLAPGM_LSx bit in LSxCLAPGM register is programmed as '0'.<br>10: Reserved.<br>11: Reserved.<br>Reset type: SYSRSn |
| 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 if CLAPGM_LSx bit in LSxCLAPGM register is programmed as '0'.<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 if CLAPGM_LSx bit in LSxCLAPGM register is programmed as '0'.<br>10: Reserved.<br>11: Reserved.<br>Reset type: SYSRSn |

**Table 3-232. LSxMSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 if CLAPGM_LSx bit in LSxCLAPGM register is programmed as '0'.<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 if CLAPGM_LSx bit in LSxCLAPGM register is programmed as '0'.<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 if CLAPGM_LSx bit in LSxCLAPGM register is programmed as '0'.<br>10: Reserved.<br>11: Reserved.<br>Reset type: SYSRSn |
| 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 if CLAPGM_LSx bit in LSxCLAPGM register is programmed as '0'.<br>10: Reserved.<br>11: Reserved.<br>Reset type: SYSRSn |

### 3.14.1.16.10 LSxCLAPGM Register (Offset = 26h) [reset = 0h]

LSxCLAPGM is shown in Figure 3-205 and described in Table 3-233.

Return to Summary Table.

Local Shared RAM Prog/Exe control Register

**Figure 3-205. LSxCLAPGM Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CLAPGM_LS7 | CLAPGM_LS6 | CLAPGM_LS5 | CLAPGM_LS4 | CLAPGM_LS3 | CLAPGM_LS2 | CLAPGM_LS1 | CLAPGM_LS0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-233. LSxCLAPGM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | CLAPGM_LS7 | R/W | 0h | Selects LS7 RAM as program vs data memory for CLA:<br>0: CLA Data memory.<br>1: CLA Program memory.<br>Reset type: SYSRSn |
| 6 | CLAPGM_LS6 | R/W | 0h | Selects LS6 RAM as program vs data memory for CLA:<br>0: CLA Data memory.<br>1: CLA Program memory.<br>Reset type: SYSRSn |
| 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 |

**Table 3-233. LSxCLAPGM Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | CLAPGM_LS1 | R/W | 0h | Selects LS1 RAM as program vs data memory for CLA:<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 |

### 3.14.1.16.11 LSxACCPROT0 Register (Offset = 28h) [reset = 0h]

LSxACCPROT0 is shown in Figure 3-206 and described in Table 3-234.

Return to Summary Table.

Local Shared RAM Config Register 0

**Figure 3-206. 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 3-234. 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 |

**Table 3-234. LSxACCPROT0 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

### 3.14.1.16.12 LSxACCPROT1 Register (Offset = 2Ah) [reset = 0h]

LSxACCPROT1 is shown in Figure 3-207 and described in Table 3-235.

Return to Summary Table.

Local Shared RAM Config Register 1

**Figure 3-207. LSxACCPROT1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | CPUWRPROT_LS7 | FETCHPROT_LS7 |
| R-0h | | | | | | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | CPUWRPROT_LS6 | FETCHPROT_LS6 |
| R-0h | | | | | | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | CPUWRPROT_LS5 | FETCHPROT_LS5 |
| R-0h | | | | | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | CPUWRPROT_LS4 | FETCHPROT_LS4 |
| R-0h | | | | | | R/W-0h | R/W-0h |

**Table 3-235. LSxACCPROT1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-26 | RESERVED | R | 0h | Reserved |
| 25 | CPUWRPROT_LS7 | R/W | 0h | CPU WR Protection For LS7 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 24 | FETCHPROT_LS7 | R/W | 0h | Fetch Protection For LS7 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_LS6 | R/W | 0h | CPU WR Protection For LS6 RAM:<br>0: CPU Writes are allowed.<br>1: CPU Writes are blocked.<br>Reset type: SYSRSn |
| 16 | FETCHPROT_LS6 | R/W | 0h | Fetch Protection For LS6 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_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 |

**Table 3-235. LSxACCPROT1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |

### 3.14.1.16.13 LSxTEST Register (Offset = 30h) [reset = 0h]

LSxTEST is shown in Figure 3-208 and described in Table 3-236.

Return to Summary Table.

Local Shared RAM TEST Register

**Figure 3-208. LSxTEST Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TEST_LS7 | | TEST_LS6 | | TEST_LS5 | | TEST_LS4 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TEST_LS3 | | TEST_LS2 | | TEST_LS1 | | TEST_LS0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 3-236. LSxTEST Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-14 | TEST_LS7 | R/W | 0h | Selects the defferent modes for LS7 RAM:<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_LS6 | R/W | 0h | Selects the defferent modes for LS6 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_LS5 | R/W | 0h | Selects the defferent 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 defferent 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 |

**Table 3-236. LSxTEST Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-6 | TEST_LS3 | R/W | 0h | Selects the defferent 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 defferent 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 defferent 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 |
| 1-0 | TEST_LS0 | R/W | 0h | Selects the defferent 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 |

### 3.14.1.16.14 LSxINIT Register (Offset = 32h) [reset = 0h]

LSxINIT is shown in Figure 3-209 and described in Table 3-237.

Return to Summary Table.

Local Shared RAM Init Register

**Figure 3-209. LSxINIT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INIT_LS7 | INIT_LS6 | INIT_LS5 | INIT_LS4 | INIT_LS3 | INIT_LS2 | INIT_LS1 | INIT_LS0 |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 3-237. LSxINIT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | INIT_LS7 | R=0/W=1 | 0h | RAM Initialization control for LS7 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 6 | INIT_LS6 | R=0/W=1 | 0h | RAM Initialization control for LS6 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 5 | INIT_LS5 | R=0/W=1 | 0h | RAM Initialization control for LS5 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 4 | INIT_LS4 | R=0/W=1 | 0h | RAM Initialization control for LS4 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 3 | INIT_LS3 | R=0/W=1 | 0h | RAM Initialization control for LS3 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 2 | INIT_LS2 | R=0/W=1 | 0h | RAM Initialization control for LS2 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |

**Table 3-237. LSxINIT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | INIT_LS1 | R=0/W=1 | 0h | RAM Initialization control for LS1 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 0 | INIT_LS0 | R=0/W=1 | 0h | RAM Initialization control for LS0 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |

### 3.14.1.16.15 LSxINITDONE Register (Offset = 34h) [reset = 0h]

LSxINITDONE is shown in Figure 3-210 and described in Table 3-238.

Return to Summary Table.

Local Shared RAM InitDone Status Register

**Figure 3-210. LSxINITDONE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INITDONE_LS7 | INITDONE_LS6 | INITDONE_LS5 | INITDONE_LS4 | INITDONE_LS3 | INITDONE_LS2 | INITDONE_LS1 | INITDONE_LS0 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 3-238. LSxINITDONE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | INITDONE_LS7 | R | 0h | RAM Initialization status for LS7 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 6 | INITDONE_LS6 | R | 0h | RAM Initialization status for LS6 RAM:<br>0: RAM Initialization is not done.<br>1: RAM Initialization is done.<br>Reset type: SYSRSn |
| 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 |

**Table 3-238. LSxINITDONE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |

### 3.14.1.16.16 GSxLOCK Register (Offset = 40h) [reset = 0h]

GSxLOCK is shown in Figure 3-211 and described in Table 3-239.

Return to Summary Table.

Global Shared RAM Config Lock Register

**Figure 3-211. GSxLOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | RESERVED | RESERVED | RESERVED | LOCK_GS3 | LOCK_GS2 | LOCK_GS1 | LOCK_GS0 |
| R-0h | R-0h | R-0h | R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 3-239. GSxLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15 | RESERVED | R | 0h | Reserved |
| 14 | RESERVED | R | 0h | Reserved |
| 13 | RESERVED | R | 0h | Reserved |
| 12 | RESERVED | R | 0h | Reserved |
| 11 | RESERVED | R | 0h | Reserved |
| 10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | LOCK_GS3 | R/W | 0h | Locks the write to access protection, master select, initialization control and test register fields for GS3 RAM:<br>0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed.<br>1: Write to ACCPROT, TEST, INIT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 2 | LOCK_GS2 | R/W | 0h | Locks the write to access protection, master select, initialization control and test register fields for GS2 RAM:<br>0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed.<br>1: Write to ACCPROT, TEST, INIT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 1 | LOCK_GS1 | R/W | 0h | Locks the write to access protection, master select, initialization control and test register fields for GS1 RAM:<br>0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed.<br>1: Write to ACCPROT, TEST, INIT and Mselect fields are blocked.<br>Reset type: SYSRSn |

**Table 3-239. GSxLOCK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | LOCK_GS0 | R/W | 0h | Locks the write to access protection, master select, initialization control and test register fields for GS0 RAM:<br>0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed.<br>1: Write to ACCPROT, TEST, INIT and Mselect fields are blocked.<br>Reset type: SYSRSn |

### 3.14.1.16.17 GSxCOMMIT Register (Offset = 42h) [reset = 0h]

GSxCOMMIT is shown in Figure 3-212 and described in Table 3-240.

Return to Summary Table.

Global Shared RAM Config Lock Commit Register

**Figure 3-212. GSxCOMMIT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | RESERVED | RESERVED | RESERVED | COMMIT_GS3 | COMMIT_GS2 | COMMIT_GS1 | COMMIT_GS0 |
| R-0h | R-0h | R-0h | R-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h |

**Table 3-240. GSxCOMMIT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15 | RESERVED | R | 0h | Reserved |
| 14 | RESERVED | R | 0h | Reserved |
| 13 | RESERVED | R | 0h | Reserved |
| 12 | RESERVED | R | 0h | Reserved |
| 11 | RESERVED | R | 0h | Reserved |
| 10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | COMMIT_GS3 | R/WSOnce | 0h | Permanently Locks the write to access protection, master select, initialization control and test register fields for GS3 RAM:<br>0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed based on value of lock field in GSxLOCK register.<br>1: Write to ACCPROT, TEST, INIT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 2 | COMMIT_GS2 | R/WSOnce | 0h | Permanently Locks the write to access protection, master select, initialization control and test register fields for GS2 RAM:<br>0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed based on value of lock field in GSxLOCK register.<br>1: Write to ACCPROT, TEST, INIT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |

**Table 3-240. GSxCOMMIT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | COMMIT_GS1 | R/WSOnce | 0h | Permanently Locks the write to access protection, master select, initialization control and test register fields for GS1 RAM:<br>0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed based on value of lock field in GSxLOCK register.<br>1: Write to ACCPROT, TEST, INIT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |
| 0 | COMMIT_GS0 | R/WSOnce | 0h | Permanently Locks the write to access protection, master select, initialization control and test register fields for GS0 RAM:<br>0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed based on value of lock field in GSxLOCK register.<br>1: Write to ACCPROT, TEST, INIT and Mselect fields are permanently blocked.<br>Reset type: SYSRSn |

### 3.14.1.16.18 GSxACCPROT0 Register (Offset = 48h) [reset = 0h]

GSxACCPROT0 is shown in Figure 3-213 and described in Table 3-241.

Return to Summary Table.

Global Shared RAM Config Register 0

**Figure 3-213. 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 3-241. 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 |

**Table 3-241. GSxACCPROT0 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

### 3.14.1.16.19 GSxACCPROT1 Register (Offset = 4Ah) [reset = 0h]

GSxACCPROT1 is shown in Figure 3-214 and described in Table 3-242.

Return to Summary Table.

Global Shared RAM Config Register 1

**Figure 3-214. GSxACCPROT1 Register**

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

**Table 3-242. GSxACCPROT1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-27 | RESERVED | R | 0h | Reserved |
| 26 | RESERVED | R | 0h | Reserved |
| 25 | RESERVED | R | 0h | Reserved |
| 24 | RESERVED | R | 0h | Reserved |
| 23-19 | RESERVED | R | 0h | Reserved |
| 18 | RESERVED | R | 0h | Reserved |
| 17 | RESERVED | R | 0h | Reserved |
| 16 | RESERVED | R | 0h | Reserved |
| 15-11 | RESERVED | R | 0h | Reserved |
| 10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 7-3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1 | RESERVED | R | 0h | Reserved |
| 0 | RESERVED | R | 0h | Reserved |

### 3.14.1.16.20 GSxACCPROT2 Register (Offset = 4Ch) [reset = 0h]

GSxACCPROT2 is shown in Figure 3-215 and described in Table 3-243.

Return to Summary Table.

Global Shared RAM Config Register 2

**Figure 3-215. GSxACCPROT2 Register**

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

**Table 3-243. GSxACCPROT2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-27 | RESERVED | R | 0h | Reserved |
| 26 | RESERVED | R | 0h | Reserved |
| 25 | RESERVED | R | 0h | Reserved |
| 24 | RESERVED | R | 0h | Reserved |
| 23-19 | RESERVED | R | 0h | Reserved |
| 18 | RESERVED | R | 0h | Reserved |
| 17 | RESERVED | R | 0h | Reserved |
| 16 | RESERVED | R | 0h | Reserved |
| 15-11 | RESERVED | R | 0h | Reserved |
| 10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 7-3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1 | RESERVED | R | 0h | Reserved |
| 0 | RESERVED | R | 0h | Reserved |

### 3.14.1.16.21 GSxACCPROT3 Register (Offset = 4Eh) [reset = 0h]

GSxACCPROT3 is shown in Figure 3-216 and described in Table 3-244.

Return to Summary Table.

Global Shared RAM Config Register 3

**Figure 3-216. GSxACCPROT3 Register**

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

**Table 3-244. GSxACCPROT3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-27 | RESERVED | R | 0h | Reserved |
| 26 | RESERVED | R | 0h | Reserved |
| 25 | RESERVED | R | 0h | Reserved |
| 24 | RESERVED | R | 0h | Reserved |
| 23-19 | RESERVED | R | 0h | Reserved |
| 18 | RESERVED | R | 0h | Reserved |
| 17 | RESERVED | R | 0h | Reserved |
| 16 | RESERVED | R | 0h | Reserved |
| 15-11 | RESERVED | R | 0h | Reserved |
| 10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 7-3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1 | RESERVED | R | 0h | Reserved |
| 0 | RESERVED | R | 0h | Reserved |

### 3.14.1.16.22 GSxTEST Register (Offset = 50h) [reset = 0h]

GSxTEST is shown in Figure 3-217 and described in Table 3-245.

Return to Summary Table.

Global Shared RAM TEST Register

**Figure 3-217. GSxTEST Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | RESERVED | | RESERVED | | RESERVED | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | RESERVED | | RESERVED | | RESERVED | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | RESERVED | | RESERVED | | RESERVED | |
| R-0h | | R-0h | | R-0h | | R-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 3-245. GSxTEST Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | RESERVED | R | 0h | Reserved |
| 29-28 | RESERVED | R | 0h | Reserved |
| 27-26 | RESERVED | R | 0h | Reserved |
| 25-24 | RESERVED | R | 0h | Reserved |
| 23-22 | RESERVED | R | 0h | Reserved |
| 21-20 | RESERVED | R | 0h | Reserved |
| 19-18 | RESERVED | R | 0h | Reserved |
| 17-16 | RESERVED | R | 0h | Reserved |
| 15-14 | RESERVED | R | 0h | Reserved |
| 13-12 | RESERVED | R | 0h | Reserved |
| 11-10 | RESERVED | R | 0h | Reserved |
| 9-8 | RESERVED | R | 0h | Reserved |
| 7-6 | TEST_GS3 | R/W | 0h | Selects the defferent 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 defferent 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 |

**Table 3-245. GSxTEST Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | TEST_GS1 | R/W | 0h | Selects the defferent 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 defferent 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 |

### 3.14.1.16.23 GSxINIT Register (Offset = 52h) [reset = 0h]

GSxINIT is shown in Figure 3-218 and described in Table 3-246.

Return to Summary Table.

Global Shared RAM Init Register

**Figure 3-218. GSxINIT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | RESERVED | RESERVED | RESERVED | INIT_GS3 | INIT_GS2 | INIT_GS1 | INIT_GS0 |
| R-0h | R-0h | R-0h | R-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 3-246. GSxINIT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15 | RESERVED | R | 0h | Reserved |
| 14 | RESERVED | R | 0h | Reserved |
| 13 | RESERVED | R | 0h | Reserved |
| 12 | RESERVED | R | 0h | Reserved |
| 11 | RESERVED | R | 0h | Reserved |
| 10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | INIT_GS3 | R=0/W=1 | 0h | RAM Initialization control for GS3 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 2 | INIT_GS2 | R=0/W=1 | 0h | RAM Initialization control for GS2 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 1 | INIT_GS1 | R=0/W=1 | 0h | RAM Initialization control for GS1 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |

**Table 3-246. GSxINIT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INIT_GS0 | R=0/W=1 | 0h | RAM Initialization control for GS0 RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |

### 3.14.1.16.24 GSxINITDONE Register (Offset = 54h) [reset = 0h]

GSxINITDONE is shown in Figure 3-219 and described in Table 3-247.

Return to Summary Table.

Global Shared RAM InitDone Status Register

**Figure 3-219. GSxINITDONE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | RESERVED | RESERVED | RESERVED | 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 3-247. GSxINITDONE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15 | RESERVED | R | 0h | Reserved |
| 14 | RESERVED | R | 0h | Reserved |
| 13 | RESERVED | R | 0h | Reserved |
| 12 | RESERVED | R | 0h | Reserved |
| 11 | RESERVED | R | 0h | Reserved |
| 10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | INITDONE_GS3 | R | 0h | RAM Initialization status for GS3 RAM:<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 |

**Table 3-247. GSxINITDONE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |

### 3.14.1.16.25 MSGxLOCK Register (Offset = 60h) [reset = 0h]

MSGxLOCK is shown in Figure 3-220 and described in Table 3-248.

Return to Summary Table.

Message RAM Config Lock Register

**Figure 3-220. MSGxLOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | RESERVED | RESERVED | LOCK_CLA1TOCPU | LOCK_CPUTOCLA1 | RESERVED |
| R-0h | | | R-0h | R-0h | R/W-0h | R/W-0h | R-0h |

**Table 3-248. MSGxLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | LOCK_CLA1TOCPU | R/W | 0h | Locks the write to access protection, master select, initialization control and test register fields for CLA1TOCPU RAM:<br>0: Write to TEST, INIT fields are allowed.<br>1: Write to TEST, INIT fields are blocked.<br>Reset type: SYSRSn |
| 1 | LOCK_CPUTOCLA1 | R/W | 0h | Locks the write to access protection, master select, initialization control and test register fields for CPUTOCLA1 RAM:<br>0: Write to TEST, INIT fields are allowed.<br>1: Write to TEST, INIT fields are blocked.<br>Reset type: SYSRSn |
| 0 | RESERVED | R | 0h | Reserved |

### 3.14.1.16.26 MSGxCOMMIT Register (Offset = 62h) [reset = 0h]

MSGxCOMMIT is shown in Figure 3-221 and described in Table 3-249.

Return to Summary Table.

Message RAM Config Lock Commit Register

**Figure 3-221. MSGxCOMMIT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | RESERVED | RESERVED | COMMIT_CLA1TOCPU | COMMIT_CPUTOCLA1 | RESERVED |
| R-0h | | | R-0h | R-0h | R/WSOnce-0h | R/WSOnce-0h | R-0h |

**Table 3-249. MSGxCOMMIT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | COMMIT_CLA1TOCPU | R/WSOnce | 0h | Locks the write to access protection, master select, initialization control and test register fields for CLA1TOCPU RAM:<br>0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed.<br>1: Write to ACCPROT, TEST, INIT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 1 | COMMIT_CPUTOCLA1 | R/WSOnce | 0h | Locks the write to access protection, master select, initialization control and test register fields for CPUTOCLA1 RAM:<br>0: Write to ACCPROT, TEST, INIT and Mselect fields are allowed.<br>1: Write to ACCPROT, TEST, INIT and Mselect fields are blocked.<br>Reset type: SYSRSn |
| 0 | RESERVED | R | 0h | Reserved |

### 3.14.1.16.27 MSGxTEST Register (Offset = 70h) [reset = 0h]

MSGxTEST is shown in Figure 3-222 and described in Table 3-250.

Return to Summary Table.

Message RAM TEST Register

**Figure 3-222. MSGxTEST Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | RESERVED | |
| R-0h | | | | | | R-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | TEST_CLA1TOCPU | | TEST_CPUTOCLA1 | | RESERVED | |
| R-0h | | R/W-0h | | R/W-0h | | R-0h | |

**Table 3-250. MSGxTEST Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-10 | RESERVED | R | 0h | Reserved |
| 9-8 | RESERVED | R | 0h | Reserved |
| 7-6 | RESERVED | R | 0h | Reserved |
| 5-4 | TEST_CLA1TOCPU | R/W | 0h | Selects the defferent 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 defferent 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 | RESERVED | R | 0h | Reserved |

### 3.14.1.16.28 MSGxINIT Register (Offset = 72h) [reset = 0h]

MSGxINIT is shown in Figure 3-223 and described in Table 3-251.

Return to Summary Table.

Message RAM Init Register

**Figure 3-223. 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 | RESERVED |
| R-0h | | | R-0h | R-0h | R=0/W=1-0h | R=0/W=1-0h | R-0h |

**Table 3-251. MSGxINIT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | INIT_CLA1TOCPU | R=0/W=1 | 0h | RAM Initialization control for CLA1TOCPU MSG RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 1 | INIT_CPUTOCLA1 | R=0/W=1 | 0h | RAM Initialization control for CPUTOCLA1 MSG RAM:<br>0: None.<br>1: Start RAM Initialization.<br>Reset type: SYSRSn |
| 0 | RESERVED | R | 0h | Reserved |

### 3.14.1.16.29 MSGxINITDONE Register (Offset = 74h) [reset = 0h]

MSGxINITDONE is shown in Figure 3-224 and described in Table 3-252.

Return to Summary Table.

Message RAM InitDone Status Register

**Figure 3-224. 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 | RESERVED |
| R-0h | | | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 3-252. 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 | RESERVED | R | 0h | Reserved |

### 3.14.1.17 MEMORY_ERROR_REGS Registers

Table 3-253 lists the memory-mapped registers for the MEMORY_ERROR_REGS. All register offset addresses not listed in Table 3-253 should be considered as reserved locations and the register contents should not be modified.

**Table 3-253. 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 3-254 shows the codes that are used for access types in this section.

**Table 3-254. MEMORY_ERROR_REGS Access Type Codes**

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

### 3.14.1.17.1 UCERRFLG Register (Offset = 0h) [reset = 0h]

UCERRFLG is shown in Figure 3-225 and described in Table 3-255.

Return to Summary Table.

Uncorrectable Error Flag Register

**Figure 3-225. 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 3-255. 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 |

### 3.14.1.17.2 UCERRSET Register (Offset = 2h) [reset = 0h]

UCERRSET is shown in Figure 3-226 and described in Table 3-256.

Return to Summary Table.

Uncorrectable Error Flag Set Register

**Figure 3-226. UCERRSET Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | CLA1RDERR | DMARDERR | CPURDERR |
| R-0h | | | | R-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 3-256. UCERRSET Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | CLA1RDERR | R=0/W=1 | 0h | 0: No action.<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/W=1 | 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/W=1 | 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 |

### 3.14.1.17.3 UCERRCLR Register (Offset = 4h) [reset = 0h]

UCERRCLR is shown in Figure 3-227 and described in Table 3-257.

Return to Summary Table.

Uncorrectable Error Flag Clear Register

**Figure 3-227. UCERRCLR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | CLA1RDERR | DMARDERR | CPURDERR |
| R-0h | | | | R-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 3-257. UCERRCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | CLA1RDERR | R=0/W=1 | 0h | 0: No action.<br>1: CLA1 Read Error Flag in UCERRFLG register will be cleared.<br>Reset type: SYSRSn |
| 1 | DMARDERR | R=0/W=1 | 0h | 0: No action.<br>1: DMA Read Error Flag in UCERRFLG register will be cleared .<br>Reset type: SYSRSn |
| 0 | CPURDERR | R=0/W=1 | 0h | 0: No action.<br>1: CPU Read Error Flag in UCERRFLG register will be cleared.<br>Reset type: SYSRSn |

#### 3.14.1.17.4 UCCPUREADDR Register (Offset = 6h) [reset = 0h]

UCCPUREADDR is shown in Figure 3-228 and described in Table 3-258.

Return to Summary Table.

Uncorrectable CPU Read Error Address

**Figure 3-228. 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 3-258. 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 |

#### 3.14.1.17.5 UCDMAREADDR Register (Offset = 8h) [reset = 0h]

UCDMAREADDR is shown in Figure 3-229 and described in Table 3-259.

Return to Summary Table.

Uncorrectable DMA Read Error Address

**Figure 3-229. 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 3-259. UCDMAREADDR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | UCDMAREADDR | R | 0h | This register captures the address location for which DMA read access resulted in uncorrectable Parity error.<br>Reset type: SYSRSn |

### 3.14.1.17.6 UCCLA1READDR Register (Offset = Ah) [reset = 0h]

UCCLA1READDR is shown in Figure 3-230 and described in Table 3-260.

Return to Summary Table.

Uncorrectable CLA1 Read Error Address

**Figure 3-230. 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 3-260. UCCLA1READDR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | UCCLA1READDR | R | 0h | This register captures the address location for which CLA1 read/fetch access resulted in uncorrectable Parity error.<br>Reset type: SYSRSn |

### 3.14.1.17.7 CERRFLG Register (Offset = 20h) [reset = 0h]

CERRFLG is shown in Figure 3-231 and described in Table 3-261.

Return to Summary Table.

Correctable Error Flag Register

**Figure 3-231. 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 3-261. 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 |

### 3.14.1.17.8 CERRSET Register (Offset = 22h) [reset = 0h]

CERRSET is shown in Figure 3-232 and described in Table 3-262.

Return to Summary Table.

Correctable Error Flag Set Register

**Figure 3-232. CERRSET Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | CLA1RDERR | DMARDERR | CPURDERR |
| R-0h | | | | R-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 3-262. CERRSET Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | CLA1RDERR | R=0/W=1 | 0h | 0: No action.<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/W=1 | 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/W=1 | 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 |

### 3.14.1.17.9 CERRCLR Register (Offset = 24h) [reset = 0h]

CERRCLR is shown in Figure 3-233 and described in Table 3-263.

Return to Summary Table.

Correctable Error Flag Clear Register

**Figure 3-233. CERRCLR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RESERVED | CLA1RDERR | DMARDERR | CPURDERR |
| R-0h | | | | R-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 3-263. CERRCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | CLA1RDERR | R=0/W=1 | 0h | 0: No action.<br>1: CLA1 Read Error Flag in CERRFLG register will be cleared.<br>Reset type: SYSRSn |
| 1 | DMARDERR | R=0/W=1 | 0h | 0: No action.<br>1: DMA Read Error Flag in CERRFLG register will be cleared .<br>Reset type: SYSRSn |
| 0 | CPURDERR | R=0/W=1 | 0h | 0: No action.<br>1: CPU Read Error Flag in CERRFLG register will be cleared.<br>Reset type: SYSRSn |

### 3.14.1.17.10 CCPUREADDR Register (Offset = 26h) [reset = 0h]

CCPUREADDR is shown in Figure 3-234 and described in Table 3-264.

Return to Summary Table.

Correctable CPU Read Error Address

**Figure 3-234. 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 3-264. 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 |

#### 3.14.1.17.11 CERRCNT Register (Offset = 2Eh) [reset = 0h]

CERRCNT is shown in Figure 3-235 and described in Table 3-265.

Return to Summary Table.

Correctable Error Count Register

**Figure 3-235. CERRCNT Register**

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

**Table 3-265. CERRCNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | CERRCNT | R | 0h | This register holds the count of how many times correctable error occurred.<br>Reset type: SYSRSn |

### 3.14.1.17.12 CERRTHRES Register (Offset = 30h) [reset = 0h]

CERRTHRES is shown in Figure 3-236 and described in Table 3-266.

Return to Summary Table.

Correctable Error Threshold Value Register

**Figure 3-236. CERRTHRES Register**

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

**Table 3-266. CERRTHRES Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | CERRTHRES | R/W | 0h | When value in CERRCNT register is greater than value configured in this register, corretable interrupt gets generated, if enabled.<br>Reset type: SYSRSn |

### 3.14.1.17.13 CEINTFLG Register (Offset = 32h) [reset = 0h]

CEINTFLG is shown in Figure 3-237 and described in Table 3-267.

Return to Summary Table.

Correctable Error Interrupt Flag Status Register

**Figure 3-237. 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 3-267. 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 |

### 3.14.1.17.14 CEINTCLR Register (Offset = 34h) [reset = 0h]

CEINTCLR is shown in Figure 3-238 and described in Table 3-268.

Return to Summary Table.

Correctable Error Interrupt Flag Clear Register

**Figure 3-238. CEINTCLR Register**

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

**Table 3-268. CEINTCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-1 | RESERVED | R | 0h | Reserved |
| 0 | CEINTCLR | R=0/W=1 | 0h | 0: No action.<br>1: Total corrected error count exceeded flag in CEINTFLG register will be cleared.<br>Reset type: SYSRSn |

### 3.14.1.17.15 CEINTSET Register (Offset = 36h) [reset = 0h]

CEINTSET is shown in Figure 3-239 and described in Table 3-269.

Return to Summary Table.

Correctable Error Interrupt Flag Set Register

**Figure 3-239. CEINTSET Register**

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

**Table 3-269. CEINTSET Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-1 | RESERVED | R | 0h | Reserved |
| 0 | CEINTSET | R=0/W=1 | 0h | 0: No action.<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 |

### 3.14.1.17.16 CEINTEN Register (Offset = 38h) [reset = 0h]

CEINTEN is shown in Figure 3-240 and described in Table 3-270.

Return to Summary Table.

Correctable Error Interrupt Enable Register

**Figure 3-240. 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 3-270. 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 |

### 3.14.1.18 NMI_INTRUPT_REGS Registers

Table 3-271 lists the memory-mapped registers for the NMI_INTRUPT_REGS. 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. NMI_INTRUPT_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | NMICFG | NMI Configuration Register | EALLOW | Go |
| 1h | NMIFLG | NMI Flag Register (SYSRsn 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 3-272 shows the codes that are used for access types in this section.

**Table 3-272. NMI_INTRUPT_REGS Access Type Codes**

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

#### 3.14.1.18.1 NMICFG Register (Offset = 0h) [reset = 0h]

NMICFG is shown in Figure 3-241 and described in Table 3-273.

Return to Summary Table.

NMI Configuration Register

**Figure 3-241. NMICFG Register**

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

**Table 3-273. NMICFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-1 | RESERVED | R=0 | 0h | Reserved |
| 0 | NMIE | R/W=1 | 0h | Global NMI Enable<br>This bit indicates that the NMI module has been enabled, which allows system error conditions to trigger an NMI to the CPU. The boot ROM sets this bit at start-up. It can only be cleared by a system reset.<br>Reset type: SYSRSn |

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

NMIFLG is shown in Figure 3-242 and described in Table 3-274.

Return to Summary Table.

NMI Flag Register (SYSRsn Clear)

**Figure 3-242. NMIFLG Register**

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

**Table 3-274. NMIFLG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | RESERVED | R=0 | 0h | Reserved |
| 13 | SWERR | R | 0h | Software-Forced NMI Flag<br>This bit can only be set by writing to the corresponding bit in the NMIFLGFRC register. It can be cleared by a system reset or by writing to the corresponding bit in the NMIFLGCLR register. No further NMIs are generated until this flag is cleared.<br>Reset type: SYSRSn<br>0h (R/W) = No software NMI forced<br>1h (R/W) = Software NMI forced |
| 12 | RESERVED | R | 0h | Reserved |
| 11 | RESERVED | R | 0h | Reserved |
| 10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | PIEVECTERR | R | 0h | PIE Vector Fetch Error NMI Flag<br>This bit indicates whether a mismatch was detected during an interrupt vector fetch. It can be cleared by a system reset or by writing to the corresponding bit in the NMIFLGCLR register.<br>Reset type: SYSRSn<br>0h (R/W) = No vector fetch error detected<br>1h (R/W) = Vector fetch error detected |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | FLUNCERR | R | 0h | Flash Uncorrectable Error NMI Flag<br>This bit indicates whether an uncorrectable ECC error occurred during a flash access. It can be cleared by a system reset or by writing to the corresponding bit in the NMIFLGCLR register.<br>Reset type: SYSRSn<br>0h (R/W) = No uncorrectable error detected<br>1h (R/W) = Uncorrectable error detected |

**Table 3-274. NMIFLG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | RAMUNCERR | R | 0h | RAM Uncorrectable Error NMI Flag<br>This bit indicates whether an uncorrectable ECC error has occurred during a RAM access by any master. It can be cleared by a system reset or by writing to the corresponding bit in the NMIFLGCLR register.<br>Reset type: SYSRSn<br>0h (R/W) = No uncorrectable error detected<br>1h (R/W) = Uncorrectable error detected |
| 1 | CLOCKFAIL | R | 0h | Clock Fail NMI Flag<br>This bit indicates whether a clock fail condition has been detected. It can be cleared by a system reset or by writing to the corresponding bit in the NMIFLGCLR register.<br>Reset type: SYSRSn<br>0h (R/W) = No clock fail detected<br>1h (R/W) = Clock fail detected |
| 0 | NMIINT | R | 0h | Global NMI Flag<br>This bit indicates whether an NMI has been generated. It can be cleared by a system reset or by writing to the corresponding bit in the NMIFLGCLR register. No further NMIs are generated until this flag is cleared.<br>Reset type: SYSRSn<br>0h (R/W) = No NMI generated<br>1h (R/W) = NMI generated |

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

NMIFLGCLR is shown in Figure 3-243 and described in Table 3-275.

Return to Summary Table.

Writing a 1 to one of these bits clears the corresponding bit in the NMIFLG register. Writes of 0 are ignored, and these bits always read 0. If an NMI arrives on the same cycle that this register is written, the NMI is latched. All other NMI flags must be cleared before the NMIINT flag is cleared, otherwise NMIINT will be set again.

**Figure 3-243. NMIFLGCLR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | SWERR | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| R=0-0h | | R=0/W=1-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | PIEVECTERR | RESERVED | RESERVED | FLUNCERR | RAMUNCERR | CLOCKFAIL | NMIINT |
| R-0h | R=0/W=1-0h | R-0h | R-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 3-275. NMIFLGCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | RESERVED | R=0 | 0h | Reserved |
| 13 | SWERR | R=0/W=1 | 0h | Clear the SWERR flag.<br>Reset type: SYSRSn |
| 12 | RESERVED | R | 0h | Reserved |
| 11 | RESERVED | R | 0h | Reserved |
| 10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | PIEVECTERR | R=0/W=1 | 0h | Clear the PIEVECTERR flag.<br>Reset type: SYSRSn |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | FLUNCERR | R=0/W=1 | 0h | Clear the FLUNCERR flag.<br>Reset type: SYSRSn |
| 2 | RAMUNCERR | R=0/W=1 | 0h | Clear the RAMUNCERR flag.<br>Reset type: SYSRSn |
| 1 | CLOCKFAIL | R=0/W=1 | 0h | Clear the CLOCKFAIL flag.<br>Reset type: SYSRSn |
| 0 | NMIINT | R=0/W=1 | 0h | Clear the NMIINT flag. This flag should only be cleared after all other active flags have been cleared.<br>Reset type: SYSRSn |

### 3.14.1.18.4 NMIFLGFRC Register (Offset = 3h) [reset = 0h]

NMIFLGFRC is shown in Figure 3-244 and described in Table 3-276.

Return to Summary Table.

Writing a 1 to one of these bits sets the corresponding bit in the NMIFLG register. Writes of 0 are ignored, and these bits always read 0. This register can be used to test the NMI functionality.

**Figure 3-244. NMIFLGFRC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | SWERR | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| R=0-0h | | R=0/W=1-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | PIEVECTERR | RESERVED | RESERVED | FLUNCERR | RAMUNCERR | CLOCKFAIL | RESERVED |
| R-0h | R=0/W=1-0h | R-0h | R-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0-0h |

**Table 3-276. NMIFLGFRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | RESERVED | R=0 | 0h | Reserved |
| 13 | SWERR | R=0/W=1 | 0h | Set the SWERR flag.<br>Reset type: SYSRSn |
| 12 | RESERVED | R | 0h | Reserved |
| 11 | RESERVED | R | 0h | Reserved |
| 10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | PIEVECTERR | R=0/W=1 | 0h | Set the PIEVECTERR flag.<br>Reset type: SYSRSn |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | FLUNCERR | R=0/W=1 | 0h | Set the FLUNCERR flag.<br>Reset type: SYSRSn |
| 2 | RAMUNCERR | R=0/W=1 | 0h | Set the RAMUNCERR flag.<br>Reset type: SYSRSn |
| 1 | CLOCKFAIL | R=0/W=1 | 0h | Set the CLOCKFAIL flag.<br>Reset type: SYSRSn |
| 0 | RESERVED | R=0 | 0h | Reserved |

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

NMIWDCNT is shown in Figure 3-245 and described in Table 3-277.

Return to Summary Table.

NMI Watchdog Counter Register

**Figure 3-245. NMIWDCNT Register**

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

**Table 3-277. NMIWDCNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | NMIWDCNT | R | 0h | NMI Watchdog Counter<br><br>This 16-bit counter increments once per SYSCLK cycle whenever any of the NMIFLG bits are set. If the counter reaches the period value in the NMIWDPRD register, the NMI module generates a reset (NMIWDRS). After this reset, the counter resets to zero and stops counting.<br><br>If all NMI flags are cleared, the counter will reset to zero and stop counting until another NMI flag is set.<br>Reset type: SYSRSn |

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

NMIWDPRD is shown in Figure 3-246 and described in Table 3-278.

Return to Summary Table.

NMI Watchdog Period Register

**Figure 3-246. NMIWDPRD Register**

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

**Table 3-278. NMIWDPRD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | NMIWDPRD | R/W | FFFFh | NMI Watchdog Period<br>These bits specify the period of the NMI watchdog timer in SYSCLK cycles. Writing a period value that is smaller than the current counter value will immediately force a reset (NMIWDRS).<br>Reset type: SYSRSn |

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

NMISHDFLG is shown in Figure 3-247 and described in Table 3-279.

Return to Summary Table.

These shadow flag bits are set whenever the corresponding bits in the NMIFLG register are set. The shadow flags are only reset by a power-on reset (POR), but any system or external reset will reset the normal flags. The shadow flags allow NMIs to be tracked across resets.

**Figure 3-247. NMISHDFLG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | SWERR | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| R=0-0h | | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | PIEVECTERR | RESERVED | RESERVED | FLUNCERR | RAMUNCERR | CLOCKFAIL | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R=0-0h |

**Table 3-279. NMISHDFLG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | RESERVED | R=0 | 0h | Reserved |
| 13 | SWERR | R | 0h | Shadow SWERR flag<br>Reset type: PORn |
| 12 | RESERVED | R | 0h | Reserved |
| 11 | RESERVED | R | 0h | Reserved |
| 10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | PIEVECTERR | R | 0h | Shadow PIEVECTERR flag<br>Reset type: PORn |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | FLUNCERR | R | 0h | Shadow FLUNCERR flag<br>Reset type: PORn |
| 2 | RAMUNCERR | R | 0h | Shadow RAMUNCERR flag<br>Reset type: PORn |
| 1 | CLOCKFAIL | R | 0h | Shadow CLOCKFAIL flag<br>Reset type: PORn |
| 0 | RESERVED | R=0 | 0h | Reserved |

### 3.14.1.19 PERIPH_AC_REGS Registers

Table 3-280 lists the memory-mapped registers for the PERIPH_AC_REGS. All register offset addresses not listed in Table 3-280 should be considered as reserved locations and the register contents should not be modified.

**Table 3-280. PERIPH_AC_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | ADCA_AC | ADCA Master Access Control Register | EALLOW | Go |
| 2h | ADCB_AC | ADCB Master Access Control Register | EALLOW | Go |
| 4h | ADCC_AC | ADCC Master Access Control Register | EALLOW | Go |
| 10h | CMPSS1_AC | CMPSS1 Master Access Control Register | EALLOW | Go |
| 12h | CMPSS2_AC | CMPSS2 Master Access Control Register | EALLOW | Go |
| 14h | CMPSS3_AC | CMPSS3 Master Access Control Register | EALLOW | Go |
| 16h | CMPSS4_AC | CMPSS4 Master Access Control Register | EALLOW | Go |
| 18h | CMPSS5_AC | CMPSS5 Master Access Control Register | EALLOW | Go |
| 1Ah | CMPSS6_AC | CMPSS6 Master Access Control Register | EALLOW | Go |
| 1Ch | CMPSS7_AC | CMPSS7 Master Access Control Register | EALLOW | Go |
| 28h | DACA_AC | DACA Master Access Control Register | EALLOW | Go |
| 2Ah | DACB_AC | DACB Master Access Control Register | EALLOW | Go |
| 38h | PGA1_AC | PGAA Master Access Control Register | EALLOW | Go |
| 3Ah | PGA2_AC | PGAB Master Access Control Register | EALLOW | Go |
| 3Ch | PGA3_AC | PGAC Master Access Control Register | EALLOW | Go |
| 3Eh | PGA4_AC | PGAD Master Access Control Register | EALLOW | Go |
| 40h | PGA5_AC | PGAE Master Access Control Register | EALLOW | Go |
| 42h | PGA6_AC | PGAF Master Access Control Register | EALLOW | Go |
| 44h | PGA7_AC | PGAG Master Access Control Register | EALLOW | Go |
| 48h | EPWM1_AC | EPWM1 Master Access Control Register | EALLOW | Go |
| 4Ah | EPWM2_AC | EPWM2 Master Access Control Register | EALLOW | Go |
| 4Ch | EPWM3_AC | EPWM3 Master Access Control Register | EALLOW | Go |
| 4Eh | EPWM4_AC | EPWM4 Master Access Control Register | EALLOW | Go |
| 50h | EPWM5_AC | EPWM5 Master Access Control Register | EALLOW | Go |
| 52h | EPWM6_AC | EPWM6 Master Access Control Register | EALLOW | Go |
| 54h | EPWM7_AC | EPWM7 Master Access Control Register | EALLOW | Go |
| 56h | EPWM8_AC | EPWM8 Master Access Control Register | EALLOW | Go |
| 70h | EQEP1_AC | EQEP1 Master Access Control Register | EALLOW | Go |
| 72h | EQEP2_AC | EQEP2 Master Access Control Register | EALLOW | Go |
| 80h | ECAP1_AC | ECAP1 Master Access Control Register | EALLOW | Go |
| 82h | ECAP2_AC | ECAP2 Master Access Control Register | EALLOW | Go |
| 84h | ECAP3_AC | ECAP3 Master Access Control Register | EALLOW | Go |
| 86h | ECAP4_AC | ECAP4 Master Access Control Register | EALLOW | Go |
| 88h | ECAP5_AC | ECAP5 Master Access Control Register | EALLOW | Go |
| 8Ah | ECAP6_AC | ECAP6 Master Access Control Register | EALLOW | Go |
| 8Ch | ECAP7_AC | ECAP7 Master Access Control Register | EALLOW | Go |
| A8h | SDFM1_AC | SDFM1 Master Access Control Register | EALLOW | Go |
| B0h | CLB1_AC | CLB1 Master Access Control Register | EALLOW | Go |
| B2h | CLB2_AC | CLB2 Master Access Control Register | EALLOW | Go |
| B4h | CLB3_AC | CLB3 Master Access Control Register | EALLOW | Go |
| B6h | CLB4_AC | CLB4 Master Access Control Register | EALLOW | Go |
| C0h | CLA1PROMCRC_AC | CLA1PROMCRC Master Access Control Register | EALLOW | Go |

**Table 3-280. PERIPH_AC_REGS Registers (continued)**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 110h | SPIA_AC | SPIA Master Access Control Register | EALLOW | Go |
| 112h | SPIB_AC | SPIB Master Access Control Register | EALLOW | Go |
| 130h | PMBUS_A_AC | PMBUSA Master Access Control Register | EALLOW | Go |
| 138h | LIN_A_AC | LINA Master Access Control Register | EALLOW | Go |
| 140h | DCANA_AC | DCANA Master Access Control Register | EALLOW | Go |
| 142h | DCANB_AC | DCANB Master Access Control Register | EALLOW | Go |
| 158h | FSIATX_AC | FSIA Master Access Control Register | EALLOW | Go |
| 15Ah | FSIARX_AC | FSIB Master Access Control Register | EALLOW | Go |
| 1AAh | HRPWM_A_AC | HRPWM Master Access Control Register | EALLOW | Go |
| 1FEh | PERIPH_AC_LOCK | Lock Register to stop Write access to peripheral Access register. | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 3-281 shows the codes that are used for access types in this section.

**Table 3-281. PERIPH_AC_REGS Access Type Codes**

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

#### 3.14.1.19.1 ADCA_AC Register (Offset = 0h) [reset = 3Fh]

ADCA_AC is shown in Figure 3-248 and described in Table 3-282.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-248. ADCA_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-282. ADCA_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.2 ADCB_AC Register (Offset = 2h) [reset = 3Fh]

ADCB_AC is shown in Figure 3-249 and described in Table 3-283.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-249. ADCB_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-283. ADCB_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.3 ADCC_AC Register (Offset = 4h) [reset = 3Fh]

ADCC_AC is shown in Figure 3-250 and described in Table 3-284.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-250. ADCC_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-284. ADCC_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.4 CMPSS1_AC Register (Offset = 10h) [reset = 3Fh]

CMPSS1_AC is shown in Figure 3-251 and described in Table 3-285.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-251. CMPSS1_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-285. CMPSS1_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.5 CMPSS2_AC Register (Offset = 12h) [reset = 3Fh]

CMPSS2_AC is shown in Figure 3-252 and described in Table 3-286.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-252. CMPSS2_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-286. CMPSS2_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.6 CMPSS3_AC Register (Offset = 14h) [reset = 3Fh]

CMPSS3_AC is shown in Figure 3-253 and described in Table 3-287.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-253. CMPSS3_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-287. CMPSS3_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.7 CMPSS4_AC Register (Offset = 16h) [reset = 3Fh]

CMPSS4_AC is shown in Figure 3-254 and described in Table 3-288.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-254. CMPSS4_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-288. CMPSS4_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.8 CMPSS5_AC Register (Offset = 18h) [reset = 3Fh]

CMPSS5_AC is shown in Figure 3-255 and described in Table 3-289.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-255. CMPSS5_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-289. CMPSS5_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.9 CMPSS6_AC Register (Offset = 1Ah) [reset = 3Fh]

CMPSS6_AC is shown in Figure 3-256 and described in Table 3-290.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-256. CMPSS6_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-290. CMPSS6_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.10 CMPSS7_AC Register (Offset = 1Ch) [reset = 3Fh]

CMPSS7_AC is shown in Figure 3-257 and described in Table 3-291.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-257. CMPSS7_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-291. CMPSS7_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.11 DACA_AC Register (Offset = 28h) [reset = 3Fh]

DACA_AC is shown in Figure 3-258 and described in Table 3-292.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-258. DACA_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-292. DACA_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.12 DACB_AC Register (Offset = 2Ah) [reset = 3Fh]

DACB_AC is shown in Figure 3-259 and described in Table 3-293.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-259. DACB_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-293. DACB_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.13 PGA1_AC Register (Offset = 38h) [reset = 3Fh]

PGA1_AC is shown in Figure 3-260 and described in Table 3-294.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-260. PGA1_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-294. PGA1_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.14 PGA2_AC Register (Offset = 3Ah) [reset = 3Fh]

PGA2_AC is shown in Figure 3-261 and described in Table 3-295.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-261. PGA2_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-295. PGA2_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.15 PGA3_AC Register (Offset = 3Ch) [reset = 3Fh]

PGA3_AC is shown in Figure 3-262 and described in Table 3-296.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-262. PGA3_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-296. PGA3_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.16 PGA4_AC Register (Offset = 3Eh) [reset = 3Fh]

PGA4_AC is shown in Figure 3-263 and described in Table 3-297.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-263. PGA4_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-297. PGA4_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.17 PGA5_AC Register (Offset = 40h) [reset = 3Fh]

PGA5_AC is shown in Figure 3-264 and described in Table 3-298.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-264. PGA5_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-298. PGA5_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.18 PGA6_AC Register (Offset = 42h) [reset = 3Fh]

PGA6_AC is shown in Figure 3-265 and described in Table 3-299.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-265. PGA6_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-299. PGA6_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.19 PGA7_AC Register (Offset = 44h) [reset = 3Fh]

PGA7_AC is shown in Figure 3-266 and described in Table 3-300.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-266. PGA7_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-300. PGA7_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.20 EPWM1_AC Register (Offset = 48h) [reset = 3Fh]

EPWM1_AC is shown in Figure 3-267 and described in Table 3-301.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-267. EPWM1_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-301. EPWM1_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.21 EPWM2_AC Register (Offset = 4Ah) [reset = 3Fh]

EPWM2_AC is shown in Figure 3-268 and described in Table 3-302.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-268. EPWM2_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-302. EPWM2_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.22 EPWM3_AC Register (Offset = 4Ch) [reset = 3Fh]

EPWM3_AC is shown in Figure 3-269 and described in Table 3-303.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-269. EPWM3_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-303. EPWM3_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.23 EPWM4_AC Register (Offset = 4Eh) [reset = 3Fh]

EPWM4_AC is shown in Figure 3-270 and described in Table 3-304.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-270. EPWM4_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-304. EPWM4_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.24 EPWM5_AC Register (Offset = 50h) [reset = 3Fh]

EPWM5_AC is shown in Figure 3-271 and described in Table 3-305.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-271. EPWM5_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-305. EPWM5_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.25 EPWM6_AC Register (Offset = 52h) [reset = 3Fh]

EPWM6_AC is shown in Figure 3-272 and described in Table 3-306.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-272. EPWM6_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-306. EPWM6_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.26 EPWM7_AC Register (Offset = 54h) [reset = 3Fh]

EPWM7_AC is shown in Figure 3-273 and described in Table 3-307.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-273. EPWM7_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-307. EPWM7_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.27 EPWM8_AC Register (Offset = 56h) [reset = 3Fh]

EPWM8_AC is shown in Figure 3-274 and described in Table 3-308.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-274. EPWM8_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-308. EPWM8_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.28 EQEP1_AC Register (Offset = 70h) [reset = 3Fh]

EQEP1_AC is shown in Figure 3-275 and described in Table 3-309.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-275. EQEP1_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-309. EQEP1_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.29 EQEP2_AC Register (Offset = 72h) [reset = 3Fh]

EQEP2_AC is shown in Figure 3-276 and described in Table 3-310.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-276. EQEP2_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-310. EQEP2_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.30 ECAP1_AC Register (Offset = 80h) [reset = 3Fh]

ECAP1_AC is shown in Figure 3-277 and described in Table 3-311.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-277. ECAP1_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-311. ECAP1_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.31 ECAP2_AC Register (Offset = 82h) [reset = 3Fh]

ECAP2_AC is shown in Figure 3-278 and described in Table 3-312.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-278. ECAP2_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-312. ECAP2_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.32 ECAP3_AC Register (Offset = 84h) [reset = 3Fh]

ECAP3_AC is shown in Figure 3-279 and described in Table 3-313.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-279. ECAP3_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-313. ECAP3_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.33 ECAP4_AC Register (Offset = 86h) [reset = 3Fh]

ECAP4_AC is shown in Figure 3-280 and described in Table 3-314.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-280. ECAP4_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-314. ECAP4_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.34 ECAP5_AC Register (Offset = 88h) [reset = 3Fh]

ECAP5_AC is shown in Figure 3-281 and described in Table 3-315.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-281. ECAP5_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-315. ECAP5_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.35 ECAP6_AC Register (Offset = 8Ah) [reset = 3Fh]

ECAP6_AC is shown in Figure 3-282 and described in Table 3-316.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-282. ECAP6_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-316. ECAP6_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.36 ECAP7_AC Register (Offset = 8Ch) [reset = 3Fh]

ECAP7_AC is shown in Figure 3-283 and described in Table 3-317.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-283. ECAP7_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-317. ECAP7_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.37 SDFM1_AC Register (Offset = A8h) [reset = 3Fh]

SDFM1_AC is shown in Figure 3-284 and described in Table 3-318.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-284. SDFM1_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-318. SDFM1_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.38 CLB1_AC Register (Offset = B0h) [reset = 3Fh]

CLB1_AC is shown in Figure 3-285 and described in Table 3-319.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-285. CLB1_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | RESERVED | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R-0h | | R/W-3h | | R/W-3h | |

**Table 3-319. CLB1_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | RESERVED | R | 0h | Reserved |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.39 CLB2_AC Register (Offset = B2h) [reset = 3Fh]

CLB2_AC is shown in Figure 3-286 and described in Table 3-320.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-286. CLB2_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | RESERVED | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R-0h | | R/W-3h | | R/W-3h | |

**Table 3-320. CLB2_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | RESERVED | R | 0h | Reserved |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.40 CLB3_AC Register (Offset = B4h) [reset = 3Fh]

CLB3_AC is shown in Figure 3-287 and described in Table 3-321.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-287. CLB3_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | RESERVED | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R-0h | | R/W-3h | | R/W-3h | |

**Table 3-321. CLB3_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | RESERVED | R | 0h | Reserved |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.41 CLB4_AC Register (Offset = B6h) [reset = 3Fh]

CLB4_AC is shown in Figure 3-288 and described in Table 3-322.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-288. CLB4_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | RESERVED | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R-0h | | R/W-3h | | R/W-3h | |

**Table 3-322. CLB4_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | RESERVED | R | 0h | Reserved |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.42 CLA1PROMCRC_AC Register (Offset = C0h) [reset = 3Fh]

CLA1PROMCRC_AC is shown in Figure 3-289 and described in Table 3-323.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-289. CLA1PROMCRC_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | RESERVED | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R-0h | | R/W-3h | | R/W-3h | |

**Table 3-323. CLA1PROMCRC_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | RESERVED | R | 0h | Reserved |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.43 SPIA_AC Register (Offset = 110h) [reset = 3Fh]

SPIA_AC is shown in Figure 3-290 and described in Table 3-324.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-290. SPIA_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-324. SPIA_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.44 SPIB_AC Register (Offset = 112h) [reset = 3Fh]

SPIB_AC is shown in Figure 3-291 and described in Table 3-325.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-291. SPIB_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-325. SPIB_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.45 PMBUS_A_AC Register (Offset = 130h) [reset = 3Fh]

PMBUS_A_AC is shown in Figure 3-292 and described in Table 3-326.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-292. PMBUS_A_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-326. PMBUS_A_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.46 LIN_A_AC Register (Offset = 138h) [reset = 3Fh]

LIN_A_AC is shown in Figure 3-293 and described in Table 3-327.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-293. LIN_A_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-327. LIN_A_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.47 DCANA_AC Register (Offset = 140h) [reset = 3Fh]

DCANA_AC is shown in Figure 3-294 and described in Table 3-328.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-294. DCANA_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | RESERVED | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R-0h | | R/W-3h | |

**Table 3-328. DCANA_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | RESERVED | R | 0h | Reserved |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.48 DCANB_AC Register (Offset = 142h) [reset = 3Fh]

DCANB_AC is shown in Figure 3-295 and described in Table 3-329.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-295. DCANB_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | RESERVED | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R-0h | | R/W-3h | |

**Table 3-329. DCANB_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | RESERVED | R | 0h | Reserved |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.49 FSIATX_AC Register (Offset = 158h) [reset = 3Fh]

FSIATX_AC is shown in Figure 3-296 and described in Table 3-330.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-296. FSIATX_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-330. FSIATX_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.50 FSIARX_AC Register (Offset = 15Ah) [reset = 3Fh]

FSIARX_AC is shown in Figure 3-297 and described in Table 3-331.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-297. FSIARX_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-331. FSIARX_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.51 HRPWM_A_AC Register (Offset = 1AAh) [reset = 3Fh]

HRPWM_A_AC is shown in Figure 3-298 and described in Table 3-332.

Return to Summary Table.

Based on control settings allows Full, Protected Read, No Access to perpheral from corresponding connected master.

**Figure 3-298. HRPWM_A_AC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | DMA1_ACC | | CLA1_ACC | | CPU1_ACC | |
| R=0-0h | | R/W-3h | | R/W-3h | | R/W-3h | |

**Table 3-332. HRPWM_A_AC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | DMA1_ACC | R/W | 3h | Defines Access control definition for the DMA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 3-2 | CLA1_ACC | R/W | 3h | Defines Access control definition for the CLA1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |
| 1-0 | CPU1_ACC | R/W | 3h | Defines Access control definition for the CPU1 as:<br>11: Full Access for both read and Write<br>10: Protected RD Access such that FIFOs, Clear on read registers are not changed + No Write Access<br>01: Reserved<br>00: No Read/Write Access to peripheral<br>Reset type: XRSn |

### 3.14.1.19.52 PERIPH_AC_LOCK Register (Offset = 1FEh) [reset = 0h]

PERIPH_AC_LOCK is shown in Figure 3-299 and described in Table 3-333.

Return to Summary Table.

Based on status bit control the Access registers are either RD/WR or RD only.

**Figure 3-299. PERIPH_AC_LOCK Register**

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

**Table 3-333. PERIPH_AC_LOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-1 | RESERVED | R=0 | 0h | Reserved |
| 0 | LOCK_AC_WR | R/WSOnce | 0h | Defines Access control definition for the CPU1 as:<br>1: Access Control registers are Read Only<br>0: Read/Write Access allowed to Access Control registers.<br>Writing '1' sets the bit, writing '0' has no effect.<br>Reset type: SYSRSn |

### 3.14.1.20 PIE_CTRL_REGS Registers

Table 3-334 lists the memory-mapped registers for the PIE_CTRL_REGS. All register offset addresses not listed in Table 3-334 should be considered as reserved locations and the register contents should not be modified.

**Table 3-334. 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 3-335 shows the codes that are used for access types in this section.

**Table 3-335. PIE_CTRL_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| W=1 | W | Write |
| **Reset or Default Value** | | |
| *-n* | | Value after reset or the default value |
| **Register Array Variables** | | |

**Table 3-335. PIE_CTRL_REGS Access Type Codes (continued)**

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

#### 3.14.1.20.1 PIECTRL Register (Offset = 0h) [reset = 0h]

PIECTRL is shown in Figure 3-300 and described in Table 3-336.

Return to Summary Table.

ePIE Control Register

**Figure 3-300. 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 3-336. 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 |

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

PIEACK is shown in Figure 3-301 and described in Table 3-337.

Return to 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 3-301. PIEACK Register**

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

**Table 3-337. PIEACK Register Field Descriptions**

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

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

PIEIER1 is shown in Figure 3-302 and described in Table 3-338.

Return to 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 3-302. 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 3-338. 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 |

**Table 3-338. PIEIER1 Register Field Descriptions (continued)**

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

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

PIEIFR1 is shown in Figure 3-303 and described in Table 3-339.

Return to 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 3-303. 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 3-339. 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 |

**Table 3-339. 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 |

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

PIEIER2 is shown in Figure 3-304 and described in Table 3-340.

Return to 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 3-304. 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 3-340. 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 |

**Table 3-340. PIEIER2 Register Field Descriptions (continued)**

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

#### 3.14.1.20.6 PIEIFR2 Register (Offset = 5h) [reset = 0h]

PIEIFR2 is shown in Figure 3-305 and described in Table 3-341.

Return to 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 3-305. 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 3-341. 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 |

**Table 3-341. 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 |

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

PIEIER3 is shown in Figure 3-306 and described in Table 3-342.

Return to 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 3-306. 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 3-342. 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 |

**Table 3-342. PIEIER3 Register Field Descriptions (continued)**

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

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

PIEIFR3 is shown in Figure 3-307 and described in Table 3-343.

Return to 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 3-307. 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 3-343. 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 |

**Table 3-343. 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 |

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

PIEIER4 is shown in Figure 3-308 and described in Table 3-344.

Return to 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 3-308. 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 3-344. 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 |

**Table 3-344. PIEIER4 Register Field Descriptions (continued)**

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

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

PIEIFR4 is shown in Figure 3-309 and described in Table 3-345.

Return to 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 3-309. 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 3-345. 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 |

**Table 3-345. 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 |

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

PIEIER5 is shown in Figure 3-310 and described in Table 3-346.

Return to 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 3-310. 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 3-346. 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 |

**Table 3-346. PIEIER5 Register Field Descriptions (continued)**

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

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

PIEIFR5 is shown in Figure 3-311 and described in Table 3-347.

Return to 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 3-311. 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 3-347. 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 |

**Table 3-347. 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 |

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

PIEIER6 is shown in Figure 3-312 and described in Table 3-348.

Return to 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 3-312. 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 3-348. 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 |

**Table 3-348. PIEIER6 Register Field Descriptions (continued)**

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

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

PIEIFR6 is shown in Figure 3-313 and described in Table 3-349.

Return to 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 3-313. 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 3-349. 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 |

**Table 3-349. 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 |

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

PIEIER7 is shown in Figure 3-314 and described in Table 3-350.

Return to 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 3-314. 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 3-350. 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 |

**Table 3-350. PIEIER7 Register Field Descriptions (continued)**

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

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

PIEIFR7 is shown in Figure 3-315 and described in Table 3-351.

Return to 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 3-315. 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 3-351. 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 |

**Table 3-351. 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 |

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

PIEIER8 is shown in Figure 3-316 and described in Table 3-352.

Return to 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 3-316. 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 3-352. 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 |

**Table 3-352. PIEIER8 Register Field Descriptions (continued)**

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

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

PIEIFR8 is shown in Figure 3-317 and described in Table 3-353.

Return to 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 3-317. 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 3-353. 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 |

**Table 3-353. 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 |

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

PIEIER9 is shown in Figure 3-318 and described in Table 3-354.

Return to 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 3-318. 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 3-354. 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 |

**Table 3-354. PIEIER9 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | INTx2 | R/W | 0h | Enable for Interrupt 9.2<br>Reset type: SYSRSn |
| 0 | INTx1 | R/W | 0h | Enable for Interrupt 9.1<br>Reset type: SYSRSn |

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

PIEIFR9 is shown in Figure 3-319 and described in Table 3-355.

Return to 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 3-319. 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 3-355. 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 |

**Table 3-355. 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 |

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

PIEIER10 is shown in Figure 3-320 and described in Table 3-356.

Return to 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 3-320. 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 3-356. 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 |

**Table 3-356. PIEIER10 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | INTx2 | R/W | 0h | Enable for Interrupt 10.2<br>Reset type: SYSRSn |
| 0 | INTx1 | R/W | 0h | Enable for Interrupt 10.1<br>Reset type: SYSRSn |

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

PIEIFR10 is shown in Figure 3-321 and described in Table 3-357.

Return to 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 3-321. 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 3-357. 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 |

**Table 3-357. 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 |

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

PIEIER11 is shown in Figure 3-322 and described in Table 3-358.

Return to 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 3-322. 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 3-358. 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 |

**Table 3-358. PIEIER11 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | INTx2 | R/W | 0h | Enable for Interrupt 11.2<br>Reset type: SYSRSn |
| 0 | INTx1 | R/W | 0h | Enable for Interrupt 11.1<br>Reset type: SYSRSn |

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

PIEIFR11 is shown in Figure 3-323 and described in Table 3-359.

Return to 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 3-323. 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 3-359. 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 |

**Table 3-359. 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 |

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

PIEIER12 is shown in Figure 3-324 and described in Table 3-360.

Return to 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 3-324. 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 3-360. 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 |

**Table 3-360. PIEIER12 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | INTx2 | R/W | 0h | Enable for Interrupt 12.2<br>Reset type: SYSRSn |
| 0 | INTx1 | R/W | 0h | Enable for Interrupt 12.1<br>Reset type: SYSRSn |

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

PIEIFR12 is shown in Figure 3-325 and described in Table 3-361.

Return to 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 3-325. 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 3-361. 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 |

**Table 3-361. 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 |

### 3.14.1.21 ROM_PREFETCH_REGS Registers

Table 3-362 lists the memory-mapped registers for the ROM_PREFETCH_REGS. All register offset addresses not listed in Table 3-362 should be considered as reserved locations and the register contents should not be modified.

**Table 3-362. 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 3-363 shows the codes that are used for access types in this section.

**Table 3-363. 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. |

#### 3.14.1.21.1 ROMPREFETCH Register (Offset = 0h) [reset = 0h]

ROMPREFETCH is shown in Figure 3-326 and described in Table 3-364.

Return to Summary Table.

ROM Prefetch Configuration Register

**Figure 3-326. 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 3-364. 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 |

### 3.14.1.22 ROM_WAIT_STATE_REGS Registers

Table 3-365 lists the memory-mapped registers for the ROM_WAIT_STATE_REGS. 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. 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 3-366 shows the codes that are used for access types in this section.

**Table 3-366. 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. |

#### 3.14.1.22.1 ROMWAITSTATE Register (Offset = 0h) [reset = 0h]

ROMWAITSTATE is shown in Figure 3-327 and described in Table 3-367.

Return to Summary Table.

ROM Wait State Configuration Register

**Figure 3-327. 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 3-367. ROMWAITSTATE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-1 | RESERVED | R | 0h | Reserved |
| 0 | WSDISABLE | R/W | 0h | 0: ROM Wait State is enabled. CPU accesses to ROM are are 1-wait.<br>1: ROM Wait State is disabled. CPU accesses to ROM are 0-wait.<br>Reset type: SYSRSn |

### 3.14.1.23 UID_REGS Registers

Table 3-368 lists the memory-mapped registers for the UID_REGS. All register offset addresses not listed in Table 3-368 should be considered as reserved locations and the register contents should not be modified.

**Table 3-368. 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 3-369 shows the codes that are used for access types in this section.

**Table 3-369. 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 |
| **Register Array Variables** | | |
| i,j,k,l,m,n | | When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula. |
| y | | When this variable is used in a register name, an offset, or an address it refers to the value of a register array. |

#### 3.14.1.23.1 UID_PSRAND0 Register (Offset = 0h) [reset = X]

UID_PSRAND0 is shown in Figure 3-328 and described in Table 3-370.

Return to Summary Table.

UID Psuedo-random 192 bit number

**Figure 3-328. UID_PSRAND0 Register**

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

**Table 3-370. UID_PSRAND0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | RandomID | R | X | Psuedorandom portion of the UID<br>Reset type: N/A |

#### 3.14.1.23.2 UID_PSRAND1 Register (Offset = 2h) [reset = X]

UID_PSRAND1 is shown in Figure 3-329 and described in Table 3-371.

Return to Summary Table.

UID Psuedo-random 192 bit number

**Figure 3-329. UID_PSRAND1 Register**

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

**Table 3-371. UID_PSRAND1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | RandomID | R | X | Psuedorandom portion of the UID<br>Reset type: N/A |

#### 3.14.1.23.3 UID_PSRAND2 Register (Offset = 4h) [reset = X]

UID_PSRAND2 is shown in Figure 3-330 and described in Table 3-372.

Return to Summary Table.

UID Psuedo-random 192 bit number

**Figure 3-330. 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-X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-372. UID_PSRAND2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | RandomID | R | X | Psuedorandom portion of the UID<br>Reset type: N/A |

#### 3.14.1.23.4 UID_PSRAND3 Register (Offset = 6h) [reset = X]

UID_PSRAND3 is shown in Figure 3-331 and described in Table 3-373.

Return to Summary Table.

UID Psuedo-random 192 bit number

**Figure 3-331. UID_PSRAND3 Register**

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

**Table 3-373. UID_PSRAND3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | RandomID | R | X | Psuedorandom portion of the UID<br>Reset type: N/A |

#### 3.14.1.23.5 UID_PSRAND4 Register (Offset = 8h) [reset = X]

UID_PSRAND4 is shown in Figure 3-332 and described in Table 3-374.

Return to Summary Table.

UID Psuedo-random 192 bit number

**Figure 3-332. 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-X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 3-374. UID_PSRAND4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | RandomID | R | X | Psuedorandom portion of the UID<br>Reset type: N/A |

#### 3.14.1.23.6 UID_PSRAND5 Register (Offset = Ah) [reset = X]

UID_PSRAND5 is shown in Figure 3-333 and described in Table 3-375.

Return to Summary Table.

UID Psuedo-random 192 bit number

**Figure 3-333. UID_PSRAND5 Register**

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

**Table 3-375. UID_PSRAND5 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | RandomID | R | X | Psuedorandom portion of the UID<br>Reset type: N/A |

#### 3.14.1.23.7 UID_UNIQUE Register (Offset = Ch) [reset = X]

UID_UNIQUE is shown in Figure 3-334 and described in Table 3-376.

Return to Summary Table.

UID Unique 32 bit number

**Figure 3-334. UID_UNIQUE Register**

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

**Table 3-376. UID_UNIQUE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | UniqueID | R | X | Unique portion of the UID. This identifier will be unique across all devices with the same PARTIDH.<br>Reset type: N/A |

### 3.14.1.23.8 UID_CHECKSUM Register (Offset = Eh) [reset = X]

UID_CHECKSUM is shown in Figure 3-335 and described in Table 3-377.

Return to Summary Table.

Fletcher checksum of UID_PSRAND and UID_UNIQUE registers

**Figure 3-335. UID_CHECKSUM Register**

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

**Table 3-377. UID_CHECKSUM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | Checksum | R | X | Fletcher checksum of UID_PSRANDx and UID_UINIQUE<br>Reset type: N/A |

### 3.14.1.24 WD_REGS Registers

Table 3-378 lists the memory-mapped registers for the WD_REGS. All register offset addresses not listed in Table 3-378 should be considered as reserved locations and the register contents should not be modified.

**Table 3-378. WD_REGS Registers**

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

Complex bit access types are encoded to fit into small table cells. Table 3-379 shows the codes that are used for access types in this section.

**Table 3-379. WD_REGS Access Type Codes**

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

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

SCSR is shown in Figure 3-336 and described in Table 3-380.

Return to Summary Table.

System Control & Status Register

**Figure 3-336. SCSR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | WDINTS | WDENINT | WDOVERRIDE |
| R=0-0h | | | | | R-1h | R/W-0h | R/W=1-1h |

**Table 3-380. SCSR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-3 | RESERVED | R=0 | 0h | Reserved |
| 2 | WDINTS | R | 1h | Watchdog Interrupt Status<br>This bit indicates the state of the active-low watchdog interrupt signal (synchronized to SYSCLK). If the watchdog interrupt is used to wake the system from a low-power mode, then that mode should only be entered while this bit is high. Likewise, this bit must go high before the watchdog can be safely disabled and re-enabled.<br>Reset type: SYSRSn<br>0h (R/W) = The watchdog interrupt signal is active.<br>1h (R/W) = The watchdog interrupt signal is inactive. |
| 1 | WDENINT | R/W | 0h | Watchdog Interrupt Enable/Reset Disable<br>This bit determines whether the watchdog triggers an interrupt (WAKE/WDOG) or a reset (WDRS) when the counter expires.<br>Reset type: SYSRSn<br>0h (R/W) = Counter expiration triggers a reset. This is the default state on power-up and after any system reset.<br>1h (R/W) = Counter expiration triggers an interrupt. |
| 0 | WDOVERRIDE | R/W=1 | 1h | Watchdog Enable Lock<br>Writing a 1 to this bit clears it and locks the WDDIS bit in the WDCR register. The bit will remain in this state until the next system reset. Reads of this bit return its current value. Writing a 0 to this bit has no effect.<br>Reset type: SYSRSn |

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

WDCNTR is shown in Figure 3-337 and described in Table 3-381.

Return to Summary Table.

Watchdog Counter Register

**Figure 3-337. 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 3-381. WDCNTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R=0 | 0h | Reserved |
| 7-0 | WDCNTR | R | 0h | Watchdog Counter<br>These bits contain the current value of the watchdog counter. This counter increments with each WDCLK (INTOSC1) cycle. If the counter overflows, either an interrupt or a reset is generated based on the value of the WDINTEN bit in the SCSR register. If the correct value is written to the WDKEY register, this counter is reset to zero.<br>Reset type: IORSn |

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

WDKEY is shown in Figure 3-338 and described in Table 3-382.

Return to Summary Table.

Watchdog Reset Key Register

**Figure 3-338. 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 3-382. WDKEY Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R=0 | 0h | Reserved |
| 7-0 | WDKEY | R/W | 0h | Watchdog Counter Reset<br>Writing 0x55 followed by 0xAA will cause the watchdog counter to reset to zero, preventing an overflow. Writing other values has no effect. Reads of this register return the value of the WDCR register.<br>Reset type: IORSn |

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

WDCR is shown in Figure 3-339 and described in Table 3-383.

Return to Summary Table.

Watchdog Control Register

**Figure 3-339. WDCR Register**

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

**Table 3-383. WDCR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R=0 | 0h | Reserved |
| 11-8 | WDPRECLKDIV | R/W | 0h | Watchdog Clock Pre-divider<br>These bits determine the watchdog clock pre-divider, which is the first of the two dividers between INTOSC1 and the watchdog counter clock (WDCLK). The frequency of WDCLK is given by the formulas:<br>PREDIVCLK = INTOSC1 / Pre-divider<br>WDCLK = PREDIVCLK / Prescaler<br>The watchdog reset or interrupt pulse is 512 INTOSC1 cycles long, so the counter period must be longer. To guarantee this, the product of the prescaler and pre-divider must be greater than or equal to four. The default pre-divider value is 512.<br>Reset type: IORSn<br>0h (R/W) = PREDIVCLK = INTOSC1 / 512<br>1h (R/W) = PREDIVCLK = INTOSC1 / 1024<br>2h (R/W) = PREDIVCLK = INTOSC1 / 2048<br>3h (R/W) = PREDIVCLK = INTOSC1 / 4096<br>4h (R/W) = Reserved<br>5h (R/W) = Reserved<br>6h (R/W) = Reserved<br>7h (R/W) = Reserved<br>8h (R/W) = PREDIVCLK = INTOSC1 / 2<br>9h (R/W) = PREDIVCLK = INTOSC1 / 4<br>Ah (R/W) = PREDIVCLK = INTOSC1 / 8<br>Bh (R/W) = PREDIVCLK = INTOSC1 / 16<br>Ch (R/W) = PREDIVCLK = INTOSC1 / 32<br>Dh (R/W) = PREDIVCLK = INTOSC1 / 64<br>Eh (R/W) = PREDIVCLK = INTOSC1 / 128<br>Fh (R/W) = PREDIVCLK = INTOSC1 / 256 |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | WDDIS | R/W | 0h | Watchdog Disable<br>Setting this bit disables the watchdog module. Clearing this bit enables the watchdog module. This bit can be locked by the WDOVERRIDE bit in the SCSR register. The watchdog is enabled on reset.<br>Reset type: IORSn |

**Table 3-383. WDCR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5-3 | WDCHK | R=0/W | 0h | Watchdog Check Bits<br><br>During any write to this register, these bits must be written with the value 101 (binary). Writing any other value will immediately trigger the watchdog reset or interrupt.<br>Reset type: IORSn |
| 2-0 | WDPS | R/W | 0h | Watchdog Clock Prescaler<br><br>These bits determine the watchdog clock prescaler, which is the second of the two dividers between INTOSC1 and the watchdog counter clock (WDCLK). The frequency of WDCLK is given by the formulas:<br><br>PREDIVCLK = INTOSC1 / Pre-divider<br>WDCLK = PREDIVCLK / Prescaler<br><br>The watchdog reset or interrupt pulse is 512 INTOSC1 cycles long, so the counter period must be longer. To guarantee this, the product of the prescaler and pre-divider must be greater than or equal to four. The default prescaler value is 1.<br>Reset type: IORSn<br>0h (R/W) = WDCLK = PREDIVCLK / 1<br>1h (R/W) = WDCLK = PREDIVCLK / 1<br>2h (R/W) = WDCLK = PREDIVCLK / 2<br>3h (R/W) = WDCLK = PREDIVCLK / 4<br>4h (R/W) = WDCLK = PREDIVCLK / 8<br>5h (R/W) = WDCLK = PREDIVCLK / 16<br>6h (R/W) = WDCLK = PREDIVCLK / 32<br>7h (R/W) = WDCLK = PREDIVCLK / 64 |

### 3.14.1.24.5 WDWCR Register (Offset = 2Ah) [reset = 0h]

WDWCR is shown in Figure 3-340 and described in Table 3-384.

Return to Summary Table.

Watchdog Windowed Control Register

**Figure 3-340. WDWCR Register**

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

**Table 3-384. WDWCR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-9 | RESERVED | R=0 | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 7-0 | MIN | R/W | 0h | Watchdog Window Threshold<br>These bits specify the lower limit of the watchdog counter reset window. If the counter is reset via the WDKEY register before the counter value reaches the value in this register, the watchdog immediately triggers a reset or interrupt.<br>Reset type: IORSn |

### 3.14.1.25 XINT_REGS Registers

Table 3-385 lists the memory-mapped registers for the XINT_REGS. All register offset addresses not listed in Table 3-385 should be considered as reserved locations and the register contents should not be modified.

**Table 3-385. 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 3-386 shows the codes that are used for access types in this section.

**Table 3-386. XINT_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | 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. |

#### 3.14.1.25.1 XINT1CR Register (Offset = 0h) [reset = 0h]

XINT1CR is shown in Figure 3-341 and described in Table 3-387.

Return to Summary Table.

XINT1 configuration register

**Figure 3-341. 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 3-387. 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 |

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

XINT2CR is shown in Figure 3-342 and described in Table 3-388.

Return to Summary Table.

XINT2 configuration register

**Figure 3-342. 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 3-388. 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 |

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

XINT3CR is shown in Figure 3-343 and described in Table 3-389.

Return to Summary Table.

XINT3 configuration register

**Figure 3-343. 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 3-389. 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 |

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

XINT4CR is shown in Figure 3-344 and described in Table 3-390.

Return to Summary Table.

XINT4 configuration register

**Figure 3-344. 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 3-390. 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 |

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

XINT5CR is shown in Figure 3-345 and described in Table 3-391.

Return to Summary Table.

XINT5 configuration register

**Figure 3-345. 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 3-391. 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 |

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

XINT1CTR is shown in Figure 3-346 and described in Table 3-392.

Return to Summary Table.

XINT1 counter register

**Figure 3-346. XINT1CTR Register**

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

**Table 3-392. 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 |

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

XINT2CTR is shown in Figure 3-347 and described in Table 3-393.

Return to Summary Table.

XINT2 counter register

**Figure 3-347. XINT2CTR Register**

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

**Table 3-393. 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 |

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

XINT3CTR is shown in Figure 3-348 and described in Table 3-394.

Return to Summary Table.

XINT3 counter register

**Figure 3-348. XINT3CTR Register**

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

**Table 3-394. 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 |

# ROM Code and Peripheral Booting

This chapter explains the boot procedure, the available boot modes, and the various details of the ROM code including memory maps, initializations, reset handling, and status information.

**Topic** **Page**

## 4.1 Introduction

The purpose of this chapter is to explain the boot ROM code functionality including the boot procedure when executed, the functions and features of the boot ROM code, and to detail the ROM memory map contents. On every reset, the device executes a boot sequence in the ROM depending on the reset type and boot configuration. This sequence will initialize the device to run application code. The boot ROM also contains peripheral bootloaders which can be used to load an application into RAM.

**Table 4-1. ROM Memory**

| ROM | SIZE |
|---|---|
| Unsecure boot ROM | 128 KB |
| Secure ROM | 64 KB |
| CLA Data ROM | 8 KB |

## 4.2 Device Boot Philosophy

The boot philosophy describes the general boot ROM procedure each time the CPU core is reset. During booting, the boot ROM code updates a boot status location in RAM that details the actions taken during this process.

Refer to *Boot Status Information* for more details on the boot status information.

**Table 4-2. Boot ROM Philosophy**

| Step | Action |
|---|---|
| 1 | After reset, the FUSE error register is checked for any errors and are handled accordingly. |
| 2 | Clock and Flash configuration. |
| 3 | Device configuration registers are programmed from OTP. |
| 4 | On power-on reset (POR), all CPU RAMs are initialized. |
| 5 | Any pending NMI is handled by the code. |
| 6 | DCSM and OTPJTAGLOCK sequences are executed. (Refer to *DCSM Usage* for details on how boot ROM interprets the OTP data after initialization) |
| 7 | Device calibration is performed, trimming the specified peripherals with set OTP values. |
| 8 | The boot mode GPIO pins are polled to determine whether to boot from SRAM, flash, or peripherals. |
| 9 | Based on the boot mode and options, the appropriate boot sequence is executed.<br>Refer to *Device Boot Flow Diagram* for a flow diagram of the device boot sequence and the emulation and standalone boot modes. |

## 4.3 Device Boot Modes

This section explains the boot modes supported on this device. The boot ROM uses the boot control GPIO pins to determine the boot mode configuration. The device can be configured to boot to RAM, boot to flash, execute a bootloader, or hold in a wait mode.

Table 4-3 shows the default boot mode options. Users have the option to customize the boot modes supported as well as the boot mode select pins.

**Table 4-3. Device Default Boot Modes**

| Boot Mode | GPIO24 (Default boot mode select pin 1) | GPIO32 (Default boot mode select pin 0) |
|---|---|---|
| Parallel IO | 0 | 0 |
| SCI / Wait boot | 0 | 1 |
| CAN | 1 | 0 |
| Flash | 1 | 1 |

**Table 4-4. All Available Boot Modes**

| Boot Mode Number | Boot Mode |
|---|---|
| 0 | Parallel IO |
| 1 | SCI / Wait boot |
| 2 | CAN |
| 3 | Flash |
| 4 | Wait |
| 5 | RAM |
| 6 | SPI Master |
| 7 | I2C Master |
| 8 | PLC(1) |

(1) The PLC Boot will work only on PLC-enabled part numbers. Please check the device data sheet for the part numbers.

**NOTE:** All the peripheral boot modes supported use the first instance of the peripheral module (SCIA, SPIA, I2CA, CANA, and so forth). Whenever these boot modes are referred to in this chapter, such as SCI boot, it is actually referring to the first module instance, meaning SCI boot on the SCIA port. The same applies to the other peripheral boots.

### 4.3.1 Configuring Alternate Boot Mode Pins

This section explains how the boot mode select pins can be customized by the user, by programming the BOOTPIN_CONFIG location in user-configurable DCSM OTP. The location in user DCSM OTP is Z1-OTP-BOOTPIN-CONFIG. When debugging, EMU-BOOTPIN-CONFIG is the emulation equivalent of Z1-OTP-BOOTPIN-CONFIG, and can be programmed to experiment with different boot modes without writing to OTP. The device can be programmed to use 0, 1, 2, or 3 boot mode select pins as needed.

**Table 4-5. BOOTPIN_CONFIG Bit Fields**

| Bit | Name | Description |
|---|---|---|
| 31-24 | Key | Write 0x5A to these 8-bits to tell the boot ROM code that the bits in this register are valid |
| 23-16 | Boot Mode Select Pin 2 (BMSP2) | Refer to BMSP0 description except for BMSP2 |
| 15-8 | Boot Mode Select Pin 1 (BMSP1) | Refer to BMSP0 description except for BMSP1 |
| 7-0 | Boot Mode Select Pin 0 (BMSP0) | Set to the GPIO pin to be used during boot (up to 255).<br>0x0 = GPIO0; 0x01 = GPIO1 and so on<br>0xFF is invalid and selects the factory default chosen BMSP0, if all other BMSPs are also set to 0xFF.<br>If any other BMSPs are not set to 0xFF, then setting a BMSP to 0xFF will disable that particular BMSP. |

**NOTE:** The following GPIOs **cannot** be used as a BMSP. If selected for a particular BMSP, the boot ROM automatically selects the factory default GPIO (the factory default for BMSP2 is 0xFF, which disables the BMSP).

- GPIO 20 to 23
- GPIO 36
- GPIO 38
- GPIO 60 to 223

**Table 4-6. Standalone Boot Mode Select Pin Decoding**

| BOOTPIN_CONFIG Key | BMSP0 | BMSP1 | BMSP2 | Realized Boot Mode |
|---|---|---|---|---|
| != 0x5A | Don't Care | Don't Care | Don't Care | Boot as defined by the factory default BMSPs (GPIO24, GPIO32) |
| = 0x5A | 0xFF | 0xFF | 0xFF | Boot as defined in the boot table for boot mode 0 (All BMSPs disabled) |
| | Valid GPIO | 0xFF | 0xFF | Boot as defined by the value of BMSP0 (BMSP1 and BMSP2 disabled) |
| | 0xFF | Valid GPIO | 0xFF | Boot as defined by the value of BMSP1 (BMSP0 and BMSP2 disabled) |
| | 0xFF | 0xFF | Valid GPIO | Boot as defined by the value of BMSP2 (BMSP0 and BMSP1 disabled) |
| | Valid GPIO | Valid GPIO | 0xFF | Boot as defined by the values of BMSP0 and BMSP1 (BMSP2 disabled) |
| | Valid GPIO | 0xFF | Valid GPIO | Boot as defined by the values of BMSP0 and BMSP2 (BMSP1 disabled) |
| | 0xFF | Valid GPIO | Valid GPIO | Boot as defined by the values of BMSP1 and BMSP2 (BMSP0 disabled) |
| | Valid GPIO | Valid GPIO | Valid GPIO | Boot as defined by the values of BMSP0, BMSP1, and BMSP2 |

### 4.3.2 Configuring Alternate Boot Mode Options

This section explains how to configure the boot definition table, BOOTDEF, for the device and the associated boot options. The 64-bit location is located in user-configurable DCSM OTP in the Z1-OTP-BOOTDEF-LOW and Z1-OTP-BOOTDEF-HIGH locations. When debugging, EMU-BOOTDEF-LOW and EMU-BOOTDEF-HIGH are the emulation equivalents of Z1-OTP-BOOTDEF-LOW and Z1-OTP-BOOTDEF-HIGH, and can be programmed to experiment with different boot mode options without writing to OTP. The range of customization to the boot definition table depends on how many boot mode select pins are being used. Refer to *Boot Mode Example Use Cases* for examples on how to use the BOOTPIN_CONFIG and BOOTDEF values.

**Table 4-7. BOOTDEF Bit Fields**

| BOOTDEF Name | Byte Position | Name | Description |
|---|---|---|---|
| BOOT_DEF0 | 7-0 | BOOT_DEF0 Mode/Options | Set the boot mode and boot mode options. This can include changing the GPIOs for a particular boot peripheral or specifying a different flash entry point. Any unsupported boot mode will cause the device to reset.<br>Refer to *GPIO Assignments* for valid BOOTDEF values. |
| BOOT_DEF1 | 15-8 | BOOT_DEF1 Mode/Options | Refer to BOOT_DEF0 descriptions |
| BOOT_DEF2 | 23-16 | BOOT_DEF2 Mode/Options | |
| BOOT_DEF3 | 31-24 | BOOT_DEF3 Mode/Options | |
| BOOT_DEF4 | 39-32 | BOOT_DEF4 Mode/Options | |
| BOOT_DEF5 | 47-40 | BOOT_DEF5 Mode/Options | |
| BOOT_DEF6 | 55-48 | BOOT_DEF6 Mode/Options | |
| BOOT_DEF7 | 63-56 | BOOT_DEF7 Mode/Options | |

### 4.3.3 Boot Mode Example Use Cases

This section demonstrates some use cases for configuring the boot mode select pins.

#### 4.3.3.1 Zero Boot Mode Select Pins

This use case demonstrates a scenario for an application that does not use any boot mode select pins and always has the device boot to flash.

1. Program the BOOTPIN_CONFIG location in OTP as follows:
   - Set BOOTPIN_CONFIG.BMSP0 to 0xFF
   - Set BOOTPIN_CONFIG.BMSP1 to 0xFF
   - Set BOOTPIN_CONFIG.BMSP2 to 0xFF
   - Set BOOTPIN_CONFIG.KEY to 0x5A for boot ROM to treat these register bits as valid.
2. Program the BOOTDEF location options for the device. This essentially sets up a device-specific boot mode table.
   - Set BOOTDEF.BOOTDEF0 to 0x03 for booting to flash with a boot mode value of 0
   - Optionally: Set BOOTDEF.BOOT_DEF0_ALT_OPTIONS to a different value to switch to one of the available flash entry point alternatives.

**Table 4-8. Zero Boot Pin Boot Table Result**

| Boot Mode Number | Boot Mode |
|---|---|
| 0 | Flash Boot (0x03) |

Refer to *Entry Points* for the available alternative entry point addresses.

#### 4.3.3.2 One Boot Mode Select Pin

This use case demonstrates a scenario for an application using one boot mode select pin to select between booting to flash or using CAN boot.

1. Program the BOOTPIN_CONFIG location in OTP as follows:
   - Set BOOTPIN_CONFIG.BMSP0 to a user specified GPIO, such as 0x0 for GPIO0

- Set BOOTPIN_CONFIG.BMSP1 to 0xFF
- Set BOOTPIN_CONFIG.BMSP2 to 0xFF
- Set BOOTPIN_CONFIG.KEY to 0x5A for boot ROM to treat these register bits as valid.

2. Program the BOOTDEF location options for the device. This essentially sets up a device-specific boot mode table.
   - Set BOOTDEF.BOOTDEF0 to 0x02 for CAN booting with a boot mode value of 0
   - Set BOOTDEF.BOOTDEF1 to 0x03 for booting to flash with a boot mode value of 1
   - Optionally: Set BOOTDEF.BOOT_DEF1_ALT_OPTIONS to a different value to switch to one of the available flash entry point alternatives.

**Table 4-9. One Boot Pin Boot Table Result**

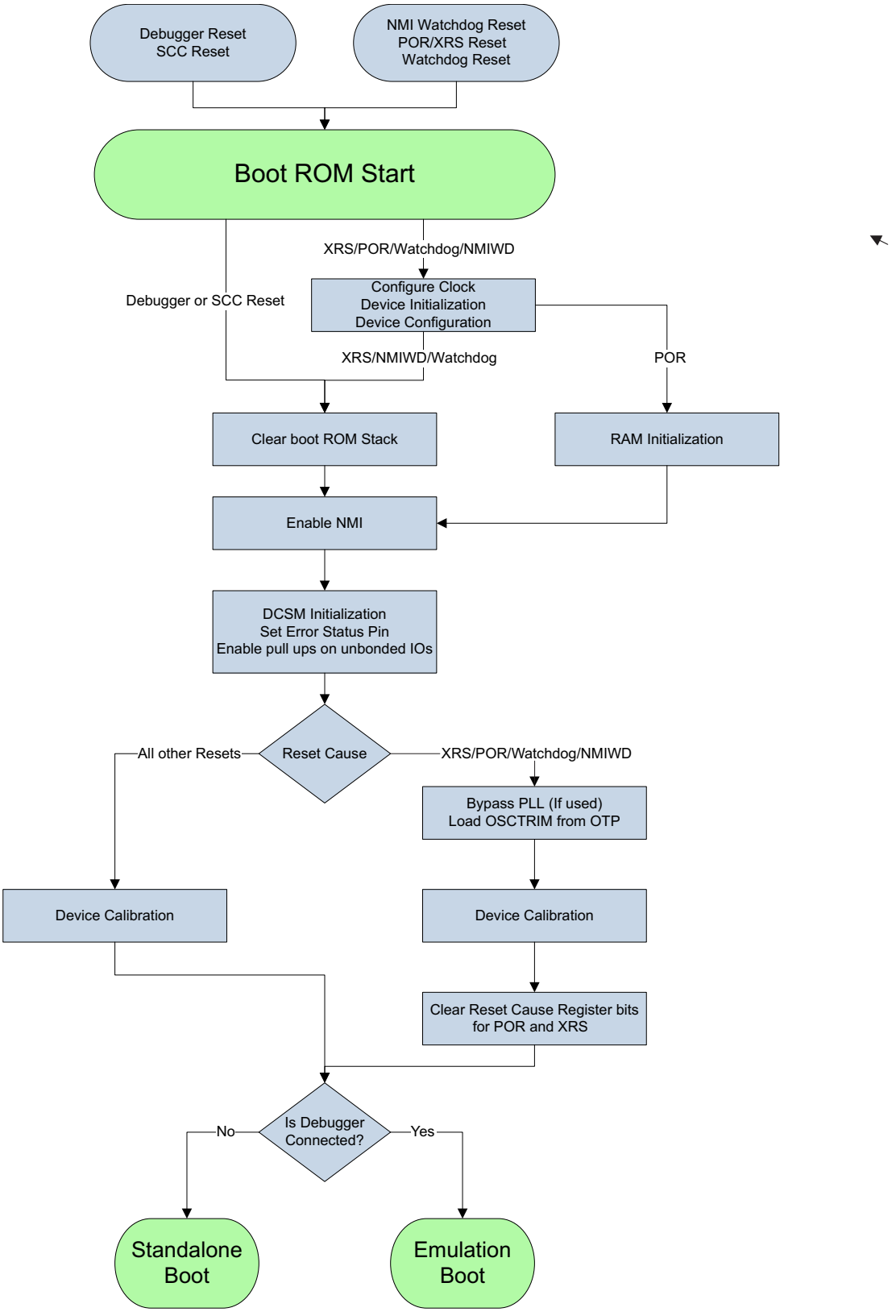
| Boot Mode Number | Boot Mode |
|---|---|
| 0 | CAN Boot (0x02) |
| 1 | Flash Boot (0x03) |

Refer to *Entry Points* for the available alternative entry point addresses.

## 4.4 Device Boot Flow Diagrams

Figure 4-1 shows the device boot flow detailing the actions executed by boot ROM after a reset.

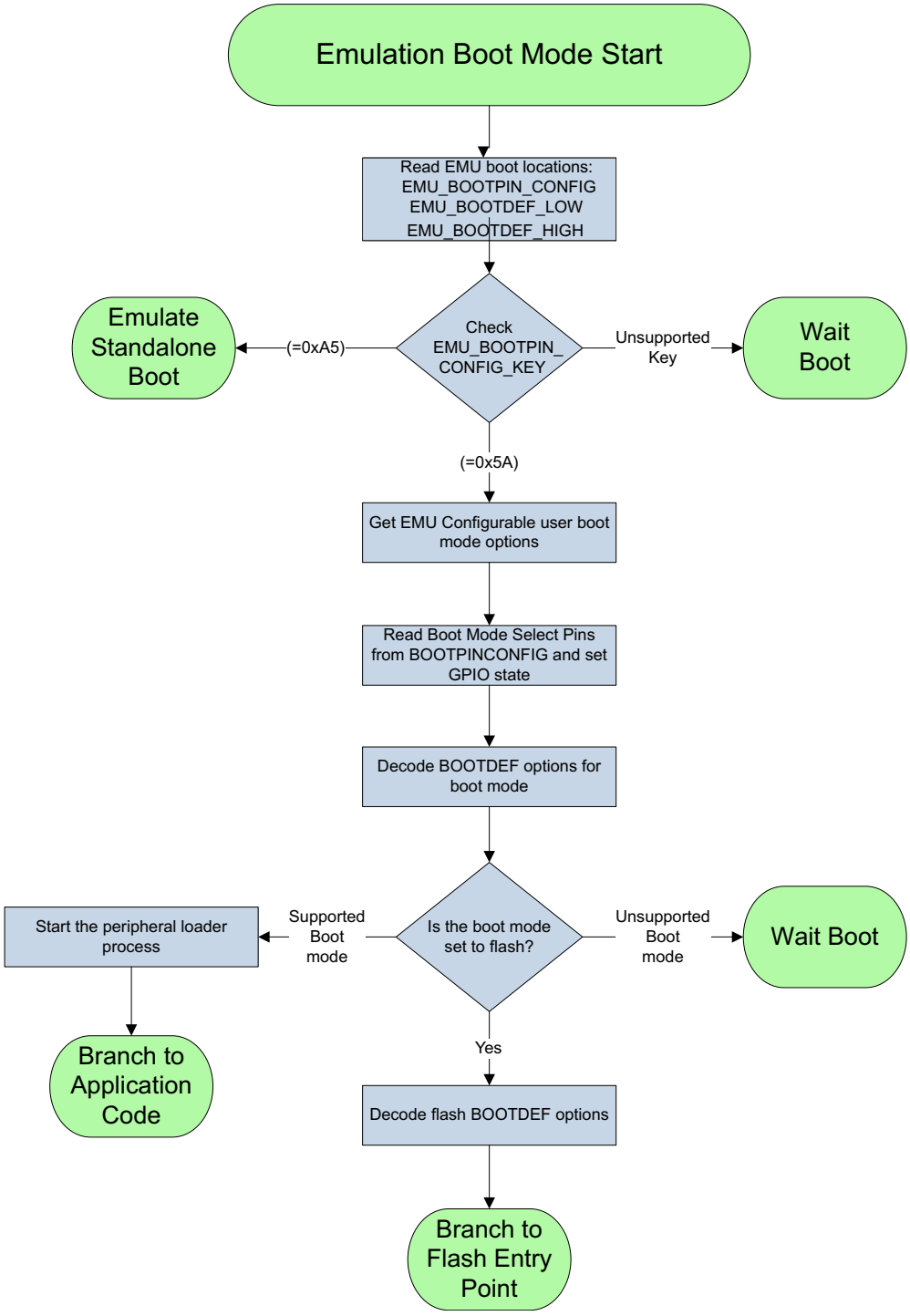
**Figure 4-1. Device Boot Flow**



### 4.4.1 Emulation Boot Flow Diagram

Figure 4-2 shows the device boot flow when running the device in emulation mode.

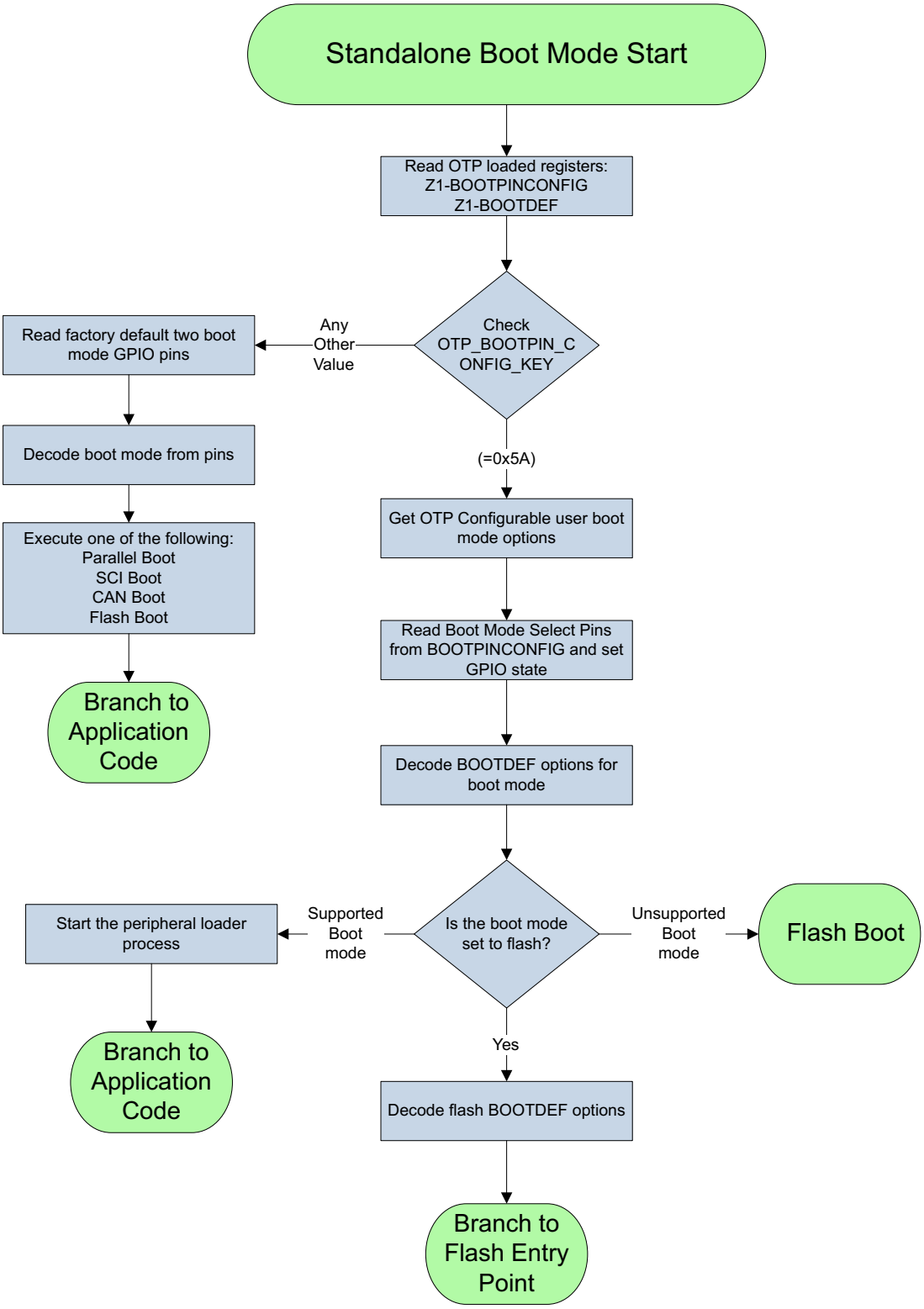
**Figure 4-2. Emulation Boot Flow**



### 4.4.2 Standalone Boot Flow Diagram

Figure 4-3 shows the device boot flow when running the device in standalone boot mode.

**Figure 4-3. Standalone Boot Flow**



## 4.5 Device Reset and Exception Handling

### 4.5.1 Reset Causes and Handling

This section explains the actions boot ROM performs upon reset after checking the reset cause.

**Table 4-10. Boot ROM Reset Causes and Actions**

| Reset Source | Boot ROM Actions |
|---|---|
| POR | 1. Adjust clock divider to /1<br>2. Device configuration<br>3. RAM initialization<br>4. Continue default boot flow |
| XRS | 1. Adjust clock divider to /1<br>2. Device configuration<br>3. Clear boot stack<br>4. Continue default boot flow |
| WDRS | 1. Adjust clock divider to /1<br>2. Device configuration<br>3. Clear boot stack<br>4. Continue default boot flow |
| NMIWDRS | 1. Adjust clock divider to /1<br>2. Device configuration<br>3. Clear boot stack<br>4. Continue default boot flow |
| SCCRESET | 1. Clear boot stack<br>2. Continue default boot flow |
| Debugger Reset | 1. Clear boot stack<br>2. Continue default boot flow |

### 4.5.2 Exceptions and Interrupts Handling

This section explains the actions boot ROM performs if any exceptions that can occur during boot.

**Table 4-11. Boot ROM Exceptions and Actions**

| Exception Event Source | Boot ROM Action | Event Logged |
|---|---|---|
| Single-bit error in FUSEERR | Ignore and continue to boot | No |
| Multi-bit error in FUSEERR | Reset the device | No |
| Clock fail condition detected | Clear NMI, log, and continue to boot | Yes |
| Double- bit ECC error from RAM | Reset the device | Yes |
| Double-bit error from Flash | Reset the device | Yes |
| Software NMI error | Let the device reset | Yes |
| ITRAP Exception | Give the ROM location where it will loop | Yes |

**NOTE:** The above NMI errors are logged into a RAM variable for an application to read it when it starts. Refer to *Boot Status Information* for more details on the boot status information.

## 4.6 Boot ROM Description

This section explains the details regarding the device boot ROM.

### 4.6.1 Boot ROM Registers

The boot ROM code accesses several memory addresses and registers during execution. There are two sets of addresses, one for emulation and one for standalone boot flow. The emulation locations emulate OTP configurations and can be written to as many times as needed. The user configurable DCSM OTP locations used in the standalone boot flow program the device OTP and thus can only be written once. Table 4-12 details these locations.

**Table 4-12. Boot ROM Registers**

| Boot Flow | DCSM Name | Boot ROM Name | Address |
|---|---|---|---|
| Emulation | Z1-GPREG1 | EMU-BOOTPIN-CONFIG | 0x0000 0D00 |
| | Z1-GPREG3 | EMU-BOOTDEF-LOW | 0x0000 0D04 |
| | Z1-BOOTCTRL | EMU-BOOTDEF-HIGH | 0x0000 0D06 |
| Standalone | Z1-GPREG1 | Z1-OTP-BOOTPIN-CONFIG | 0x0005 F008 |
| | Z1-GPREG2 | Z1-OTP-BOOT-GPREG2 | 0x0005 F00A |
| | Z1-GPREG3 | Z1-OTP-BOOTDEF-LOW | 0x0005 F00C |
| | Z1-BOOTCTRL | Z1-OTP-BOOTDEF-HIGH | 0x0005 F004 |

### 4.6.2 Boot ROM User OTP

The boot ROM user configurable DCSM OTP field descriptions and memory addresses are detailed in Table 4-13.

**Table 4-13. User-Configurable DCSM OTP Fields**

| Field name | Description | Bank 0 |
|---|---|---|
| | | USER OTP Address |
| GPREG1[0:15] | BOOTPIN_CONFIG [0:15] | 0x7800C |
| GPREG1[16:31] | BOOTPIN_CONFIG [16:31] | 0x7800D |
| GPREG2[0:15] | GPREG2[0:15] | 0x7800E |
| GPREG2[16:31] | GPREG2[16:31] | 0x7800F |
| GPREG3[0:15] | BOOTDEF_CONFIG[0:15] | 0x7801C |
| GPREG3[16:31] | BOOTDEF_CONFIG[16:31] | 0x7801D |
| BOOTCTRL[0:15] | BOOTDEF_CONFIG[32:47] | 0x7801E |
| BOOTCTRL[16:31] | BOOTDEF_CONFIG[48:63] | 0x7801F |

### 4.6.3 Entry Points

This section gives details about the entry point addresses for various boot modes. These entry points tell the boot ROM where to branch to at the end of booting as per the selected boot mode.

**Table 4-14. Entry Point Addresses**

| Entry Point | Address |
|---|---|
| RAM | 0x0000 0000 |
| Flash (Option 1 – Default) | 0x0008 0000 |
| Flash (Option 2) | 0x0008 EFF0 |
| Flash (Option 3) | 0x0009 0000 |
| Flash (Option 4) | 0x0009 EFF0 |
| PLC | 0x003F 1800 |

### 4.6.4 Wait Points

During boot ROM execution, there are situations where the CPU may enter a wait loop in the code. This state can occur for a variety of reasons. Table 4-15 details the address ranges that the CPU PC register value will fall between if it has entered one of these instances.

**Table 4-15. Wait Point Addresses**

| Address Range | Description |
|---|---|
| 0x003FAD74 – 0x003FB0CD | In Wait Boot |
| 0x000706DC – 0x000706DF | In SCI Boot |
| 0x003FBCD1 – 0x003FBD67 | In NMI Handler |
| 0x003FBD67 – 0x003FBDD5 | In ITRAP ISR |

### 4.6.5 Memory Maps

This section details the ROM memory maps.

#### 4.6.5.1 Boot ROM Memory Map

**Table 4-16. Boot ROM Memory Map (Silicon revision 0,A)**

| Memory | Start Address | End Address | Length |
|---|---|---|---|
| ROM Signature | 0x003F 0000 | 0x003F 0001 | 0x0002 |
| PLC Lite(1) | 0x003F 0002 | 0x003F 1001 | 0x1000 |
| PLC COM(1) | 0x003F 1002 | 0x003F 17A1 | 0x07A0 |
| PLC App Boot(1) | 0x003F 1800 | 0x003F 1AFF | 0x0300 |
| PLC DFU(1) | 0x003F 1B00 | 0x003F 30FF | 0x1600 |
| PLC G3(1) | 0x003F 3100 | 0x003F 535F | 0x2260 |
| PLC Tables(1) | 0x003F 5360 | 0x003F 57EF | 0x0490 |
| PLC MM Tables(1) | 0x003F 5800 | 0x003F 7101 | 0x1902 |
| TI-RTOS (ROM) | 0x003F 7200 | 0x003F 91FF | 0x2000 |
| FPU32 Table | 0x003F 9200 | 0x003F A9FF | 0x1800 |
| Boot | 0x003F AA00 | 0x003F E9CF | 0x3FD0 |
| CPU Spintac Data(1) | 0x003F EA12 | 0x003F EA21 | 0x0010 |
| CPU Fast Data(1) | 0x003F EA22 | 0x003F EB21 | 0x1000 |
| PIE Mismatch Handler | 0x003F FF22 | 0x003F FF71 | 0x0050 |
| CRC Table | 0x003F FF72 | 0x003F FF79 | 0x0008 |
| Version | 0x003F FF7A | 0x003F FF7B | 0x0002 |
| Checksum | 0x003F FF7C | 0x003F FFBD | 0x0042 |
| Vectors | 0x003F FFBE | 0x003F FFFF | 0x0042 |
| TI-RTOS (Flash) | 0x0008 1010 | 0x0008 13EE | 0x3FDF |

(1) Check the data manual to determine if these are available for your device part number. If not available, treat these sections as reserved.

**Table 4-17. Boot ROM Memory Map (Silicon revision B)**

| Memory | Start Address | End Address | Length |
|---|---|---|---|
| ROM Signature | 0x003F 0000 | 0x003F 0001 | 0x0002 |
| PLC Lite(1) | 0x003F 0002 | 0x003F 1001 | 0x1000 |
| PLC COM(1) | 0x003F 1002 | 0x003F 17A1 | 0x07A0 |
| PLC App Boot(1) | 0x003F 1800 | 0x003F 1AFF | 0x0300 |

(1) Check the data manual to determine if these are available for your device part number. If not available, treat these sections as reserved.

**Table 4-17. Boot ROM Memory Map (Silicon revision B) (continued)**

| Memory | Start Address | End Address | Length |
|---|---|---|---|
| PLC DFU[1] | 0x003F 1B00 | 0x003F 30FF | 0x1600 |
| PLC G3[1] | 0x003F 3100 | 0x003F 535F | 0x2260 |
| PLC Tables[1] | 0x003F 5360 | 0x003F 57EF | 0x0490 |
| PLC MM Tables[1] | 0x003F 5800 | 0x003F 7101 | 0x1902 |
| TI-RTOS (ROM) | 0x003F 7200 | 0x003F 91FF | 0x2000 |
| FPU32 Tables (1024pt CFFT/RFFT) | 0x003F 9200 | 0x003F 9FF7 | 0x0DF8 |
| FPU32 Tables (512pt CFFT) | 0x003F A200 | 0x003F A9FF | 0x0800 |
| Boot | 0x003F AA00 | 0x003F E9CF | 0x3FD0 |
| CPU Spintac Data[1] | 0x003F EA12 | 0x003F EA21 | 0x0010 |
| CPU Fast Data[1] | 0x003F EA22 | 0x003F EB21 | 0x1000 |
| Flash API Library (ROM) | 0x003F EB22 | 0x003F F634 | 0x0B13 |
| Flash API Table | 0x003F FEFA | 0x003F FF21 | 0x0028 |
| PIE Mismatch Handler | 0x003F FF22 | 0x003F FF71 | 0x0050 |
| CRC Table | 0x003F FF72 | 0x003F FF79 | 0x0008 |
| Version | 0x003F FF7A | 0x003F FF7B | 0x0002 |
| Checksum | 0x003F FF7C | 0x003F FFBD | 0x0042 |
| Vectors | 0x003F FFBE | 0x003F FFFF | 0x0042 |
| TI-RTOS (Flash) | 0x0008 1010 | 0x0008 13EE | 0x03DF |

#### 4.6.5.2 CLA Data ROM Memory Map

**Table 4-18. CLA Data ROM Memory Map**

| Memory | Start Address | End Address | Length |
|---|---|---|---|
| FFT Tables (Load) | 0x0100 1070 | 0x0100 186F | 0x0800 |
| Data (Load) | 0x0100 1870 | 0x0100 1FF9 | 0x078A |
| Version (Load) | 0x0100 1FFA | 0x0100 1FFF | 0x0006 |
| FFT Tables (Run) | 0x0000 F070 | 0x0000 F86F | 0x0800 |
| Data (Run) | 0x0000 F870 | 0x0000 FFF9 | 0x078A |
| Version (Run) | 0x0000 FFFA | 0x0000 FFFF | 0x0006 |

#### 4.6.5.3 Reserved RAM and Flash Memory Map

This section details memory usage in RAM and Flash that is reserved for boot ROM to use. These memory sections should be reserved in the user application.

**Table 4-19. Reserved RAM and Flash Memory Map (Silicon revision A)**

| Memory | Description | Start Address | End Address | Length |
|---|---|---|---|---|
| RAM | Boot ROM | 0x0000 0002 | 0x0000 00F3 | 0x00F1 |
| | TI-RTOS(1) | 0x0000 0780 | 0x0000 07FF | 0x0080 |
| Flash | TI-RTOS(1)(2) | 0x0008 1010 | 0x0008 13EE | 0x03DF |

(1) If the user is 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 should be verified that there is no secure zone claiming this sector.

**Table 4-20. Reserved RAM and Flash Memory Map (Silicon revision B)**

| Memory | Description | Start Address | End Address | Length |
|---|---|---|---|---|
| RAM | Boot ROM | 0x0000 0002 | 0x0000 00F3 | 0x00F1 |
| | Flash API Library(1) | 0x0000 0760 | 0x0000 077F | 0x0020 |
| | TI-RTOS(1) | 0x0000 0780 | 0x0000 07FF | 0x0080 |
| Flash | TI-RTOS(2) | 0x0008 1010 | 0x0008 13EE | 0x03DF |

(1) If the user is not planning on using Flash API or TI-RTOS in ROM, then these memory locations are free to be used by the application.
(2) For using the TI-RTOS in flash sector A, TI recommends that this sector be made unsecure, or at minimum, the sector should be verified that there is no secure zone claiming this sector.

### *4.6.6 ROM Tables*

This section details the boot ROM and CLA ROM symbol tables.

#### 4.6.6.1 Boot ROM Tables

The boot ROM and flash API symbols and their addresses can be located in the .map file included with the released boot ROM source and header code. Within the .map file, locate the Global Symbols category to get a list of the boot ROM symbols and addresses present.

#### 4.6.6.2 CLA ROM Tables

This section has the symbol tables included in ROM for the CLA and their addresses.

**Table 4-21. CLA ROM Tables**

| Table | Address |
|---|---|
| _cla_twiddleFactors | 0x0000 f070 |
| _cla_bitReversalTable | 0x0000 f470 |
| _CLAatan2HalfPITable | 0x0000 f870 |
| _CLAINV2PI | 0x0000 f874 |
| _CLAatan2Table | 0x0000 f876 |
| _CLAasinHalfPITable | 0x0000 f9fc |
| _CLAatan2TableEnd | 0x0000 f9fc |
| _CLAasinTable | 0x0000 fa00 |
| _CLAacosinHalfPITable | 0x0000 fb86 |
| _CLAasinTableEnd | 0x0000 fb86 |
| _CLAacosinTable | 0x0000 fb8a |
| _CLAacosinTableEnd | 0x0000 fd0a |
| _CLAsinTable | 0x0000 fd0a |
| _CLAsincosTable | 0x0000 fd0a |
| _CLAsincosTable_Sin0 | 0x0000 fd0a |
| _CLAcosTable | 0x0000 fd4a |
| _CLAsincosTable_Cos0 | 0x0000 fd4a |
| _CLAsinTableEnd | 0x0000 fe0a |
| _CLAcosTableEnd | 0x0000 fe4c |
| _CLAsincosTable_TABLE_SIZE | 0x0000 fe4c |
| _CLAsincosTable_TABLE_SIZEDivTwoPi | 0x0000 fe4e |
| _CLAsincosTable_TwoPiDivTABLE_SIZE | 0x0000 fe50 |
| _CLAsincosTable_TABLE_MASK | 0x0000 fe52 |
| _CLAsincosTable_Coef0 | 0x0000 fe54 |
| _CLAsincosTable_Coef1 | 0x0000 fe56 |
| _CLAsincosTable_Coef1_pos | 0x0000 fe58 |
| _CLAsincosTable_Coef2 | 0x0000 fe5a |
| _CLAsincosTable_Coef3 | 0x0000 fe5c |
| _CLAsincosTable_Coef3_neg | 0x0000 fe5e |
| _CLALNV2 | 0x0000 fe60 |
| _CLAsincosTableEnd | 0x0000 fe60 |
| _CLALNVe | 0x0000 fe62 |
| _CLALNV10 | 0x0000 fe64 |
| _CLABIAS | 0x0000 fe66 |
| _CLALN_TABLE_MASK1 | 0x0000 fe68 |
| _CLALN_TABLE_MASK2 | 0x0000 fe6a |
| _CLALnTable | 0x0000 fe6c |
| _CLAINV1 | 0x0000 ff32 |
| _CLALnTableEnd | 0x0000 ff32 |
| _CLAINV2 | 0x0000 ff34 |
| _CLAINV3 | 0x0000 ff36 |
| _CLAINV4 | 0x0000 ff38 |
| _CLAINV5 | 0x0000 ff3a |
| _CLAINV6 | 0x0000 ff3c |
| _CLAINV7 | 0x0000 ff3e |
| _CLALOG10 | 0x0000 ff40 |
| _CLAExpTable | 0x0000 ff42 |
| _CLAExpTableEnd | 0x0000 fff4 |

### 4.6.7 Boot Modes

The available boot modes supported on this device are detailed in this section. Each boot mode allows for various options, providing configurations with different IOs to be used depending on the application.

#### 4.6.7.1 Wait Boot Mode

The wait boot mode puts the CPU in a loop and does not branch to the user application code. The device can enter wait boot mode either manually or because an error occurred during boot up. TI recommends using wait boot when using a debugger to avoid any JTAG complications.

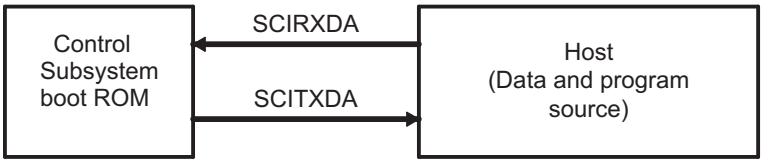
Actions resulting in entering wait boot mode:

- Wait boot is set by the user as the boot mode
- The boot mode is unrecognized and a debugger is connected to the device
- The emulation BOOTPIN_CONFIG key is not equal to 0xA5 or 0x5A
- An error occurs during emulation boot and the boot mode pins are decoded with a value not recognized as a valid boot mode

#### 4.6.7.2 SCI Boot Mode

The SCI boot mode asynchronously transfers code from SCI-A to internal memory. This boot mode only supports an incoming 8-bit data stream and follows the data flow as outlined in Example 4-1.

**Figure 4-4. Overview of SCI Bootloader Operation**



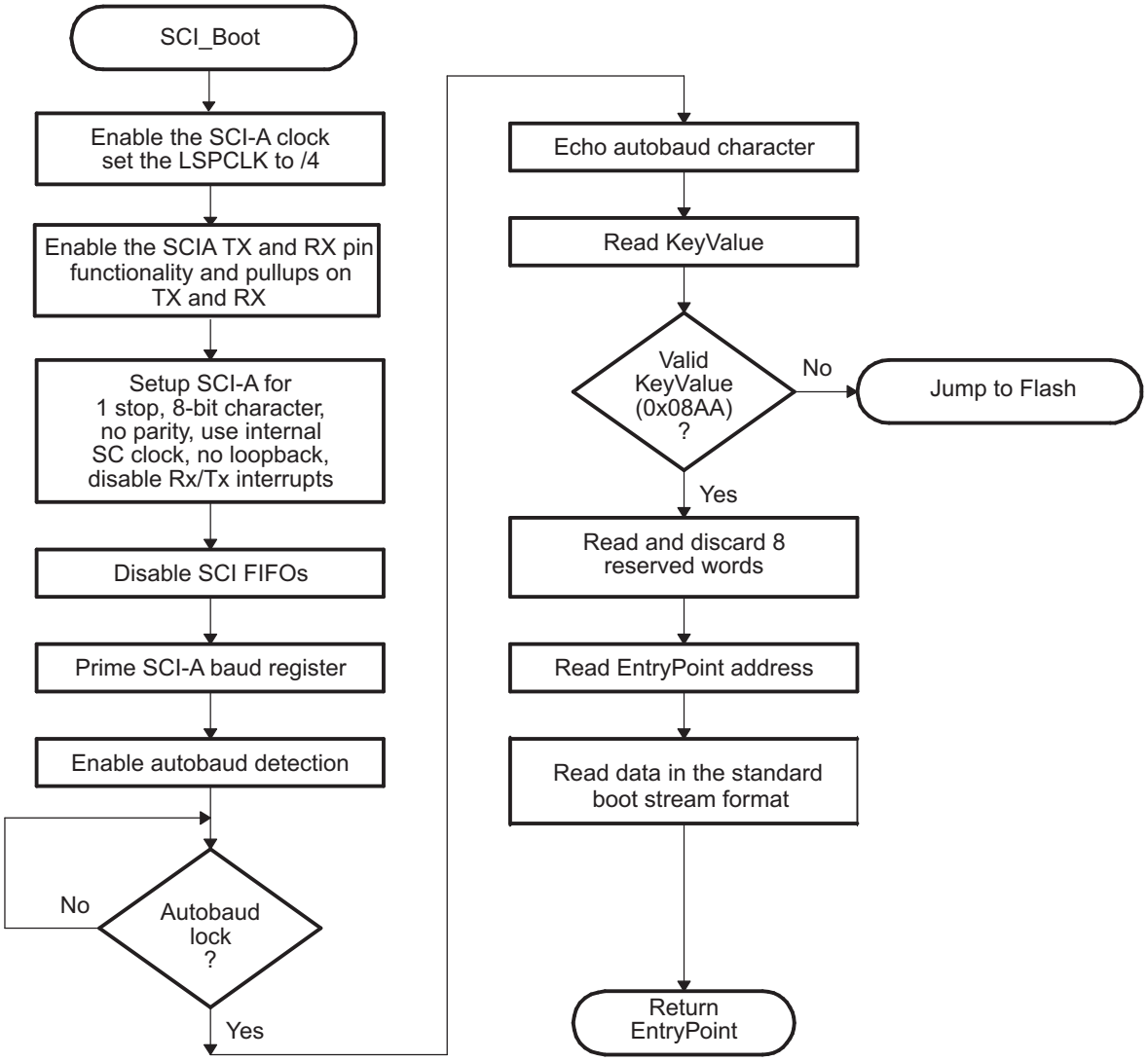

The device communicates with the external host by communication through the SCI-A peripheral. The autobaud feature of the SCI port is used to lock baud rates with the host. For this reason the SCI loader is very flexible and you can use a number of different baud rates to communicate with the device.

After each data transfer, the bootloader will echo back the 8-bit character received to the host. This allows the host to check that each character was received by the bootloader.

At higher baud rates, the slew rate of the incoming data bits can be affected by transceiver and connector performance. While normal serial communications may work well, this slew rate may limit reliable auto-baud detection at higher baud rates (typically beyond 100kbaud) and cause the auto-baud lock feature to fail. To avoid this, the following is recommended:

1. Achieve a baud-lock between the host and SCI bootloader using a lower baud rate.
2. Load the incoming application or custom loader at this lower baud rate.
3. The host may then handshake with the loaded application to set the SCI baud rate register to the desired high baud rate.

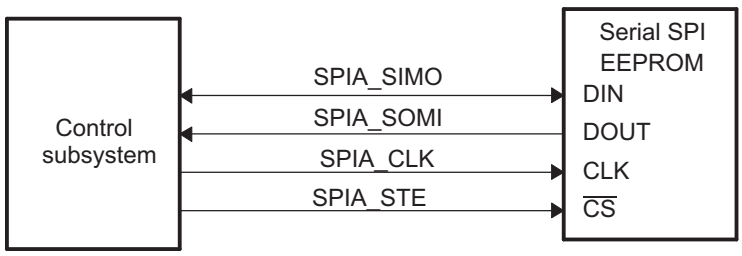
**Figure 4-5. Overview of SCI Boot Function**




### 4.6.7.3 SPI Boot Mode

The SPI loader expects an SPI-compatible 16-bit or 24-bit addressable serial EEPROM or serial flash device to be present on the SPI-A pins as indicated in Figure 4-6. The SPI bootloader supports an 8-bit data stream. It does not support a 16-bit data stream.

**Figure 4-6. SPI Loader**




The SPI boot ROM loader initializes the SPI module to interface to a serial SPI EEPROM or flash. Devices of this type include, but are not limited to, the Xicor X25320 (4Kx8) and Xicor X25256 (32Kx8) SPI serial SPI EEPROMs and the Atmel AT25F1024A serial flash.

The SPI boot ROM loader initializes the SPI with the following settings: FIFO enabled, 8-bit character, internal SPICLK master mode and talk mode, clock phase = 1, polarity = 0, using the slowest baud rate.

If the download is to be performed from an SPI port on another device, then that device must be setup to operate in the slave mode and mimic a serial SPI EEPROM. Immediately after entering the SPI_Boot function, the pin functions for the SPI pins are set to primary and the SPI is initialized. The initialization is done at the slowest speed possible. Once the SPI is initialized and the key value read, you could specify a change in baud rate or low speed peripheral clock.

**Table 4-22. SPI 8-Bit Data Stream**

| Byte | Contents |
|---|---|
| 1 | LSB: AA (KeyValue for memory width = 8-bits) |
| 2 | MSB: 08h (KeyValue for memory width = 8-bits) |
| 3 | LSB: LOSPCP |
| 4 | MSB: SPIBRR |
| 5 | LSB: reserved for future use |
| 6 | MSB: reserved for future use |
| ... | ... |
| ... | Data for this section. |
| | ... |
| 17 | LSB: reserved for future use |
| 18 | MSB: reserved for future use |
| 19 | LSB: Upper half (MSW) of Entry point PC[23:16] |
| 20 | MSB: Upper half (MSW) of Entry point PC[31:24] (Note: Always 0x00) |
| 21 | LSB: Lower half (LSW) of Entry point PC[7:0] |
| 22 | MSB: Lower half (LSW) of Entry point PC[15:8] |
| ... | .... |
| ... | Data for this section. |
| | ... |
| ... | Blocks of data in the format size/destination address/data as shown in the generic data stream description |
| ... | ... |
| ... | Data for this section. |
| | ... |
| n | LSB: 00h |
| n+1 | MSB: 00h - indicates the end of the source |

The data transfer is done in "burst" mode from the serial SPI EEPROM. The transfer is carried out entirely in byte mode (SPI at 8 bits/character). A step-by-step description of the sequence follows:

Step 1. The SPI-A port is initialized

Step 2. The GPIO19 (SPISTE) pin is used as a chip-select for the serial SPI EEPROM or flash

Step 3. The SPI-A outputs a read command for the serial SPI EEPROM or flash

Step 4. The SPI-A sends the serial SPI EEPROM an address 0x0000; that is, the host requires that the EEPROM or flash must have the downloadable packet starting at address 0x0000 in the EEPROM or flash. The loader is compatible with both 16-bit addresses and 24-bit addresses.

Step 5. The next word fetched must match the key value for an 8-bit data stream (0x08AA). The least significant byte of this word is the byte read first and the most significant byte is the next byte fetched. This is true of all word transfers on the SPI. If the key value does not match, then the load is aborted and the bootloader jumps to flash.

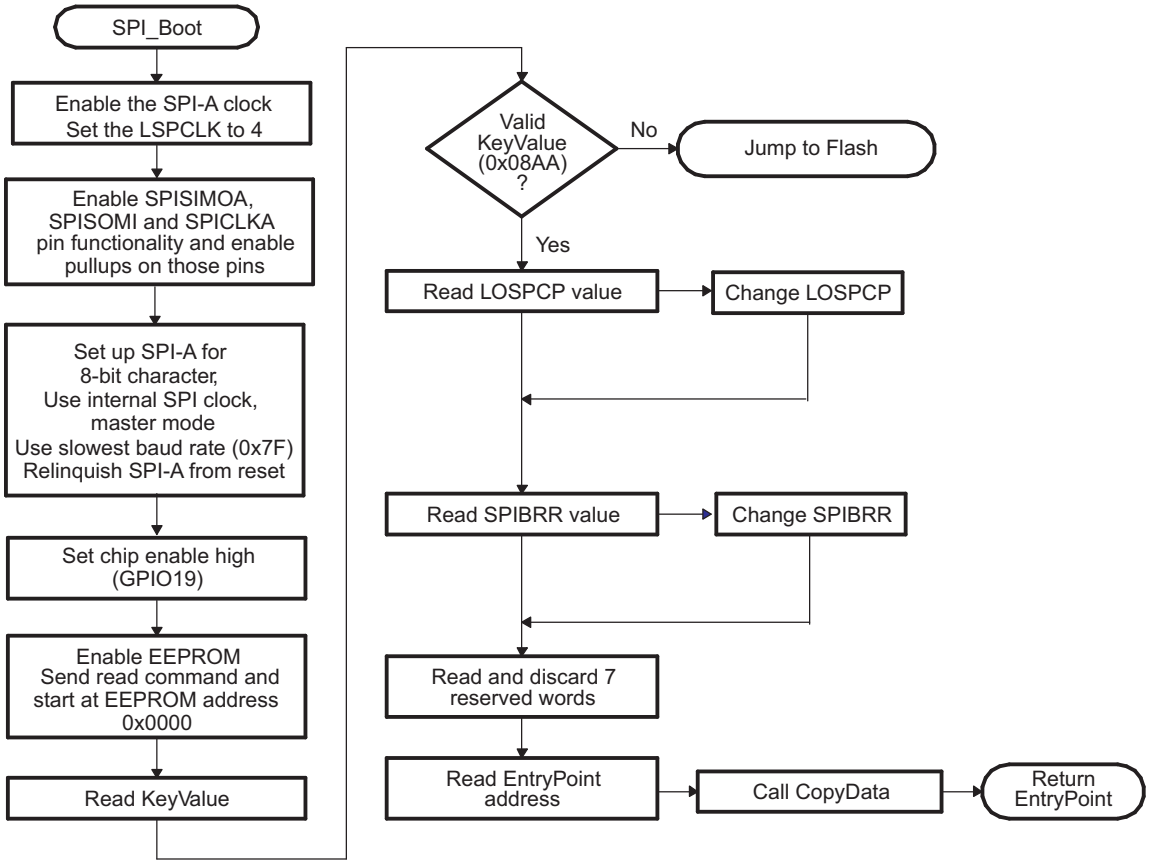
Step 6. The next 2 bytes fetched can be used to change the value of the low speed peripheral clock register (LOSPCP) and the SPI baud rate register (SPIBRR). The first byte read is the LOSPCP value and the second byte read is the SPIBRR value. The next 7 words are reserved for future enhancements. The SPI bootloader reads these 7 words and discards them.

Step 7. The next two words makeup the 32-bit entry point address where execution will continue after

the boot load process is complete. This is typically the entry point for the program being downloaded through the SPI port.

Step 8. Multiple blocks of code and data are then copied into memory from the external serial SPI EEPROM through the SPI port. The blocks of code are organized in the standard data stream structure presented earlier. This is done until a block size of 0x0000 is encountered. At that point in time the entry point address is returned to the calling routine that then exits the bootloader and resumes execution at the address specified.

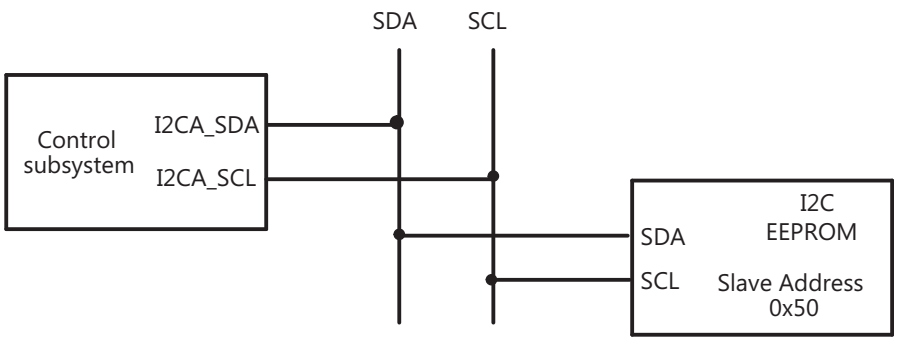
**Figure 4-7. Data Transfer From EEPROM Flow**




### 4.6.7.4 I2C Boot Mode

The I2C bootloader expects an 8-bit wide I2C-compatible EEPROM device to be present at address 0x50 on the I2C-A bus as indicated in Figure 4-8. The EEPROM must adhere to conventional I2C EEPROM protocol, as described in this section, with a 16-bit base address architecture.

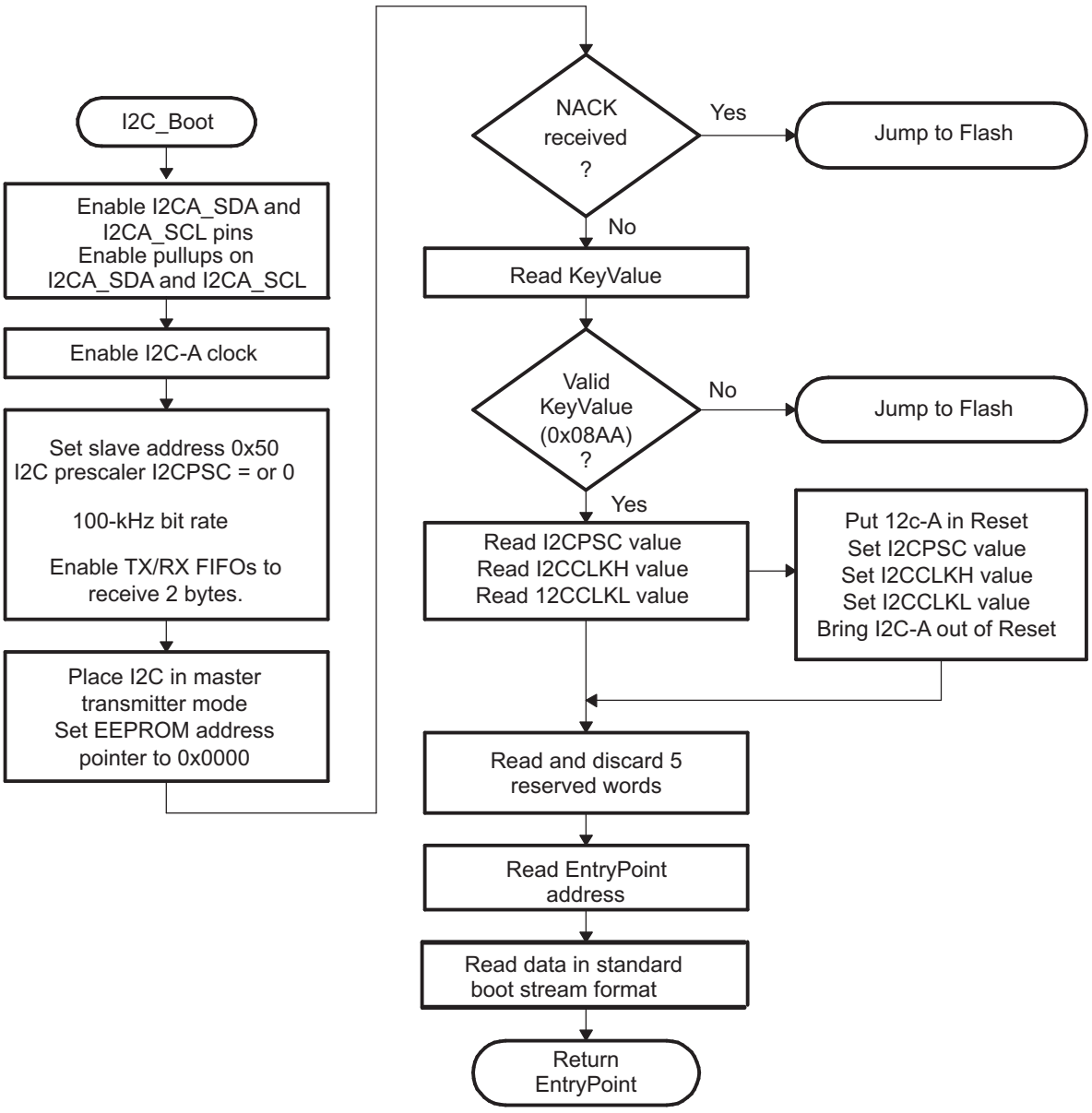
**Figure 4-8. EEPROM Device at Address 0x50**



If the download is to be performed from a device other than an EEPROM, then that device must be set up to operate in the slave mode and mimic the I2C EEPROM. Immediately after entering the I2C boot function, the GPIO pins are configured for I2C-A operation and the I2C is initialized. The following requirements must be met when booting from the I2C module:

- The input frequency to the device must be in the appropriate range.
- The EEPROM must be at slave address 0x50.

**Figure 4-9. Overview of I2C Boot Function**




The bit-period prescalers (I2CCLKH and I2CCLKL) are configured by the bootloader to run the I2C at a 50 percent duty cycle at 100-kHz bit rate (standard I2C mode) when the system clock is 10 MHz. These registers can be modified after receiving the first few bytes from the EEPROM. This allows the communication to be increased up to a 400-kHz bit rate (fast I2C mode) during the remaining data reads.

Arbitration, bus busy, and slave signals are not checked. Therefore, no other master is allowed to control the bus during this initialization phase. If the application requires another master during I2C boot mode, that master must be configured to hold off sending any I2C messages until the application software signals that it is past the bootloader portion of initialization.

The non-acknowledgment bit is checked only during the first message sent to initialize the EEPROM base address. This is to make sure that an EEPROM is present at address 0x50 before continuing. If an EEPROM is not present, the non-acknowledgment bit is not checked during the address phase of the data read messages (I2C_Get Word). If a non acknowledgment is received during the data read messages, the I2C bus will hang. Table 18-1 shows the 8-bit data stream used by the I2C.

**Table 4-23. I2C 8-Bit Data Stream**

| Byte | Contents |
|---|---|
| 1 | LSB: AA (KeyValue for memory width = 8 bits) |
| 2 | MSB: 08h (KeyValue for memory width = 8 bits) |
| 3 | LSB: I2CPSC[7:0] |
| 4 | reserved |
| 5 | LSB: I2CCLKH[7:0] |
| 6 | MSB: I2CCLKH[15:8] |
| 7 | LSB: I2CCLKL[7:0] |
| 8 | MSB: I2CCLKL[15:8] |
| ... | ... |
| ... | Data for this section. |
| | ... |
| 17 | LSB: Reserved for future use |
| 18 | MSB: Reserved for future use |
| 19 | LSB: Upper half of entry point PC |
| 20 | MSB: Upper half of entry point PC[22:16] (Note: Always 0x00) |
| 21 | LSB: Lower half of entry point PC[15:8] |
| 22 | MSB: Lower half of entry point PC[7:0] |
| ... | ... |
| ... | Data for this section. |
| | ... |
| | Blocks of data in the format size/destination address/data as shown in the generic data stream description. |
| ... | ... |
| ... | Data for this section. |
| | ... |
| n | LSB: 00h |
| n+1 | MSB: 00h - indicates the end of the source |

The I2C EEPROM protocol required by the I2C bootloader is shown in Figure 4-10 and Figure 4-11. The first communication, which sets the EEPROM address pointer to 0x0000 and reads the KeyValue (0x08AA) from it, is shown in Figure 4-10. All subsequent reads are shown in Figure 4-11 and are read two bytes at a time.

**Figure 4-10. Random Read**

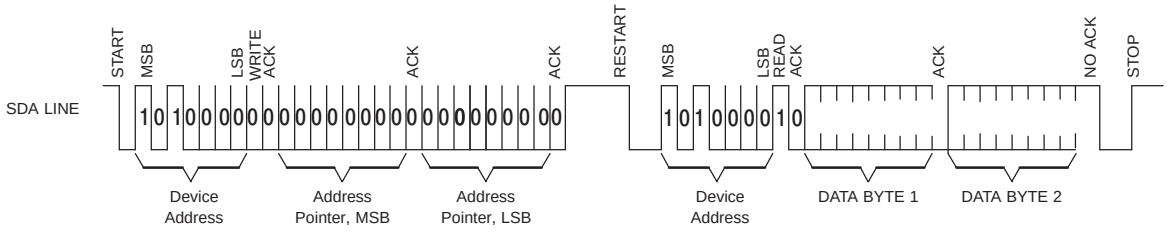
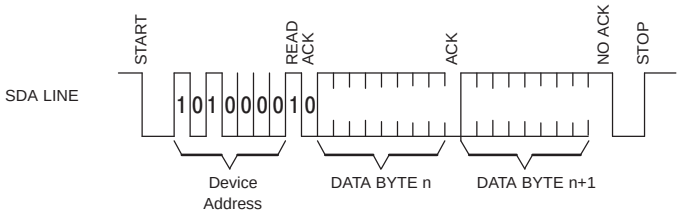


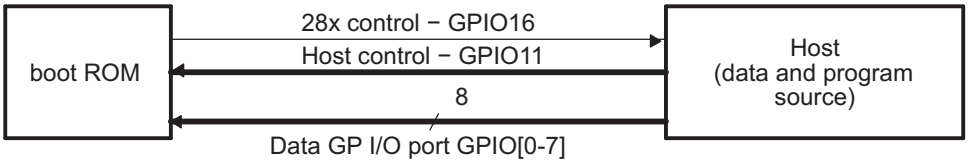
**Figure 4-11. Sequential Read**




### 4.6.7.5 Parallel Boot Mode

The parallel general purpose I/O (GPIO) boot mode asynchronously transfers code from GPIO0 to GPIO7 internal memory. Each value is 8 bits long and follows the same data flow as outlined in Figure 4-12.

**Figure 4-12. Overview of Parallel GPIO Bootloader Operation**




The control subsystem communicates with the external host device by polling/driving the GPIO16 and GPIO11 lines. The handshake protocol shown in Figure 4-13 must be used to successfully transfer each word via GPIO [0-7]. This protocol is very robust and allows for a slower or faster host to communicate with the master subsystem.

Two consecutive 8-bit words are read to form a single 16-bit word. The most significant byte (MSB) is read first followed by the least significant byte (LSB). In this case, data is read from GPIO[0-7].

The 8-bit data stream is shown in Table 4-24.

**Table 4-24. Parallel GPIO Boot 8-Bit Data Stream**

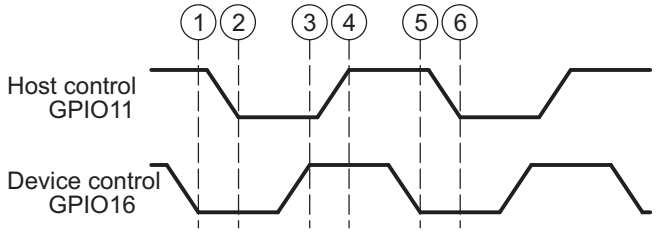
| Bytes | | GPIO[ 7:0] (Byte 1 of 2) | GPIO[ 7:0] (Byte 2 of 2) | Description |
|---|---|---|---|---|
| 1 | 2 | AA | 08 | 0x08AA (KeyValue for memory width = 16bits) |
| 3 | 4 | 00 | 00 | 8 reserved words (words 2 - 9) |
| ... | ... | ... | ... | ... |
| 17 | 18 | 00 | 00 | Last reserved word |
| 19 | 20 | BB | 00 | Entry point PC[22:16] |
| 21 | 22 | DD | CC | Entry point PC[15:0] (PC = 0x00BBCCDD) |
| 23 | 24 | NN | MM | Block size of the first block of data to load = 0xMMNN words |
| 25 | 26 | BB | AA | Destination address of first block Addr[31:16] |
| 27 | 28 | DD | CC | Destination address of first block Addr[15:0] (Addr = 0xAABBCCDD) |
| 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) |

**Table 4-24. Parallel GPIO Boot 8-Bit Data Stream (continued)**

| Bytes | | GPIO[ 7:0] (Byte 1 of 2) | GPIO[ 7:0] (Byte 2 of 2) | Description |
|---|---|---|---|---|
| n+2 | n+3 | 00 | 00 | Block size of 0000h - indicates end of the source program |

The device first signals the host that it is ready to begin data transfer by pulling the GPIO16 pin low. The host load then initiates the data transfer by pulling the GPIO11 pin low. The complete protocol is shown in Figure 4-13:

**Figure 4-13. Parallel GPIO Bootloader Handshake Protocol**




1. The device indicates it is ready to start receiving data by pulling the GPIO16 pin low.
2. The bootloader waits until the host puts data on GPIO [0-7]. The host signals to the device that data is ready by pulling the GPIO11 pin low.
3. The device reads the data and signals the host that the read is complete by pulling GPIO16 high.
4. The bootloader waits until the host acknowledges the device by pulling GPIO11 high.
5. The device again indicates it is ready for more data by pulling the GPIO16 pin low.

This process is repeated for each data value to be sent.

Figure 4-14 shows an overview of the Parallel GPIO bootloader flow.

**Figure 4-14. Parallel GPIO Mode Overview**

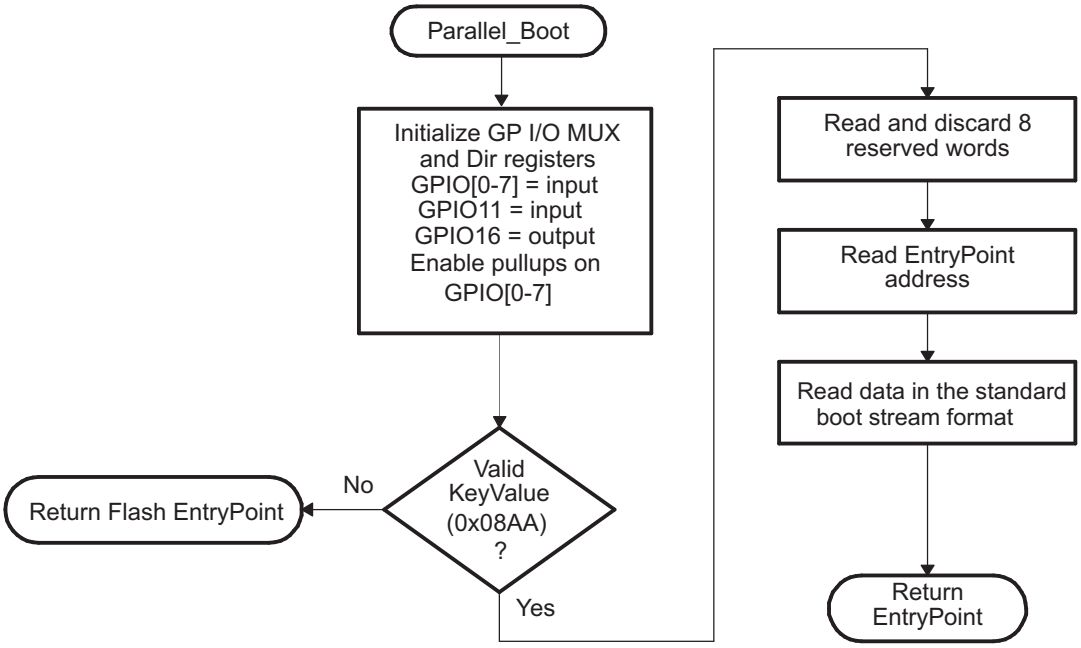



Figure 4-15 shows the transfer flow from the host side. The operating speed of the CPU and host are not critical in this mode as the host will wait for the device and the device will in turn wait for the host. In this manner the protocol will work with both a host running faster and a host running slower than the device.

**Figure 4-15. Parallel GPIO Mode - Host Transfer Flow**

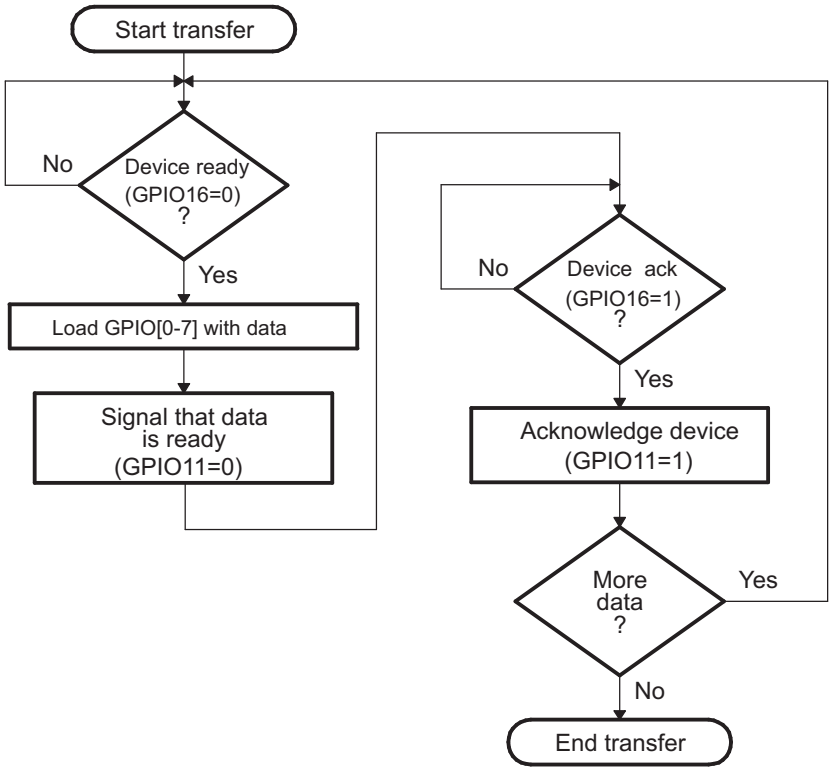
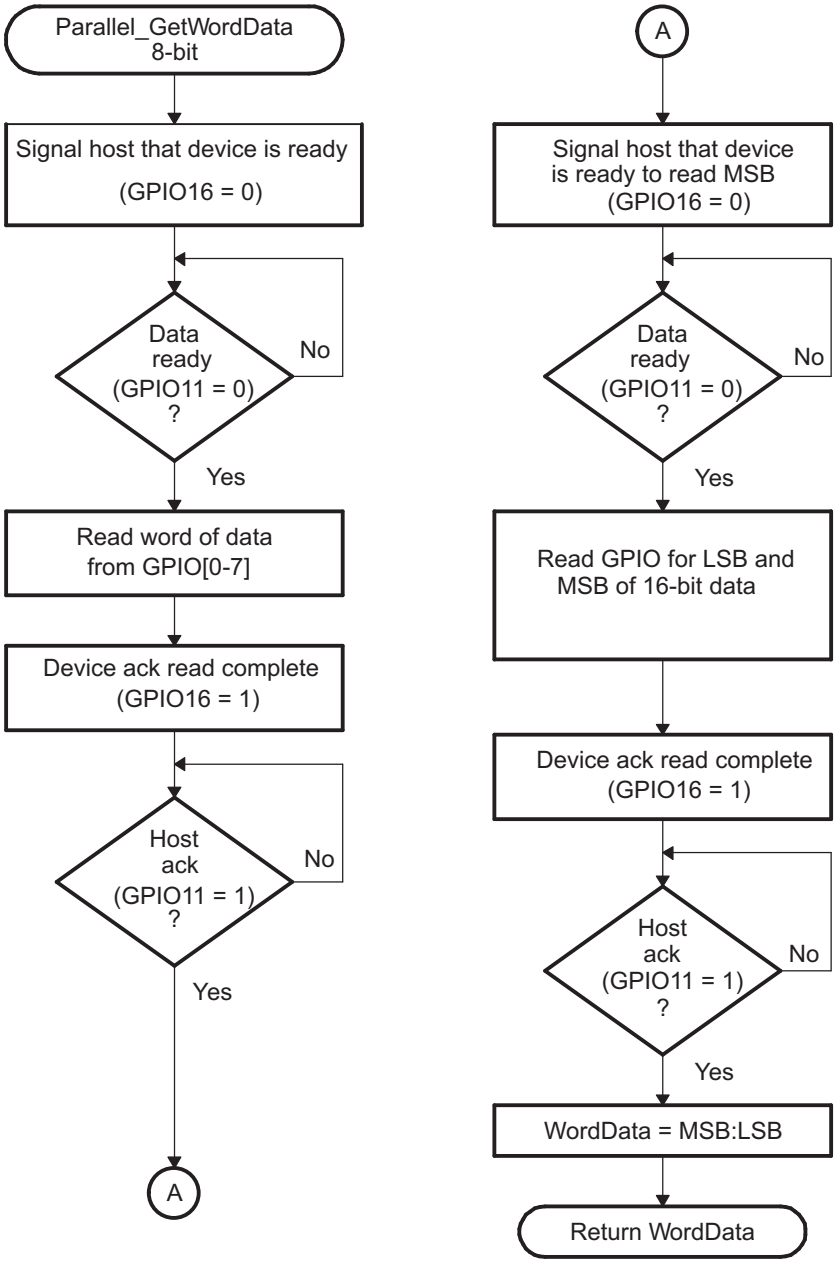



Figure 4-16 shows the flow used to read a single word of data from the parallel port.

- **8-bit data stream**

  The 8-bit routine, shown in Figure 4-16, discards the upper 8 bits of the first read from the port and treats the lower 8 bits masked with GPIO7 in bit position 7 and GPIO6 in bit position 6 as the the least significant byte (LSB) of the word to be fetched. The routine will then perform a second read to fetch the most significant byte (MSB). It then combines the MSB and LSB into a single 16-bit value to be passed back to the calling routine.

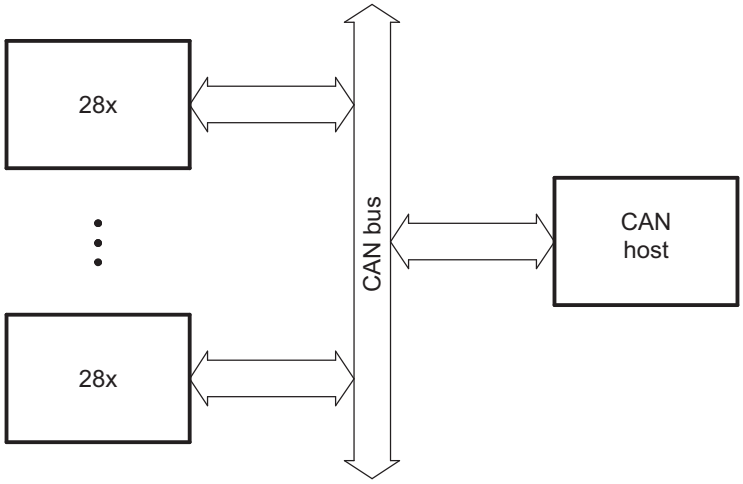
**Figure 4-16. 8-Bit Parallel GetWord Function**




### 4.6.7.6 CAN Boot Mode

The CAN bootloader asynchronously transfers code from CAN-A to internal memory. The host can be any CAN node. The communication is first done with 11-bit standard identifiers (with a MSGID of 0x1) using two bytes per data frame. The host can download a kernel to reconfigure the CAN if higher data throughput is desired.

**Figure 4-17. Overview of CAN-A Bootloader Operation**




The bit timing registers are programmed in such a way that a 100 kbps bit rate is achieved with a 20 MHz external oscillator, a shown in Table 4-25.

**Table 4-25. Bit-Rate Value for Internal Oscillators**

| OSCCLK | SYSCLK | Bit Rate |
|---|---|---|
| 20 MHz | 10 MHz | 100 kbps |

The SYSCLKOUT values shown are the reset values with the default PLL setting. The BRP and bit-time values are hard-coded to 10 and 20, respectively.

Mailbox 1 is programmed with a standard MSGID of 0x1 for boot-loader communication. The CAN host should transmit only 2 bytes at a time, LSB first and MSB next. For example, to transmit the word 0x08AA to the device, transmit AA first, followed by 08. The program flow of the CAN bootloader is identical to the SCI bootloader. The data sequence for the CAN bootloader is shown in Table 4-26:

**Table 4-26. CAN 8-Bit Data Stream**

| Bytes | | Byte 1 of 2 | Byte 2 of 2 | Description |
|---|---|---|---|---|
| 1 | 2 | AA | 08 | 0x08AA (KeyValue for memory width = 16 bits) |
| 3 | 4 | 00 | 00 | reserved |
| 5 | 6 | 00 | 00 | reserved |
| 7 | 8 | 00 | 00 | reserved |
| 9 | 10 | 00 | 00 | reserved |
| 11 | 12 | 00 | 00 | reserved |
| 13 | 14 | 00 | 00 | reserved |
| 15 | 16 | 00 | 00 | reserved |
| 17 | 18 | 00 | 00 | reserved |
| 19 | 20 | BB | AA | Entry point PC[22:16] |
| 21 | 22 | DD | CC | Entry point PC[15:0] (PC = 0xAABBCCDD) |
| 23 | 24 | NN | MM | Block size of the first block of data to load = 0xMMNN words |
| 25 | 26 | BB | AA | Destination address of first block Addr[31:16] |
| 27 | 28 | DD | CC | Destination address of first block Addr[15:0] (Addr = 0xAABBCCDD) |
| 29 | 30 | BB | AA | First word of the first block in the source being loaded = 0xAABB |
| ... | | | | .... |
| ... | | | | Data for this section. |
| | | | | ... |
| . | | BB | AA | Last word of the first block of the source being loaded = 0xAABB |

**Table 4-26. CAN 8-Bit Data Stream (continued)**

| Bytes | | Byte 1 of 2 | Byte 2 of 2 | Description |
|---|---|---|---|---|
| . | | NN | MM | Block size of the 2nd block to load = 0xMMNN words |
| . | | BB | AA | Destination address of second block Addr[31:16] |
| . | | DD | CC | Destination address of second block Addr[15:0] |
| . | | BB | AA | First word of the second block in the source being loaded |
| . | | | | … |
| n | n+1 | BB | AA | Last word of the last block of the source being loaded<br>(More sections if required) |
| n+2 | n+3 | 00 | 00 | Block size of 0000h - indicates end of the source program |

### *4.6.8 Boot Data Stream Structure*

This section details the data transfer protocols or stream structures that allow boot data transfer between boot ROM and host device. This data transfer protocol is compatible to the respective bootloaders on the Piccolo class of C2000 devices.

#### 4.6.8.1 Bootloader Data Stream Structure

The following two tables and associated examples show the structure of the data stream incoming to the bootloader. The basic structure is the same for all the bootloaders and is based on the C54x source data stream generated by the C54x hex utility. The C28x hex utility (hex2000.exe) has been updated to support this structure. The hex2000.exe utility is included with the C2000 code generation tools. All values in the data stream structure are in hex.

The first 16-bit word in the data stream is known as the key value. The key value is used to tell the bootloader the width of the incoming stream: 8 or 16 bits. Note that not all bootloaders will accept both 8 and 16-bit streams. Please refer to the detailed information on each loader for the valid data stream width. For an 8-bit data stream, the key value is 0x08AA and for a 16-bit stream it is 0x10AA. If a bootloader receives an invalid key value, then the load is aborted.

The next eight words are used to initialize register values or otherwise enhance the bootloader by passing values to it. If a bootloader does not use these values then they are reserved for future use and the bootloader simply reads the value and then discards it. Currently only the SPI and I2C and parallel bootloaders use these words to initialize registers.

The tenth and eleventh words comprise the 22-bit entry point address. This address is used to initialize the PC after the boot load is complete. This address is most likely the entry point of the program downloaded by the bootloader.

The twelfth word in the data stream is the size of the first data block to be transferred. The size of the block is defined as 8-bit data stream format. For example, to transfer a block of 20 8-bit data values from an 8-bit data stream, the block size would be 0x000A to indicate 10 16-bit words.

The next two words tell the loader the destination address of the block of data. Following the size and address will be the 16-bit words that makeup that block of data.

This pattern of block size/destination address repeats for each block of data to be transferred. Once all the blocks have been transferred, a block size of 0x0000 signals to the loader that the transfer is complete. At this point the loader will return the entry point address to the calling routine which in turn will cleanup and exit. Execution will then continue at the entry point address as determined by the input data stream contents.

**Table 4-27. LSB/MSB Loading Sequence in 8-Bit Data Stream**

| Byte | | Contents | |
|---|---|---|---|
| | | LSB (First Byte of 2) | MSB (Second Byte of 2) |
| 1 | 2 | LSB: AA (KeyValue for memory width = 8 bits) | MSB: 08h (KeyValue for memory width = 8 bits) |
| 3 | 4 | LSB: Register initialization value or reserved | MSB: Register initialization value or reserved |
| 5 | 6 | LSB: Register initialization value or reserved | MSB: Register initialization value or reserved |
| 7 | 8 | LSB: Register initialization value or reserved | MSB: Register initialization value or reserved |
| ... | ... | ... | ... |
| ... | ... | ... | ... |
| 17 | 18 | LSB: Register initialization value or reserved | MSB: Register initialization value or reserved |
| 19 | 20 | LSB: Upper half of Entry point PC[23:16] | MSB: Upper half of entry point PC[31:24] (Always 0x00) |
| 21 | 22 | LSB: Lower half of Entry point PC[7:0] | MSB: Lower half of Entry point PC[15:8] |
| 23 | 24 | LSB: Block size in words of the first block to load. If the block size is 0, this indicates the end of the source program. Otherwise another block follows. For example, a block size of 0x000A would indicate 10 words or 20 bytes in the block. | MSB: block size |
| 25 | 26 | LSB: MSW destination address, first block Addr[23:16] | MSB: MSW destination address, first block Addr[31:24] |
| 27 | 28 | LSB: LSW destination address, first block Addr[7:0] | MSB: LSW destination address, first block Addr[15:8] |
| 29 | 30 | LSB: First word of the first block being loaded | MSB: First word of the first block being loaded |
| ... | ... | ... | ... |
| ... | ... | ... | ... |
| . | . | LSB: Last word of the first block to load | MSB: Last word of the first block to load |
| . | . | LSB: Block size of the second block | MSB: Block size of the second block |
| . | . | LSB: MSW destination address, second block Addr[23:16] | MSB: MSW destination address, second block Addr[31:24] |
| . | . | LSB: LSW destination address, second block Addr[7:0] | MSB: LSW destination address, second block Addr[15:8] |
| . | . | LSB: First word of the second block being loaded | MSB: First word of the second block being loaded |
| ... | ... | ... | ... |
| ... | ... | ... | ... |
| . | . | LSB: Last word of the second block | MSB: Last word of the second block |
| . | . | LSB: Block size of the last block | MSB: Block size of the last block |
| . | . | LSB: MSW of destination address of last block Addr[23:16] | MSB: MSW destination address, last block Addr[31:24] |
| . | . | LSB: LSW destination address, last block Addr[7:0] | MSB: LSW destination address, last block Addr[15:8] |
| . | . | LSB: First word of the last block being loaded | MSB: First word of the last block being loaded |
| ... | ... | ... | ... |
| ... | ... | ... | ... |
| . | . | LSB: Last word of the last block | MSB: Last word of the last block |
| n | n+1 | LSB: 00h | MSB: 00h - indicates the end of the source |

### *Example 4-1. Data Stream Structure 8-bit*

```
AA 08         ; 0x08AA 8-bit key value
00 00 00 00   ; 8 reserved words
00 00 00 00
00 00 00 00
00 00 00 00
3F 00 00 80   ; 0x003F8000 EntryAddr, starting point after boot load completes
05 00         ; 0x0005 - First block consists of 5 16-bit words
3F 00 10 90   ; 0x003F9010 - First block will be loaded starting at 0x3F9010
01 00         ; Data loaded = 0x0001 0x0002 0x0003 0x0004 0x0005
02 00
03 00
04 00
05 00
02 00         ; 0x0002 - 2nd block consists of 2 16-bit words
```

***Example 4-1. Data Stream Structure 8-bit (continued)***

```
3F 00 00 80   ; 0x003F8000 - 2nd block will be loaded starting at 0x3F8000
00 77         ; Data loaded = 0x7700 0x7625
25 76
00 00         ; 0x0000 - Size of 0 indicates end of data stream

After load has completed the following memory values will have been initialized as follows:

Location   Value
0x3F9010   0x0001
0x3F9011   0x0002
0x3F9012   0x0003
0x3F9013   0x0004
0x3F9014   0x0005
0x3F8000   0x7700
0x3F8001   0x7625
PC Begins execution at 0x3F8000
```

### 4.6.9 GPIO Assignments

This section details the GPIOs and boot options used for each boot mode set in BOOT_DEFx located at Z1-OTP-BOOTDEF-LOW and Z1-OTP-BOOTDEF-HIGH. Refer to *Configuring Alternate Boot Mode Pins* on how to manipulate BOOT_DEFx. When selecting a boot mode option, make sure to verify that the necessary pins are available in the pin mux options for the specific device package being used.

**Table 4-28. SCI Boot Options**

| Option | BOOTDEFx Value | SCIATX GPIO | SCIARX GPIO |
|---|---|---|---|
| 0 (default) | 0x01 | GPIO29 | GPIO28 |
| 1 | 0x21 | GPIO16 | GPIO17 |
| 2 | 0x41 | GPIO8 | GPIO9 |
| 3 | 0x61 | GPIO48 | GPIO49 |
| 4 | 0x81 | GPIO24 | GPIO25 |

**NOTE:** Pull-ups are enabled on the SCIATX and SCIARX pins.

**Table 4-29. CAN Boot Options**

| Option | BOOTDEFx Value | CANTXA GPIO | CANRXA GPIO |
|---|---|---|---|
| 0 (default) | 0x02 | GPIO32 | GPIO33 |
| 1 | 0x22 | GPIO4 | GPIO5 |
| 2 | 0x42 | GPIO31 | GPIO30 |
| 3 | 0x62 | GPIO37 | GPIO35 |

**NOTE:** Pull-ups are enabled on the CANTXA and SCIARX pins.

**Table 4-30. Flash Boot Options**

| Option | BOOTDEFx Value | Flash Entry Point (Address) | Flash Bank,Sector |
|---|---|---|---|
| 0 (default) | 0x03 | Flash – Default Option 1 (0x00080000) | Bank 0, Sector 0 |
| 1 | 0x23 | Flash – Option 2 (0x0008EFF0) | Bank 0, Sector 14 |
| 2 | 0x43 | Flash – Option 3 (0x00090000) | Bank 1, Sector 0 |

**Table 4-30. Flash Boot Options (continued)**

| Option | BOOTDEFx Value | Flash Entry Point (Address) | Flash Bank,Sector |
|---|---|---|---|
| 3 | 0x63 | Flash – Option 4 (0x0009EFF0) | Bank 1, Sector 14 |

**Table 4-31. Wait Boot Options**

| Option | BOOTDEFx Value | Watchdog Status |
|---|---|---|
| 0 | 0x04 | Enabled |
| 1 | 0x24 | Disabled |

**Table 4-32. SPI Boot Options**

| Option | BOOTDEFx Value | SPIA_SIMO | SPIA_SOMI | SPIA_CLK | SPIA_STE |
|---|---|---|---|---|---|
| 1 | 0x26 | GPIO8 | GPIO10 | GPIO9 | GPIO11 |
| 2 | 0x46 | GPIO54 | GPIO55 | GPIO56 | GPIO57 |
| 3 | 0x66 | GPIO16 | GPIO17 | GPIO56 | GPIO57 |
| 4 | 0x86 | GPIO8 | GPIO17 | GPIO9 | GPIO11 |

**NOTE:** Pull-ups are enabled on the SPIA_SIMO, SPIA_SOMI, SPIA_CLK, and SPIA_STE pins.

**Table 4-33. I2C Boot Options**

| Option | BOOTDEFx Value | SDAA GPIO | SCLA GPIO |
|---|---|---|---|
| 0 | 0x07 | GPIO32 | GPIO33 |
| 1 | 0x47 | GPIO26 | GPIO27 |
| 2 | 0x67 | GPIO42 | GPIO43 |

**NOTE:** Pull-ups are enabled on the SDAA and SCLA pins.

**Table 4-34. Parallel Boot Options**

| Option | BOOTDEFx Value | D0-D7 GPIO | DSP Control GPIO | Host Control GPIO |
|---|---|---|---|---|
| 0 (default) | 0x00 | GPIO0-GPIO7 | GPIO16 | GPIO11 |

**NOTE:** Pull-ups are enabled on GPIO0 to GPIO7.

**Table 4-35. RAM Boot Options**

| Option | BOOTDEFx Value | RAM Entry Point (Address |
|---|---|---|
| 0 | 0x05 | 0x00000000 |

### *4.6.10 DCSM Usage*

This section explains how the bit field values from the user-configurable DCSM OTP location, Z1-OTP-BOOT-GPREG2, are decoded by the boot ROM after DCSM initialization is complete.

**Table 4-36. DCSM Z1-OTP-BOOT-GPREG2 Bit Fields**

| Bit | Name | Description | Boot ROM Action |
|---|---|---|---|
| 31-24 | Key | Write 0x5A to these 8-bits to tell the boot ROM code that the bits in this register are valid. | If user set to 0x5A, boot ROM uses the values in this register. If set to any other value, boot ROM ignores the values in this register. |
| 23-8 | Reserved | Reserved | No action |
| 7-6 | Reserved | Reserved | No action |
| 5-4 | Error Status Pin | Sets the GPIO pin to be used as the ERRORSTS<br>0x0 – GPIO24<br>0x1 – GPIO28<br>0x2 – GPIO29<br>0x3 – ERRORSTS disabled (Default) | Boot ROM configures the appropriate mux for the selected GPIO pin. |
| 3-0 | Reserved | | |

**NOTE:** Users should program user DCSM OTP locations Z1-OTP-BOOT-GPREG2 and Z1-OTP-BOOTPIN-CONFIG at the same time, because they share ECC.

### 4.6.11 Clock Initialization

During boot up, the boot ROM initializes the device clocking, depending upon the reset source, to assist in faster boot time response. Clock configurations are performed by the boot ROM code only for POR or XRS reset types. For all other resets, the boot ROM starts executing with the clocks that were already set up before reset.

**Table 4-37. Boot Clock Sources**

| Source | Frequency | Description |
|---|---|---|
| INTOSC2 | 10 MHz | Default clock source |
| INTOSC1 | 10 MHz | Set as clock source if missing clock is detected at power up or right after device reset |

**Table 4-38. Clock State after Boot ROM**

| Reset Source | Clock State |
|---|---|
| POR/XRS | Bypassed PLL.<br>PLL multiplier is set to 0x0.<br>Clock divider is set to /1. |
| All other Resets | Maintain clocks setup before device reset. |

**NOTE:** When the PLL is used by the bootloader, it is bypassed by the boot ROM code before branching to the user application.

### 4.6.12 Boot Status Information

Boot ROM keeps a record of the different events that can occur during boot ROM execution. This is because NMI and other exceptions are enabled by default in the device, and must be handled accordingly. Boot ROM stores the boot status information in a RAM location so that the user application can look at this boot status and take the necessary actions per the application's needs to handle these events.

#### 4.6.12.1 Booting Status

This section details the boot status RAM location and its bit field definitions. When the specific bit field is set, the described event or action has occurred.

**Table 4-39. Boot Status Address**

| Description | Address |
|---|---|
| BROM_STATUS | 0x0000 0002 |

**Table 4-40. Boot Status Bit Fields**

| Bit | Description |
|---|---|
| 22 | Boot ROM detected a missing clock NMI |
| 21 | Boot ROM detected a RAM bit error NMI |
| 20 | Boot ROM detected a Flash bit error NMI |
| 17 | Boot ROM detected a PIE mismatch |
| 16 | Boot ROM detected an ITRAP |
| 15 | Boot ROM has completed running |
| 13 | Boot ROM handled POR |
| 12 | Boot ROM handled XRS |
| 11 | Boot ROM handled all the resets |
| 10 | POR memory test has completed |
| 9 | DCSM initialization has completed |
| 8 | PLC boot has started |
| 7 | CAN boot has started |
| 6 | I2C boot has started |
| 5 | SPI boot has started |
| 4 | SCI boot has started |
| 3 | RAM boot has started |
| 2 | Parallel boot has started |
| 1 | Flash boot has started |
| 0 | Boot ROM has started running |

#### 4.6.12.2 Flash Single Bit Error Status

After DCSM is initialized during a reset by the boot ROM, the flash single bit error status is stored at the high and low RAM location detailed in this section. Both the high and low status are 32 bits wide.

**Table 4-41. Flash Single Bit Error Status Addresses**

| Description | Address |
|---|---|
| Flash Single Bit Error (Low) | 0x0000 0004 |
| Flash Single Bit Error (High) | 0x0000 0006 |

#### 4.6.12.3 PBIST Status

When PBIST is enabled to run, the status of the test is set within a status variable in boot ROM. This variable is reset on every run of boot ROM. If the user application requires the preservation of this status, the PBIST status must be copied to the application's RAM

### *4.6.13 ROM Version*

The ROM revision and release date information is stored at the ROM locations specified in this section.

**Table 4-42. Boot ROM Version Information**

| Start Address | End Address | Contents |
|---|---|---|
| 0x003F FF7A | 0x003F FF7B | Revision Number |
| 0x003F FF7C | 0x003F FF7D | Revision Date |

Interpreting the contents:

- Reading a revision number value of “0x100” represents version “1.0”.
- Reading a revision date value of “0x0715” represents “07/15” or “July 2015”.

## 4.7 Application Notes for Using the Bootloaders

### *4.7.1 The C2000 Hex Utility*

To use the features of the bootloader, you must generate a data stream and boot table as described in Bootloader Data Stream Structure. The hex conversion utility tool, included with the 28x code generation tools, can generate the required data stream including the required boot table. This section describes the hex2000 utility. An example of a file conversion performed by hex2000 is described in Example 4-2.

The hex utility supports creation of the boot table required for the SCI, SPI, I2C, CAN, and parallel I/O loaders. That is, the hex utility adds the required information to the file such as the key value, reserved bits, entry point, address, block start address, block length and terminating value. The contents of the boot table vary slightly depending on the boot mode and the options selected when running the hex conversion utility. The actual file format required by the host (ASCII, binary, hex, etc.) will differ from one specific application to another and some additional conversion may be required.

To build the boot table, follow these steps:

1. **Assemble or compile the code.**

   This creates the object files that will then be used by the linker to create a single output file.

2. **Link the file.**

   The linker combines all of the object files into a single output file in common object file format (COFF). The specified linker command file is used by the linker to allocate the code sections to different memory blocks. Each block of the boot table data corresponds to an initialized section in the COFF file. Uninitialized sections are not converted by the hex conversion utility. The following options may be useful:

   The linker -m option can be used to generate a map file. This map file will show all of the sections that were created, their location in memory and their length. It can be useful to check this file to make sure that the initialized sections are where you expect them to be.

   The linker -w option configures the linker to show if the linker assigned a section to a memory region automatically. For example, if you have a section in your code called ramfuncs.

3. **Run the hex conversion utility.**

   Choose the appropriate options for the desired boot mode and run the hex conversion utility to convert the COFF file produced by the linker to a boot table.

See the *TMS320C28x Assembly Language Tools User's Guide* (SPRU513) and the *TMS320C28x Optimizing C/C++ Compiler User's Guide* (SPRU514) for more information on the compiling and linking process.

Table 4-43 summarizes the hex conversion utility options available for the bootloader. See the *TMS320C28x Assembly Language Tools User's Guide* (SPRU513) for a detailed description of the hex2000 operations used to generate a boot table. Updates will be made to support the I2C boot. See the Codegen release notes for the latest information.

**Table 4-43. Boot Loader Options**

| Option | Description |
|---|---|
| -boot | Convert all sections into bootable form (use instead of a SECTIONS directive) |
| -sci8 | Specify the source of the bootloader table as the SCI-A port, 8-bit mode |

**Table 4-43. Boot Loader Options (continued)**

| Option | Description |
|---|---|
| -spi8 | Specify the source of the bootloader table as the SPI-A port, 8-bit mode |
| -gpio8 | Specify the source of the bootloader table as the GPIO port, 8-bit mode |
| -bootorg value | Specify the source address of the bootloader table |
| -lospcp value | Specify the initial value for the LOSPCP register. This value is used only for the spi8 boot table format and ignored for all other formats. If the value is greater than 0x7F, the value is truncated to 0x7F. |
| -spibrr value | Specify the initial value for the SPIBRR register. This value is used only for the spi8 boot table format and ignored for all other formats. If the value is greater than 0x7F, the value is truncated to 0x7F. |
| -e value | Specify the entry point at which to begin execution after boot loading. The value can be an address or a global symbol. This value is optional. The entry point can be defined at compile time using the linker -e option to assign the entry point to a global symbol. The entry point for a C program is normally _c_int00 unless defined otherwise by the -e linker option. |
| -i2c8 | Specify the source of the bootloader table as the I2C-A port, 8-bit |
| -i2cpsc value | Specify the value for the I2CPSC register. This value will be loaded and take effect after all I2C options are loaded, prior to reading data from the EEPROM. This value will be truncated to the least significant eight bits and should be set to maintain an I2C module clock of 7-12 MHz. |
| -i2cclkh value | Specify the value for the I2CCLKH register. This value will be loaded and take effect after all I2C options are loaded, prior to reading data from the EEPROM. |
| -i2cclkl value | Specify the value for the I2CCLKL register. This value will be loaded and take effect after all I2C options are loaded, prior to reading data from the EEPROM. |

### *Example 4-2. HEX2000.exe Command Syntax*

```
C: HEX2000 GPIO34TOG.OUT -boot -gpio8 -a


Where:
- boot   Convert all sections into bootable form.
- gpio8  Use the GPIO in 8-bit mode data format. The eCAN
         uses the same data format as the GPIO in 8-bit mode.
- a      Select ASCII-Hex as the output format.
```

# Control Law Accelerator (CLA)

The control law accelerator (CLA) Type-2 is an independent, fully-programmable, 32-bit floating-point math processor that brings concurrent control-loop execution to the C28x family. The low interrupt latency of the CLA allows it to read ADC samples "just-in-time." This significantly reduces the ADC sample to output delay to enable faster system response and higher MHz control loops. By using the CLA to service time-critical control loops, the main CPU is free to perform other system tasks such as communications and diagnostics. This chapter provides an overview of the architectural structure and components of the control law accelerator.

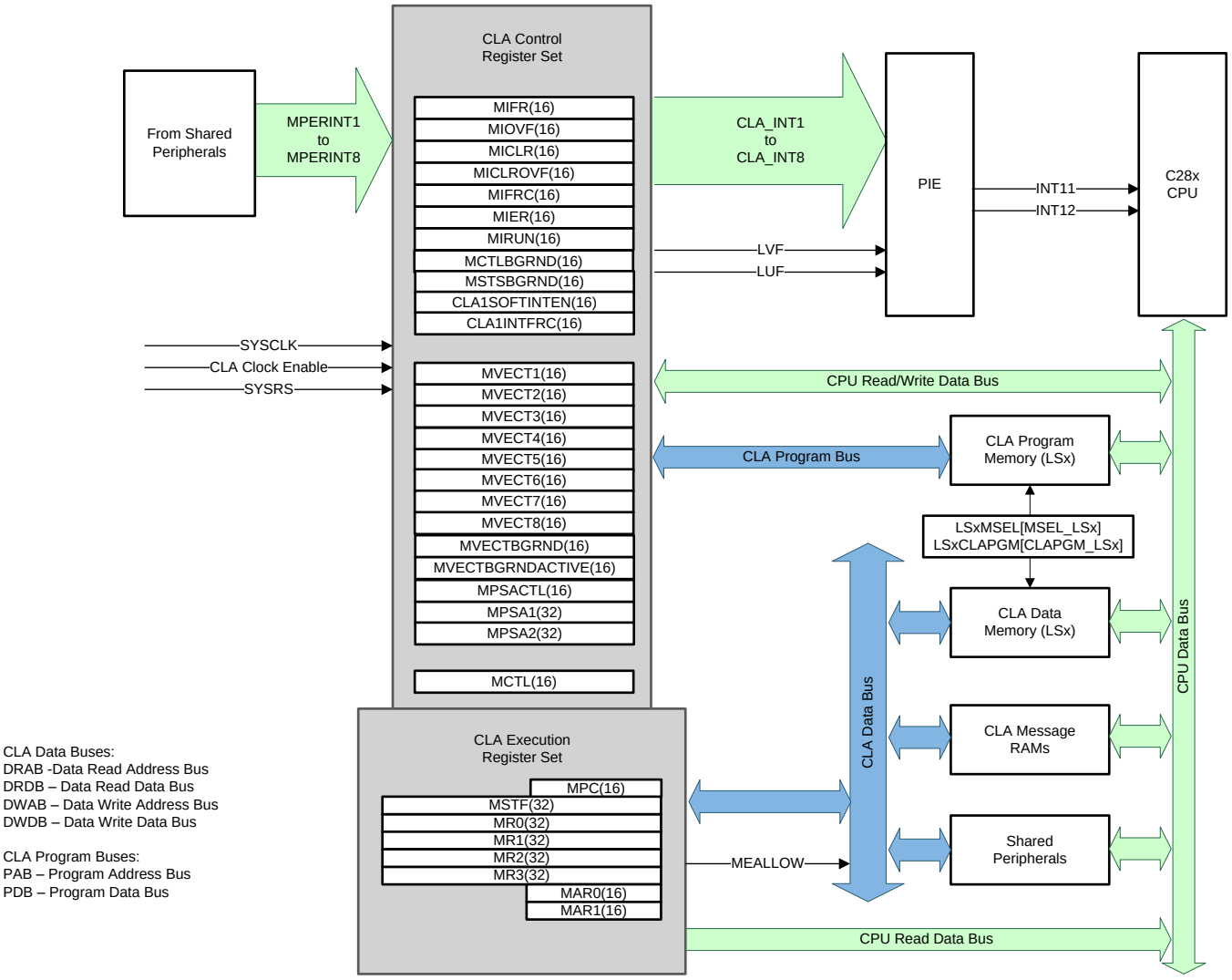
**Topic** **Page**

## 5.1 Control Law Accelerator (CLA) Overview

The control law accelerator extends the capabilities of the C28x CPU by adding parallel processing. Time-critical control loops serviced by the CLA can achieve low ADC sample to output delay. Thus, the CLA enables faster system response and higher frequency control loops. Utilizing the CLA for time-critical tasks frees up the main CPU to perform other system and communication functions concurrently. The following is a list of major features of the CLA.

- Clocked at the same rate as the main CPU (SYSCLKOUT).
- An independent architecture allowing CLA algorithm execution independent of the main C28x CPU.
  - Complete bus architecture:
    - Program Address Bus (PAB) and Program Data Bus (PDB)
    - Data Read Address Bus (DRAB), Data Read Data Bus (DRDB), Data Write Address Bus (DWAB), and Data Write Data Bus (DWDB)
  - Independent eight stage pipeline.
  - 16-bit program counter (MPC)
  - Four 32-bit result registers (MR0-MR3)
  - Two 16-bit auxiliary registers (MAR0, MAR1)
  - Status register (MSTF)
- Instruction set includes:
  - IEEE single-precision (32-bit) floating point math operations
  - Floating-point math with parallel load or store
  - Floating-point multiply with parallel add or subtract
  - 1/X and 1/sqrt(X) estimations
  - Data type conversions.
  - Conditional branch and call
  - Data load/store operations
- The CLA program code can consist of up to eight tasks or interrupt service routines, or 7 tasks and a main background task.
  - The start address of each task is specified by the MVECT registers.
  - No limit on task size as long as the tasks fit within the configurable CLA program memory space.
  - One task is serviced at a time until its completion. There is no nesting of tasks.
  - Upon task completion a task-specific interrupt is flagged within the PIE.
  - When a task finishes the next highest-priority pending task is automatically started.
  - The Type-2 CLA can have a main task that runs continuously in the background, while other high priority events trigger a foreground task.
- Task trigger mechanisms:
  - C28x CPU via the IACK instruction
  - Task1 to Task8: up to 256 possible trigger sources from peripherals connected to the shared bus on which the CLA assumes secondary ownership.
  - Task8 can be set to be the background task, while Tasks 1 through 7 take peripheral triggers.
- Memory and Shared Peripherals:
  - Two dedicated message RAMs for communication between the CLA and the main CPU.
  - The C28x CPU can map CLA program and data memory to the main CPU space or CLA space.
  - The CLA, on reset, is the secondary master for all peripherals which can have either the CLA or DMA as their secondary master.

**Figure 5-1. CLA Block Diagram**



## 5.2 CLA Interface

This chapter describes how the C28x main CPU can interface to the CLA and vice versa.

### 5.2.1 CLA Memory

The CLA can access three types of memory: program, data and message RAMs. The behavior and arbitration for each type of memory is described in detail below.

- **CLA Program Memory**

  The CLA program can be loaded to any of the local shared memories (LSxRAM) on the core that it is tied to. At reset, all memory blocks are mapped to the master CPU. While mapped to the CPU space, the CPU can copy the CLA program code into the memory block(s). During debug, the block(s) can also be loaded directly by Code Composer Studio™.

  Once the memory is initialized with CLA code, the master CPU maps it to the CLA program space by:

  1. Assigning ownership of the memory block to the CLA by writing a 1 to the memory block's MemCfgRegs.LSxMSEL[MSEL_LSx] bit.
  2. Specifying the memory block as a code block for the CLA by writing a 1 to the MemCfgRegs.LSxCLAPGM[CLAPGM_LSx] bit.

  When an LSx memory is configured as CLA program memory, only debug accesses are allowed on cycles that the CLA is not fetching a new instruction. A detailed explanation of the memory configurations and access arbitration (CPU/CLA/DEBUG) process can be found in Section 3.11.

  All CLA program fetches are performed as 32-bit read operations and all opcodes must be aligned to an even address. Since all CLA opcodes are 32-bits, this alignment naturally occurs.

- **CLA Data Memory**

  Any of the device's LSxRAMs can serve as data memory blocks to the CLA. At reset, all blocks are mapped to the master CPU memory space, whereby the master can initialize the memory with data tables, coefficients, and so on, for the CLA to use.

  Once the memory is initialized with CLA data, the master CPU maps it to the CLA data space by :

  1. Assigning ownership of the memory block to the CLA by writing a 1 to the memory block's MemCfgRegs.LSxMSEL[MSEL_LSx] bit.
  2. Specifying the memory block as a data block for the CLA by writing a 0 to the MemCfgRegs.LSxCLAPGM[CLAPGM_LSx] bit. The value of this bit at reset is 0.

  When an LSx memory is configured as a CLA data memory, the CLA read/write access are arbitrated along with CPU accesses. The user has the option of turning on CPU fetch or write protection to the memory by writing to the appropriate bits of the MemCfgRegs.LSxACCPROTx registers. A detailed explanation of the memory configurations and access arbitration (CPU/CLA/DEBUG) process can be found in Section 3.11.

  The CLA RAMs and registers are protected by the security module. Refer to the *DCSM* chapter of this manual for more details on the security scheme.

- **CLA Shared Message RAMs**

  There are two small memory blocks for data sharing and communication between the CLA and the master CPU on each CPU subsystem. The message RAMs are always mapped to both CPU and CLA memory spaces and are protected by the DCSM. The message RAMs allow data accesses only; no program fetches can be performed.

  – **CLA to CPU Message RAM**
    The CLA can use this block to pass data to the master CPU. This block is both readable and writable by the CLA. This block is also readable by the master CPU but writes by the master CPU are ignored

  – **CPU to CLA Message RAM**
    The master CPU can use this block to pass data and messages to the CLA. This message RAM is both readable and writable by the master CPU. The CLA can perform reads but writes by the CLA are ignored.

### 5.2.2 CLA Memory Bus

The CLA has dedicated bus architecture similar to that of the C28x CPU where there is a program read, data read, and data write bus. Thus, there can be simultaneous instruction fetch, data read, and data write in a single cycle. Like the C28x CPU, the CLA expects memory logic to align any 32-bit read or write to an even address. If the address-generation logic generates an odd address, the CLA will begin reading or writing at the previous even address. This alignment does not affect the address values generated by the address-generation logic.

- **CLA Program Bus**

  The CLA program bus has a access range of 32K 32-bit instructions. Since all CLA instructions are 32 bits, this bus always fetches 32 bits at a time and the opcodes must be even-word aligned. The amount of program space available for the CLA is limited to the number of available LSxRAM blocks. This number is device-dependent and will be described in the device-specific data manual.

- **CLA Data Read Bus**

  The CLA data read bus has a 64K x 16 address range. The bus can perform 16 or 32-bit reads and will automatically stall if there are memory access conflicts. The data read bus has access to both the message RAMs, CLA data memory, and the shared peripherals.

- **CLA Data Write Bus**

  The CLA data write bus has a 64K x 16 address range. This bus can perform 16 or 32-bit writes. The bus will automatically stall if there are memory access conflicts. The data write bus has access to the CLA to CPU message RAM, CLA data memory, and the shared peripherals.

### 5.2.3 Shared Peripherals and EALLOW Protection

For a given CPU subsystem, the CPU, CLA, and DMA can share access to some peripherals. There is a 3-way arbitration among the different master's that is described in Section 5.3. Each peripheral has an access control register with two bit fields, CPUnAC, CLAnAC, and DMAnAC (n being the instance) that determine what kind of access is given to that particular master.

**Note:** The CLA read access time to the bus is 2-wait states while write access is 0-wait.

Refer to the device data manual for the list of peripherals connected to the bus.

Several peripheral control registers are protected from spurious 28x CPU writes by the EALLOW protection mechanism. These same registers are also protected from spurious CLA writes. The EALLOW bit in the main CPU status register 1 (ST1) indicates the state of protection for the main CPU. Likewise the MEALLOW bit in the CLA status register (MSTF) indicates the state of write protection for the CLA. The MEALLOW CLA instruction enables write access by the CLA to EALLOW protected registers. Likewise the MEDIS CLA instruction will disable write access. This way the CLA can enable/disable write access independent of the main CPU.

The ADC offers the option to generate an early interrupt pulse at the start of a sample conversion. If this option is used to start an ADC-triggered CLA task, the user may use the intervening cycles, until the completion of the conversion, to perform preliminary calculations or loads and stores before finally reading the ADC value. The CLA pipeline activity for this scenario is shown in Section 5.5.

### 5.2.4 CLA Tasks and Interrupt Vectors

The CLA program code is divided up into tasks or interrupt service routines. Tasks do not have a fixed starting location or length. The CLA program memory can be divided up as desired. The CLA knows where a task begins by the content of the associated interrupt vector (MVECT1 to MVECT8) and the end is indicated by the MSTOP instruction.

The CLA supports eight tasks. Task 1 has the highest priority and task 8 has the lowest priority. The Type-2 CLA offers the option of setting the lowest priority task, for example, Task 8, as a background task that, once triggered, runs continuously until the user either terminates it or resets the CLA or the device. The remaining tasks, 1 through 7, maintain their priority levels and interrupt the background task when triggered.

The background task is enabled by setting the BGEN bit in the MCTLBGRND register; this causes the hardware to disable task 8 in the MIER register. The background task derives its interrupt vector from the MVECTBGRND register instead of MVECT8.

A task can be requested by a peripheral interrupt or by software:

- **Peripheral interrupt trigger**

  Each task can be triggered by up to 256 interrupt sources. The trigger for each task is defined by writing an appropriate value to the DmaClaSrcSelRegs.CLA1TASKSRCSELx[TASKx] bit field. Each of the possible 256 values specifies an interrupt source from a specific peripheral on the shared bus. The configuration options are listed in Table 5-1.

**Table 5-1. Configuration Options**

| Select Value (8-bit) | CLA Trigger Source |
|---|---|
| 0 | Software Trigger |
| 1 | ADCA1_INT |
| 2 | ADCA2_INT |
| 3 | ADCA3_INT |
| 4 | ADCA4_INT |
| 5 | ADCA_EVT_INT |
| 6 | ADCB1_INT |
| 7 | ADCB2_INT |
| 8 | ADCB3_INT |
| 9 | ADCB4_INT |
| 10 | ADCB_EVT_INT |
| 11 | ADCC1_INT |
| 12 | ADCC2_INT |
| 13 | ADCC3_INT |
| 14 | ADCC4_INT |
| 15 | ADCC_EVT_INT |
| 28:16 | Reserved |
| 29 | XINT1 |
| 30 | XINT2 |
| 31 | XINT3 |
| 32 | XINT4 |
| 33 | XINT5 |
| 35:34 | Reserved |
| 36 | EPWM1INT |
| 37 | EPWM2INT |
| 38 | EPWM3INT |
| 39 | EPWM4INT |
| 40 | EPWM5INT |
| 41 | EPWM6INT |

**Table 5-1. Configuration Options (continued)**

| Select Value (8-bit) | CLA Trigger Source |
|---|---|
| 42 | EPWM7INT |
| 43 | EPWM8INT |
| 67:44 | Reserved |
| 68 | TINT0 |
| 69 | TINT1 |
| 70 | TINT2 |
| 71 | Reserved |
| 72 | Reserved |
| 73 | Reserved |
| 74 | Reserved |
| 75 | ECAP1INT |
| 76 | ECAP2INT |
| 77 | ECAP3INT |
| 78 | ECAP4INT |
| 79 | ECAP5INT |
| 80 | ECAP6INT |
| 81 | ECAP7INT |
| 82 | Reserved |
| 83 | EQEP1INT |
| 84 | EQEP2INT |
| 91:85 | Reserved |
| 92 | ECAP6INT2 |
| 93 | ECAP7INT2 |
| 94 | Reserved |
| 95 | SD1INT |
| 96 | SD1DRINT1 |
| 97 | SD1DRINT2 |
| 98 | SD1DRINT3 |
| 99 | SD1DRINT4 |
| 104:100 | Reserved |
| 105 | PMBUSAINT |
| 108:106 | Reserved |
| 109 | SPITXINTA |
| 110 | SPIRXINTA |
| 111 | SPITXINTB |
| 112 | SPIRXINTB |
| 116:113 | Reserved |
| 117 | LINA_INT1 |
| 118 | LINA_INT0 |
| 120:119 | Reserved |
| 121 | CLA1PROMCRC |
| 122 | Reserved |
| 123 | FSITX_INT1 |
| 124 | FSITX_INT2 |
| 125 | FSIRX_INT1 |
| 126 | FSIRX_INT2 |
| 127 | CLB1INT |

**Table 5-1. Configuration Options (continued)**

| Select Value (8-bit) | CLA Trigger Source |
|---|---|
| 128 | CLB2INT |
| 129 | CLB3INT |
| 130 | CLB4INT |
| 255:131 | Reserved |

For example, task 1 (MVECT1) can be set to trigger on EPWMINT1 by writing 36 to DmaClaSrcSelRegs.CLA1TASKSRCSEL1.bit.TASK1. To disable the triggering of a task by a peripheral, the user must set the DmaClaSrcSelRegs.CLA1TASKSRCSELx[TASKx] bit field to 0. It should be noted that a CLA task only triggers on a level transition (an edge) of the configured interrupt source.

- **Software Trigger**

  Tasks can also be started by the main CPU software writing to the MIFRC register or by the IACK instruction. Using the IACK instruction is more efficient because it does not require you to issue an EALLOW to set MIFR bits. Set the MCTL[IACKE] bit to enable the IACK feature. Each bit in the operand of the IACK instruction corresponds to a task. For example IACK #0x0001 will set bit 0 in the MIFR register to start task 1. Likewise IACK #0x0003 will set bits 0 and 1 in the MIFR register to start task 1 and task 2.

- **Background Task**

  The Type-2 CLA allows the user to use Task 8 as a background task that runs continuously until the user disables it or resets the device (or the CLA using a soft reset). The background task vector is given by the MVECTBGRND register and its operation controlled by the MCTLBGRND register; it is enabled by setting the BGEN bit to 1. The user can then kick off the task through software by writing a 1 to the BGSTART bit (TRIGEN must be 0), or through a peripheral by setting the TRIGEN bit to 1 and then setting the trigger source in the bit-field, DmaClaSrcSelRegs.CLA1TASKSRCSEL2.bit.TASK8. By default the background task is interruptible; the highest priority pending task will be executed first. When a task completes, and there aren't any pending tasks, the execution returns to the background task. The CLA keeps track of the branching point by saving the return address to the MVECTBGRNDACTIVE register, and then popping this address to the MPC when execution returns. The user may choose to make sections of the background task un-interruptible; it is possible to do this with the MSETC BGINTM assembly instruction.

  Subsequently, enabling interrupts with the MCLRC BGINTM instruction.

  The background interrupt mask bit, BGINTM, can be queried in the MSTSBGRND register. This register also provides the current status of the background task; if it is currently executing, the RUN bit is set to 1, if another trigger for the background task is received while it has already started the overflow (BGOVF) bit is set.

The CLA has its own fetch mechanism and can run and execute a task independent of the main CPU. Only one task is serviced at a time; there is no nesting of tasks unless the background task in enabled, then one level of nesting is possible. The task currently running is indicated in the MIRUN register; if the background task is enabled, and running, it is reflected in the MSTSBGRND register (the RUN bit).

Interrupts that have been received but not yet serviced are indicated in the flag register (MIFR). If an interrupt request from a peripheral is received and that same task is already flagged, then the overflow flag bit is set. Overflow flags will remain set until they are cleared by the main CPU. If the CLA is idle (no task is currently running) or is executing the background task, then the highest priority interrupt request that is both flagged (MIFR) and enabled (MIER) will start. The flow is as follows:

1. The associated RUN register bit is set (MIRUN) and the flag bit (MIFR) is cleared.
2. The CLA begins execution at the location indicated by the associated interrupt vector (MVECTx). MVECT contains the absolute 16-bit address of the task in the lower 64K memory space. If a task is interrupting the background task then the current program address is stored in the MVECTBGRNDACTIVE register before execution jumps to the task; this saved address is restored to the MPC when the task completes and execution returns to the background task.
3. The CLA executes instructions until the MSTOP instruction is found. This indicates the end of the task.
4. The MIRUN bit is cleared.

5. The task-specific interrupt to the PIE is issued. This informs the main CPU that the task has completed.
6. The CLA returns to idle (or to the background task, if enabled). Once a task completes the next highest-priority pending task is automatically serviced and this sequence repeats.

### 5.2.5 CLA Software Interrupt to CPU

The CLA can issue a software interrupt to the C28x CPU (on the same subsystem) at any point in the code through the use of the CLA1SOFTINTEN and CLA1INTFRC registers. Please see Section 5.7 for a description of these registers. If a software interrupt is selected for a CLA task, then an end-of-task interrupt will not be issued to the C28x when that task completes.

## 5.3 CLA, DMA, and CPU Arbitration

Typically, CLA activity is independent of the CPU activity. Under the circumstance where CLA, DMA or CPU are attempting to access memory or a peripheral register within the same interface concurrently, an arbitration procedure will occur. This section describes this arbitration.

### 5.3.1 CLA, DMA, and CPU Arbitration

The Local Shared RAMs can have multiple masters, that is, the CPU, CLA and the DMA. The arbitration follows a fixed arbitration scheme:

- DMA WRITE
- DMA READ
- CLA WRITE
- CLA READ
- CPU WRITE
- CPU READ

They are covered in detail in the Section 3.11.

### 5.3.2 CLA Message RAM

Message RAMs consist of two blocks

- CLA1 to CPU – CLA to CPU Message RAM
- CPU to CLA1 – CPU to CLA Message RAM

These blocks are for passing data between the CPU and the CLA. No opcode fetches, from either the CLA or CPU, are allowed from the message RAMs. A write protection violation will not be generated if the CLA attempts to write to the CPU to CLA1 message RAM but the write will be ignored. The arbitration scheme for the message RAMs are the same as those for the shared memories described in the Section 3.11.

The two message RAMs have the following characteristics:

- CPU.CLA1 to CPU Message RAM:
  The following accesses are allowed:
  - CPU reads
  - CPU.CLA1 data reads and writes
  - CPU debug reads and writes

  The following accesses are ignored:
  - CPU writes
- CPU to CLA1 Message RAM:
  The following accesses are allowed:
  - CPU reads and writes
  - CPU.CLA1 reads

– CPU debug reads and writes

The following accesses are ignored:

– CPU.CLA1 writes

## 5.4 CLA Configuration and Debug

This section discusses the steps necessary to configure and debug the CLA.

### 5.4.1 Building a CLA Application

The control law accelerator can be programmed in either CLA assembly code, using the instructions described in Section 5.6, or a reduced subset of the C language. CLA assembly code resides in the same project with C28x code. The only restriction is the CLA code must be in its own assembly section. This can be easily done using the .sect assembly directive. This does not prevent CLA and C28x code from being linked into the same memory region in the linker command file.

System and CLA initialization are performed by the main CPU. This would typically be done in C or C++ but can also include C28x assembly code. The main CPU will also copy the CLA code to the program memory and, if needed, initialize the CLA data RAM(s). Once system initialization is complete and the application begins, the CLA will service its interrupts using the CLA assembly code (or tasks). Concurrently the main CPU can perform other tasks.

The CLA type 1 requires Codegen V6.2.4 or later with the following switch: --cla_support=cla1.

### 5.4.2 Typical CLA Initialization Sequence

A typical CLA initialization sequence is performed by the main CPU as described in this section.

1. **Copy CLA code into the CLA program RAM**

   The source for the CLA code can initially reside in the flash or a data stream from a communications peripheral or anywhere the main CPU can access it. The debugger can also be used to load code directly to the CLA program RAM during development.

2. **Initialize CLA data RAM, if necessary**

   Populate the CLA data RAM with any required data coefficients or constants.

3. **Configure the CLA registers**

   Configure the CLA registers, but keep interrupts disabled until later (leave MIER = 0):

   - **Enable the CLA clock in the PCLKCR3 register**

     PCLKCR3 register is defined in the *System Control and Interrupts* chapter.

   - **Populate the CLA task interrupt vectors**
     - MVECT1 to MVECT8

       Each vector needs to be initialized with the start address of the task to be executed when the CLA receives the associated interrupt. This address is the full 16-bit starting address of the task in the lower 64K section of memory.

     - MVECT1 to MVECT7, and MVECTBGRND

       When using the background task, its vector (MVECTBGRND) must be loaded with the start address of the task in lower 64K of memory. Note that Task 8 is ignored when the background task is enabled

   - **Select the task interrupt sources**

     For each task select the interrupt source in the CLA1TASKSRCSELx register. If a task is going to be generated by software, select no interrupt. Since the background task takes the place of Task 8, it will use the same peripheral trigger source as task 8.

   - **Enable IACK to start a task from software, if desired**

     To enable the IACK instruction to start a task set the MCTL[IACKE] bit. Using the IACK instruction avoids having to set and clear the EALLOW bit. If the background task is enabled, the IACK bit for task 8 is ignored; the user must, instead, write to the BGSTART bit of the MCTLBGRND register to start the background task (TRIGEN should be 0 to avoid a peripheral trigger from causing an overflow, for example, MSTSBGRND.BGOVF is set to 1).

   - **Map CLA data RAM(s) to CLA space, if necessary**

     Map the data RAM to the CLA space by first, assigning ownership of the memory block to the CLA by writing a 1 to the memory block's MemCfgRegs.LSxMSEL[MSEL_LSx] bit, and then specifying

the memory block as a CLA data block by writing a 0 to the MemCfgRegs.LSxCLAPGM[CLAPGM_LSx] bit. When an LSx memory is configured as a CLA data memory, the CLA read/write 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 will be flagged. The CLA overflow and underflow flags also have associated interrupts within the PIE.

5. **Enable CLA tasks/interrupts**

   Set appropriate bits in the interrupt enable register (MIER) to allow the CLA to service interrupts.

6. **Initialize other peripherals**

   Initialize any peripherals (ePWM, ADC, and others) that will generate an interrupt to the CLA and be serviced by a CLA task.

   The CLA is now ready to service interrupts and the message RAMs can be used to pass data between the CPU and the CLA. Typically mapping of the CLA program and data RAMs occurs only during the initialization process. If after some time you want to re-map these memories back to CPU space, then disable interrupts and make sure all tasks have completed by checking the MIRUN register.

### 5.4.3 Debugging CLA Code

Debugging the CLA code is a simple process that occurs independently of the main CPU. The type 2 CLA adds a true software breakpoint feature.

#### 5.4.3.1 Software Breakpoint Support

A new instruction, MDEBUGSTOP1, has been added. It is meant to be used as a software breakpoint; the instruction at which the execution must halt is replaced with this instruction. It differs from MDEBUGSTOP in that it flushes all the instructions that have already been fetched, and on a step, or run free, command will re-fetch the same instruction which it replaced. Table 5-2 illustrates the concept.

**Table 5-2. Pipeline Behavior of the MDEBUGSTOP1 Instruction**

| Cycles | F1 | F2 | D1 | D2 | R1 | R2 | E | W | Comments |
|---|---|---|---|---|---|---|---|---|---|
| 1 | i1 | | | | | | | | |
| 2 | i2 | i1 | | | | | | | |
| 3 | i3 | i2 | i1 | | | | | | |
| 4 | i4 | i3 | i2 | i1 | | | | | |
| 5 | MDEBUGSTOP1 | i4 | i3 | i2 | i1 | | | | |
| 6 | i6 | MDEBUGSTOP1 | i4 | i3 | i2 | i1 | | | |
| 7 | i7 | i6 | MDEBUGSTOP1 | i4 | i3 | i2 | i1 | | |
| 9 | i8 | Flushed (MNOP) | Flushed (MNOP) | Flushed (MNOP) | i4 | i3 | i2 | i1 | CLA halted |
| 10 | i5(MDEBUGSTOP1) | Flushed (MNOP) | Flushed (MNOP) | Flushed (MNOP) | Flushed (MNOP) | i4 | i3 | i2 | CLA step/run |
| 11 | i6 | i5(MDEBUGSTOP1) | Flushed (MNOP) | Flushed (MNOP) | Flushed (MNOP) | Flushed (MNOP) | i4 | i3 | CLA step/run |
| 12 | i7 | i6 | i5(MDEBUGSTOP1) | Flushed (MNOP) | Flushed (MNOP) | Flushed (MNOP) | Flushed (MNOP) | i4 | CLA step/run |
| 13 | i8 | i7 | i6 | i5(MDEBUGSTOP1) | Flushed (MNOP) | Flushed (MNOP) | Flushed (MNOP) | Flushed (MNOP) | CLA step/run |

A software breakpoint is placed at instruction i5. The instruction, i5, is then replaced with MDEBUGSTOP1. It takes 3 cycles for the MDEBUGSTOP1 to reach the D2 phase at which point the instructions i6, i7, and i8 that were previously fetched are now flushed from the pipeline. The instruction, i5, is then re-fetched and execution continues as before.

#### 5.4.3.2 Legacy Breakpoint

1. **Insert a breakpoint in CLA code**

   Insert a CLA breakpoint (MDEBUGSTOP instruction) into the code where you want the CLA to halt, then rebuild and reload the code. Because the CLA does not flush its pipeline when you single-step, the MDEBUGSTOP instruction must be inserted as part of the code. The debugger cannot insert it as needed.

   If CLA breakpoints are not enabled, then the MDEBUGSTOP will be ignored and is treated as a MNOP. The MDEBUGSTOP instruction can be placed anywhere in the CLA code as long as it is not within three instructions of a MBCNDD, MCCNDD, or MRCNDD instruction. When programming in C, the user can use the __mdebugstop() intrinsic instead; the compiler will ensure that the placement of the MDEBUSTOP instruction in the generated assembly does not violate any of the pipeline restrictions.

2. **Enable CLA breakpoints**

   Enable the CLA breakpoints in the debugger. In Code Composer Studio, this is done by connecting to the CLA core (or tap) from the debug perspective. Breakpoints are disabled when the core is disconnected.

3. **Start the task**

   There are three ways to start the task:

   - The peripheral can assert an interrupt
   - The main CPU can execute an IACK instruction, or
   - The user can manually write to the MIFRC register in the debugger window

   When the task starts, the CLA will execute instructions until the MDEBUGSTOP is in the D2 phase of the pipeline. At this point, the CLA will halt and the pipeline will be frozen. The MPC register will reflect the address of the MDEBUGSTOP instruction.

4. **Single-step the CLA code**

   Once halted, the user can single-step the CLA code. The behavior of a CLA single-step is different than the main C28x. When issuing a CLA single-step, the pipeline is clocked only one cycle and then again frozen. On the 28x CPU, the pipeline is flushed for each single-step.

   You can also run to the next MDEBUGSTOP or to the end of the task. If another task is pending, it will automatically start when you run to the end of the task.

   **NOTE:** A CLA fetch has higher priority than CPU debug reads. For this reason, it is possible for the CLA to permanently block CPU debug accesses if the CLA is executing in a loop. This might occur when initially developing CLA code due to a bug that causes an infinite loop. To avoid locking up the main CPU, the program memory will return all 0x0000 for CPU debug reads when the CLA is running. When the CLA is halted or idle then normal CPU debug read and write access to CLA program memory can be performed.

   If the CLA gets caught in an infinite loop, you can use a soft or hard reset to exit the condition. A debugger reset will also exit the condition.

   There are special cases that can occur when single-stepping a task such that the program counter, MPC, reaches the MSTOP instruction at the end of the task.

   - **MPC halts at or after the MSTOP with a task already pending**

     If you are single-stepping or halted in "task A" and "task B" comes in before the MPC reaches the MSTOP, then "task B" will start if you continue to step through the MSTOP instruction. Basically if "task B" is pending before the MPC reaches MSTOP in "task A" then there is no issue in "task B" starting and no special action is required.

   - **MPC halts at or after the MSTOP with no task pending**

     In this case you have single-stepped or halted in "task A" and the MPC has reached the MSTOP with no tasks pending. If "task B" comes in at this point, it will be flagged in the MIFR register but it may or may not start if you continue to single-step through the MSTOP instruction of "task A."

     It depends on exactly when the new task comes in. To reliably start "task B" perform a soft reset and reconfigure the MIER bits. Once this is done, you can start single-stepping "task B."

     This case can be handled slightly differently if there is control over when "task B" comes in (for example using the IACK instruction to start the task). In this case you have single-stepped or halted in "task A" and the MPC has reached the MSTOP with no tasks pending. Before forcing "task B," run free to force the CLA out of the debug state. Once this is done you can force "task B" and continue debugging.

5. **Disable CLA breakpoints, if desired**

   In Code Composer Studio you can disable the CLA breakpoints by disconnecting the CLA core in the debug perspective. Make sure to first issue a run or reset; otherwise, the CLA will be halted and no other tasks will start.

### *5.4.4 CLA Illegal Opcode Behavior*

If the CLA fetches an opcode that does not correspond to a legal instruction, it will behave as follows:

- The CLA will halt with the illegal opcode in the D2 phase of the pipeline as if it were a breakpoint. This will occur whether CLA breakpoints are enabled or not.
- The CLA will issue the task-specific interrupt to the PIE.

- The MIRUN bit for the task will remain set.

Further single-stepping is ignored once execution halts due to an illegal op-code. To exit this situation, issue either a soft or hard reset of the CLA as described in Section 5.4.5.

### 5.4.5 Resetting the CLA

There may be times when you need to reset the CLA. For example, during code debug the CLA may enter an infinite loop due to a code bug. The CLA has two types of resets: hard and soft. Both of these resets can be performed by the debugger or by the main CPU.

- **Hard Reset**

  Writing a 1 to the MCTL[HARDRESET] bit will perform a hard reset of the CLA. The behavior of a hard reset is the same as a system reset (via $\overline{\text{XRS}}$ or the debugger). In this case all CLA configuration and execution registers will be set to their default state and CLA execution will halt.

- **Soft Reset**

  Writing a 1 to the MCTL[SOFTRESET] bit performs a soft reset of the CLA. If a task is executing it will halt and the associated MIRUN bit will be cleared. All bits within the interrupt enable (MIER) register will also be cleared so that no new tasks start. In addition, the background task's start bit (MCTLBGRN.BGSTART) and Trigger Enable bit (MCTLBGRND.TRIGEN) are reset. The MVECTBGRNACTIVE is set to the value of MVECTBGRND, and the status register (MSTSBGRND) is also reset.

## 5.5 Pipeline

This section describes the CLA pipeline stages and presents cases where pipeline alignment must be considered.

### 5.5.1 Pipeline Overview

The CLA pipeline is very similar to the C28x pipeline. The pipeline has eight stages:

- **Fetch 1 (F1)**

  During the F1 stage the program read address is placed on the CLA program address bus.
- **Fetch 2 (F2)**

  During the F2 stage the instruction is read using the CLA program data bus.
- **Decode 1 (D1)**

  During D1 the instruction is decoded.
- **Decode 2 (D2)**

  Generate the data read address. Changes to MAR0 and MAR1 due to post-increment using indirect addressing takes place in the D2 phase. Conditional branch decisions are also made at this stage based on the MSTF register flags.
- **Read 1 (R1)**

  Place the data read address on the CLA data-read address bus. If a memory conflict exists, the R1 stage will be stalled.
- **Read 2 (R2)**

  Read the data value using the CLA data read data bus.
- **Execute (EXE)**

  Execute the operation. Changes to MAR0 and MAR1 due to loading an immediate value or value from memory take place in this stage.
- **Write (W)**

  Place the write address and write data on the CLA write data bus. If a memory conflict exists, the W stage will be stalled.

### 5.5.2 CLA Pipeline Alignment

The majority of the CLA instructions do not require any special pipeline considerations. This section lists the few operations that do require special consideration.

- **Write Followed by Read**

  In both the C28x and the CLA pipeline the read operation occurs before the write. This means that if a read operation immediately follows a write, then the read will complete first as shown in Table 5-3. In most cases this does not cause a problem since the contents of one memory location does not depend on the state of another. For accesses to peripherals where a write to one location can affect the value in another location the code must wait for the write to complete before issuing the read as shown in Table 5-4.

  This behavior is different for the 28x CPU. For the 28x CPU any write followed by read to the same location is protected by what is called write-followed-by-read protection. This protection automatically stalls the pipeline so that the write will complete before the read. In addition some peripheral frames are protected such that a 28x CPU write to one location within the frame will always complete before a read to the frame. The CLA does not have this protection mechanism. Instead the code must wait to perform the read.

**Table 5-3. Write Followed by Read - Read Occurs First**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| I1 MMOV16 @Reg1, MR3 | I1 | | | | | | | |
| I2 MMOV16 MR2, @Reg2 | I2 | I1 | | | | | | |
| | | I2 | I1 | | | | | |
| | | | I2 | I1 | | | | |
| | | | | I2 | I1 | | | |
| | | | | | I2 | I1 | | |
| | | | | | | I2 | I1 | |
| | | | | | | | I2 | I1 |

**Table 5-4. Write Followed by Read - Write Occurs First**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| I1 MMOV16 @Reg1, MR3 | I1 | | | | | | | |
| I2 | I2 | I1 | | | | | | |
| I3 | I3 | I2 | I1 | | | | | |
| I4 | I4 | I3 | I2 | I1 | | | | |
| I5 MMOV16 MR2, @Reg2 | I5 | I4 | I3 | I2 | I1 | | | |
| | | I5 | I4 | I3 | I2 | I1 | | |
| | | | I5 | I4 | I3 | I2 | I1 | |
| | | | | I5 | I4 | I3 | I2 | I1 |
| | | | | | I5 | I4 | I3 | |
| | | | | | | I5 | I4 | |
| | | | | | | | I5 | |

- **Delayed Conditional instructions: MBCNDD, MCCNDD and MRCNDD**

  Referring to Example 5-1, the following applies to delayed conditional instructions:

  – **I1**

    I1 is the last instruction that can effect the CNDF flags for the branch, call or return instruction. The CNDF flags are tested in the D2 phase of the pipeline. That is, a decision is made whether to branch or not when MBCNDD, MCCNDD or MRCNDD is in the D2 phase.

  – **I2, I3 and I4**

    The three instructions preceding MBCNDD can change MSTF flags but will have no effect on whether the MBCNDD instruction branches or not. This is because the flag modification will occur after the D2 phase of the branch, call or return instruction. These three instructions must not be a MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD or MRCNDD.

  – **I5, I6 and I7**

    The three instructions following a branch, call or return are always executed irrespective of whether the condition is true or not. These instructions must not be MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD or MRCNDD.

  For a more detailed description refer to the functional description for MBCNDD, MCCNDD and MRCNDD.

***Example 5-1. Code Fragment For MBCNDD, MCCNDD or MRCNDD***

```
<Instruction 1>    ; I1 Last instruction that can affect flags for
                   ;     the branch, call or return operation

<Instruction 2>    ; I2 Cannot be stop, branch, call or return
<Instruction 3>    ; I3 Cannot be stop, branch, call or return
<Instruction 4>    ; I4 Cannot be stop, branch, call or return

<branch/call/ret>  ; MBCNDD, MCCNDD or MRCNDD

                   ; I5-I7: Three instructions after are always
                   ; executed whether the branch/call or return is
                   ; taken or not

<Instruction 5>    ; I5 Cannot be stop, branch, call or return
<Instruction 6>    ; I6 Cannot be stop, branch, call or return
<Instruction 7>    ; I7 Cannot be stop, branch, call or return

<Instruction 8>    ; I8
<Instruction 9>    ; I9
....
```

- **Stop or Halting a Task: MSTOP and MDEBUGSTOP**

  The MSTOP and MDEBUGSTOP instructions cannot be placed three instructions before or after a conditional branch, call or return instruction (MBCNDD, MCCNDD or MRCNDD). Refer to Example 5-1. To single-step through a branch/call or return, insert the MDEBUGSTOP at least four instructions back and step from there.

- **Loading MAR0 or MAR1**

  A load of auxiliary register MAR0 or MAR1 will occur in the EXE phase of the pipeline. Any post increment of MAR0 or MAR1 using indirect addressing will occur in the D2 phase of the pipeline. Referring to Example 5-2, the following applies when loading the auxiliary registers:

  – **I1 and I2**

    The two instructions following the load instruction will use the value in MAR0 or MAR1 before the update occurs.

  – **I3**

    Loading of an auxiliary register occurs in the EXE phase while updates due to post-increment addressing occur in the D2 phase. Thus I3 cannot use the auxiliary register or there will be a conflict. In the case of a conflict, the update due to address-mode post increment will win and the auxiliary register will not be updated with #_X.

  – **I4**

    Starting with the 4th instruction MAR0 or MAR1 will have the new value.

- **Background Task Interrupted Close to a Branch**

  When the background task is running, if another task request happens (and MSTSBGRND.BGINTM is not set), then the following sequence of operations happen.

  – A check is made to determine if the following instructions are not in the pipeline (D2 – R2).
    - MBCNDD
    - MCCNDD
    - MRCNDD

  If any of the above instructions are present in the pipeline, the back ground task continues to execute until such time when the condition is satisfied. Once it is satisfied, the following actions are performed:

  – The MPC value of the D1 phase instruction is saved to the MVECTBGRNDACTIVE register.

  – The run status bit of the background task (MSTSBGRND.RUNSTS) is cleared.

  – An MSTOP instruction is forced into the D2 phase of the pipeline; it will cause the background task

to terminate.

When the background task terminates, the CLA picks the next highest pending task and begins execution. It is important to note that while the background task is pending it has the lowest priority and will, therefore, yield to any other pending task. Once all pending non-background tasks have completed execution the CLA will restore the program counter (MPC), that is, load the address from the MVECTBGRNDACTIVE register to the MPC, set the background status to RUN (MSTSBGRND.RUN = 1), and continue execution from that point.

- **MSTOP in the Background Task**

  If an MSTOP instruction occurs in the D1 phase while the background task is running, the following sequence of operations happens:

  – The RUN bit (MSTSBGRND.RUN) is cleared.
  – The address stored in MVECTBGRND is copied over to MVECTBGRNDACTIVE.
  – An interrupt, signaling the background task has completed execution, is generated. This interrupt signal is ANDed with the interrupt from Task 8 and fed to the PIE. It should be noted that if an illegal instruction occurs inside the background task, the interrupt for task 8 is fired.
  – When the background task terminates, the CLA resumes arbitration among the pending tasks.

***Example 5-2. Code Fragment for Loading MAR0 or MAR1***

```
; Assume MAR0 is 50 and #_X is 20

   MMOVI16 MAR0, #_X ; Load MAR0 with address of X (20)
   <Instruction 1>   ; I1 Will use the old value of MAR0 (50)
   <Instruction 2>   ; I2 Will use the old value of MAR0 (50)
   <Instruction 3>   ; I3 Cannot use MAR0
   <Instruction 4>   ; I4 Will use the new value of MAR0 (20)
   <Instruction 5>   ; I5 Will use the new value of MAR0 (20
   ....
```

#### 5.5.2.1 ADC Early Interrupt to CLA Response

The ADC offers the option to generate an early interrupt pulse when the ADC begins conversion. This option is selected by setting the ADCCTL1[INTPULSEPOS] bit as documented in the Analog-to-Digital Converter and Comparator section in this manual. If this option is used to start a CLA task then the CLA will be able to read the result as soon as the conversion completes and the ADC result register updates. This just-in-time sampling along with the low interrupt response of the CLA enable faster system response and higher frequency control loops.

The timing for the ADC conversion is shown in the ADC Reference Guide timing diagrams. If the ADCCLK is a divided down version of the SYSCLK, the user will have to account for the conversion time in SYSCLK cycles. For example, if using the 12-bit ADC with ADCCLK at ¼ SYSCLK, it would take 10.5 ADCCLK (42 SYSCLK) cycles to complete a conversion.

From a CLA perspective, the pipeline activity is shown in Table 5-5 an n-cycle (SYSCLK) conversion. The nth-2 instruction is in the R2 phase just in time to read the result register. While the previous n-3 instructions in the task (I1 to In-3) will enter the R2 phase of the pipeline too soon to read the conversion, they can be efficiently used for pre-processing calculations needed by the task.

**Table 5-5. ADC to CLA Early Interrupt Response**

| ADC Activity | CLA Activity | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|---|
| Sample | | | | | | | | | |
| Sample | | | | | | | | | |
| ... | | | | | | | | | |
| Sample | | | | | | | | | |
| Conversion (1) | Interrupt Received | | | | | | | | |
| Conversion (2) | Task Startup | | | | | | | | |
| Conversion (3) | Task Startup | | | | | | | | |

**Table 5-5. ADC to CLA Early Interrupt Response (continued)**

| ADC Activity | CLA Activity | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|---|
| Conversion (4) | I1 | I1 | | | | | | | |
| Conversion (5) | I2 | I2 | I1 | | | | | | |
| Conversion (6) | I3 | I3 | I2 | I1 | | | | | |
| Conversion (7) | ... | ... | ... | ... | ... | ... | ... | | |
| Conversion (n-6) | I(n-6) | I(n-6) | | | | | | | |
| Conversion (n-5) | I(n-5) | I(n-5) | I(n-6) | | | | | | |
| Conversion (n-4) | I(n-4) | I(n-4) | I(n-5) | I(n-6) | | | | | |
| Conversion (n-3) | I(n-3) | I(n-3) | I(n-4) | I(n-5) | I(n-6) | | | | |
| Conversion (n-2) | I(n-2) Read ADC RESULT | I(n-2) | I(n-3) | I(n-4) | I(n-5) | I(n-6) | | | |
| Conversion (n-1) | | | I(n-2 | I(n-3) | I(n-4) | I(n-5) | I(n-6) | | |
| Conversion (n) | | | | I(n-2) | I(n-3) | I(n-4) | I(n-5) | | |
| Conversion Complete | | | | | I(n-2) | I(n-3) | I(n-4) | | |
| RESULT Latched | | | | | | I(n-2) | I(n-3) | | |
| RESULT Available | | | | | | | I(n-2) | | |

### 5.5.3 Parallel Instructions

Parallel instructions are single opcodes that perform two operations in parallel. The following types of parallel instructions are available: math operation in parallel with a move operation, or two math operations in parallel. Both operations complete in a single cycle and there are no special pipeline alignment requirements.

***Example 5-3. Math Operation with Parallel Load***

```
;  MADDF32 || MMOV32 instruction: 32-bit floating-point add with parallel move
;  MADDF32 is a 1 cycle operation
;  MMOV32 is a 1 cycle operation
         MADDF32    MR0,  MR1, #2    ; MR0 = MR1 + 2,
      || MMOV32   MR1,  @Val         ; MR1 gets the contents of Val
                                     ; <-- MMOV32 completes here (MR1 is valid)
                                     ; <-- DDF32 completes here (MR0 is valid)
         MMPYF32 MR0, MR0, MR1       ; Any instruction, can use MR1 and/or MR0
```

***Example 5-4. Multiply with Parallel Add***

```
;  MMPYF32 || MADDF32 instruction: 32-bit floating-point multiply with parallel add
;  MMPYF32 is a 1 cycle operation
;  MADDF32 is a 1 cycle operation
     MMPYF32 MR0, MR1, MR3        ; MR0 = MR1 * MR3
 ||  MADDF32 MR1, MR2, MR0        ; MR1 = MR2 + MR0 (Uses value of MR0 before MMPYF32)
                                  ; <-- MMPYF32 and MADDF32 complete here (MR0 and MR1 are valid)
     MMPYF32 MR1, MR1, MR0        ; Any instruction, can use MR1 and/or MR0
```

## 5.6 Instruction Set

This section describes the assembly language instructions of the control law accelerator. Also described are parallel operations, conditional operations, resource constraints, and addressing modes. The instructions listed here are independent from C28x and C28x+FPU instruction sets.

### 5.6.1 Instruction Descriptions

This section gives detailed information on the instruction set. Each instruction may present the following information:

- Operands
- Opcode
- Description
- Exceptions
- Pipeline
- Examples
- See also

The example INSTRUCTION is shown to familiarize you with the way each instruction is described. The example describes the kind of information you will find in each part of the individual instruction description and where to obtain more information. CLA instructions follow the same format as the C28x; the source operand(s) are always on the right and the destination operand(s) are on the left.

The explanations for the syntax of the operands used in the instruction descriptions for the C28x CLA are given in Table 5-6.

**Table 5-6. Operand Nomenclature**

| Symbol | Description |
|---|---|
| #16FHi | 16-bit immediate (hex or float) value that represents the upper 16-bits of an IEEE 32-bit floating-point value. Lower 16-bits of the mantissa are assumed to be zero. |
| #16FHiHex | 16-bit immediate hex value that represents the upper 16-bits of an IEEE 32-bit floating-point value. Lower 16-bits of the mantissa are assumed to be zero. |
| #16FLoHex | A 16-bit immediate hex value that represents the lower 16-bits of an IEEE 32-bit floating-point value |
| #32Fhex | 32-bit immediate value that represents an IEEE 32-bit floating-point value |
| #32F | Immediate float value represented in floating-point representation |
| #0.0 | Immediate zero |
| #SHIFT | Immediate value of 1 to 32 used for arithmetic and logical shifts. |
| addr | Opcode field indicating the addressing mode |
| CNDF | Condition to test the flags in the MSTF register |
| FLAG | Selected flags from MSTF register (OR) 8 bit mask indicating which floating-point status flags to change |
| MAR0 | auxiliary register 0 |
| MAR1 | auxiliary register 1 |
| MARx | Either MAR0 or MAR1 |
| mem16 | 16-bit memory location accessed using direct, indirect, or offset addressing modes |
| mem32 | 32-bit memory location accessed using direct, indirect, or offset addressing modes |
| MRa | MR0 to MR3 registers |
| MRb | MR0 to MR3 registers |
| MRc | MR0 to MR3 registers |
| MRd | MR0 to MR3 registers |
| MRe | MR0 to MR3 registers |
| MRf | MR0 to MR3 registers |
| MSTF | CLA Floating-point Status Register |
| shift | Opcode field indicating the number of bits to shift. |
| VALUE | Flag value of 0 or 1 for selected flag (OR) 8 bit mask indicating the flag value; 0 or 1 |

Each instruction has a table that gives a list of the operands and a short description. Instructions always have their destination operand(s) first followed by the source operand(s).

**Table 5-7. INSTRUCTION dest, source1, source2 Short Description**

| | Description |
|---|---|
| dest1 | Description for the 1st operand for the instruction |
| source1 | Description for the 2nd operand for the instruction |
| source2 | Description for the 3rd operand for the instruction |
| Opcode | This section shows the opcode for the instruction |
| Description | Detailed description of the instruction execution is described. Any constraints on the operands imposed by the processor or the assembler are discussed. |
| Restrictions | Any constraints on the operands or use of the instruction imposed by the processor are discussed. |
| Pipeline | This section describes the instruction in terms of pipeline cycles as described in Section 5.5 |
| Example | Examples of instruction execution. If applicable, register and memory values are given before and after instruction execution. Some examples are code fragments while other examples are full tasks that assume the CLA is correctly configured and the main CPU has passed it data. |
| Operands | Each instruction has a table that gives a list of the operands and a short description. Instructions always have their destination operand(s) first followed by the source operand(s). |

### 5.6.2 Addressing Modes and Encoding

The CLA uses the same address to access data and registers as the main CPU. For example if the main CPU accesses an ePWM register at address 0x00 6800, then the CLA will access it using address 0x6800. Since all CLA accessible memory and registers are within the low 64k x 16 of memory, only the low 16-bits of the address are used by the CLA.

To address the CLA data memory, message RAMs and shared peripherals, the CLA supports two addressing modes:

- Direct addressing mode: Uses the address of the variable or register directly.
- Indirect addressing with 16-bit post increment. This mode uses either XAR0 or XAR1.

The CLA does not use a data page pointer or a stack pointer. The two addressing modes are encoded as shown Table 5-8.

**Table 5-8. Addressing Modes**

| Addressing Mode | 'addr' Opcode Field Encode (1) | Description |
|---|---|---|
| **@dir** | **0000** | **Direct Addressing Mode** |
| | | Example 1: MMOV32 MR1, @_VarA |
| | | Example 2: MMOV32 MR1, @_EPwm1Regs.CMPA.all |
| | | In this case the 'mmmm mmmm mmmm mmmm' opcode field will be populated with the 16-bit address of the variable. This is the low 16-bits of the address that you would use to access the variable using the main CPU. |
| | | For example @_VarA will populate the address of the variable VarA. and @_EPwm1Regs.CMPA.all will populate the address of the CMPA register. |
| ***MAR0[#imm16]++** | **0001** | **MAR0 Indirect Addressing with 16-bit Immediate Post Increment** |
| ***MAR1[#imm16]++** | **0010** | **MAR1 Indirect Addressing with 16-bit Immediate Post Increment** |
| | | addr = MAR0 (or MAR1) Access memory using the address stored in MAR0 (or MAR1). MAR0 (or MAR1) += #imm16 Then post increment MAR0 (or MAR1) by #imm16. |
| | | Example 1: MMOV32 MR0, *MAR0[2]++ |
| | | Example 2: MMOV32 MR1, *MAR1[-2]++ |
| | | For a post increment of 0 the assembler will accept both *MAR0 and *MAR0[0]++. |
| | | The 'mmmm mmmm mmmm mmmm' opcode field will be populated with the signed 16-bit pointer offset. For example if #imm16 is 2, then the opcode field will be 0x0002. Likewise if #imm16 is -2, then the opcode field will be 0xFFFE. |
| | | If addition of the 16-bit immediate causes overflow, then the value will wrap 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** |
| | | addr = MAR0 (or MAR1) + #imm16to the base Add the offset #imm16 address stored in MAR0(MAR1) to access the desired memory location |
| | | Example 1: MMOV32 MR0, *MAR0+[2] |
| | | Example 1: MMOV32 MR1, *MAR1+[-2] |
| | | The ‘mmmm mmmm mmmm mmmm’ opcode field will be populated with the signed 16-bit pointer offset. For example if #imm16 is 2, then the opcode field will be 0x0002. Likewise if #imm16 is -2, then the opcode field will be 0xFFFE. |
| | | If the addition of the 16-bit immediate causes overflow, the value will wrap 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 5-9.

**Table 5-9. Shift Field Encoding**

| Shift Value | 'shift' Opcode Field Encode |
|---|---|
| 1 | 0000 |
| 2 | 0001 |
| 3 | 0010 |
| .... | .... |
| 32 | 1111 |

Table 5-11 shows the condition field encoding for conditional instructions such as MNEGF, MSWAPF, MBCNDD, MCCNDD, and MRCNDD.

For instructions that use MRx (where x could be 'a' through 'f') as operands, the trailing alphabet appears in the opcode as a two-bit field. For example,

```
MMPYF32 MRa, MRb, MRc ||
MADDF32 MRd, MRe, MRf
```

whose opcode is,

```
LSW: 0000 ffee ddcc bbaa
MSW: 0111 1010 0000 0000
```

The two-bit field specifies one of four working registers according to Table 5-10.

**Table 5-10. Operand Encoding**

| Two-Bit Field | Working Register |
|---|---|
| b'00 | MR0 |
| b'01 | MR1 |
| b'10 | MR2 |
| b'11 | MR3 |

**Table 5-11. Condition Field Encoding**

| Encode (1) | CNDF | Description | MSTF Flags Tested |
|---|---|---|---|
| 0000 | NEQ | Not equal to zero | ZF == 0 |
| 0001 | EQ | Equal to zero | ZF == 1 |
| 0010 | GT | Greater than zero | ZF == 0 AND NF == 0 |
| 0011 | GEQ | Greater than or equal to zero | NF == 0 |
| 0100 | LT | Less than zero | NF == 1 |
| 0101 | LEQ | Less than or equal to zero | ZF == 1 OR NF == 1 |
| 1010 | TF | Test flag set | TF == 1 |
| 1011 | NTF | Test flag not set | TF == 0 |
| 1100 | LU | Latched underflow | LUF == 1 |
| 1101 | LV | Latched overflow | LVF == 1 |
| 1110 | UNC | Unconditional | None |
| 1111 | UNCF (2) | Unconditional with flag modification | None |

(1) Values not shown are reserved.

(2) This is the default operation if no CNDF field is specified. This condition will allow the ZF and NF flags to be modified when a conditional operation is executed. All other conditions will not modify these flags.

### 5.6.3 Instructions

The instructions are listed alphabetically, preceded by a summary.

**Table 5-12. General Instructions**

**Table 5-12. General Instructions (continued)**

## 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 MMABSF32 instruction.

```
if (MRb < 0) {MRa = -MRb};
        else {MRa =  MRb};
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified as follows:

```
NF = 0;
ZF = 0;
if ( MRa(30:23) == 0) ZF = 1;
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ MR0, #-2.0 ; MR0 = -2.0 (0xC0000000)
MMABSF32 MR0, MR0  ; MR0 = 2.0 (0x40000000), ZF = NF = 0

MMOVIZ MR0, #5.0  ; MR0 = 5.0 (0x40A00000)
MMABSF32 MR0, MR0  ; MR0 = 5.0 (0x40A00000), ZF = NF = 0

MMOVIZ MR0, #0.0  ; MR0 = 0.0
MMABSF32 MR0, MR0  ; MR0 = 0.0 ZF = 1, NF = 0
```

**See also**

MNEGF32 MRa, MRb {, CNDF}

## MADD32 MRa, MRb, MRc *32-Bit Integer Add*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point destination register (MR0 to MR3) |
| MRc | CLA floating-point destination register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 000cc bbaa
MSW: 0111 1110 1100 0000
```

**Description**

32-bit integer addition of MRb and MRc.

```
MRa(31:0) = MRb(31:0) + MRc(31:0);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified based on the integer results of the operation.

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1; };
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given A = (int32)1
;       B = (int32)2
;       C = (int32)-7
;
; Calculate Y2 = A + B + C
;
_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

## 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 will accept either a hex or float as the immediate value. That is, the value -1.5 can be represented as #-1.5 or #0xBFC0.

```
MRa = MRb + #16FHi:0;
```

This instruction can also be written as MADDF32 MRa, MRb, #16FHi.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MADDF32 generates an underflow condition.
- LVF = 1 if MADDF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Add to MR1 the value 2.0 in 32-bit floating-point format
; Store the result in MR0
  MADDF32 MR0, #2.0, MR1    ; MR0 = 2.0 + MR1

; Add to MR3 the value -2.5 in 32-bit floating-point format
; Store the result in MR2
  MADDF32 MR2, #-2.5, MR3   ; MR2 = -2.5 + MR3

; Add to MR3 the value 0x3FC00000 (1.5)
; Store the result in MR3
  MADDF32 MR3, #0x3FC0, MR3 ; MR3 = 1.5 + MR3
```

**See also**

MADDF32 MRa, MRb, #16FHi
MADDF32 MRa, MRb, MRc
MADDF32 MRd, MRe, MRf || MMOV32 MRa, mem32
MADDF32 MRd, MRe, MRf || MMOV32 mem32, MRa
MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRe, MRf

## MADDF32 MRa, MRb, #16FHi *32-Bit Floating-Point Addition*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |
| #16FHi | A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. |

**Opcode**

```
LSW: 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 will accept either a hex or float as the immediate value. That is, the value -1.5 can be represented as #-1.5 or #0xBFC0.

```
MRa = MRb + #16FHi:0;
```

This instruction can also be written as MADDF32 MRa, #16FHi, MRb.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MADDF32 generates an underflow condition.
- LVF = 1 if MADDF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.

**Example 1**

```
; X is an array of 32-bit floating-point values
; Find the maximum value in an array X
; and store it in Result
;
_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

## 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

## MADDF32 MRd, MRe, MRf||MMOV32 mem32, MRa *32-Bit Floating-Point Addition with Parallel Move*

**Operands**

| | |
|---|---|
| MRd | CLA floating-point destination register for the MADDF32 (MR0 to MR3) |
| MRe | CLA floating-point source register for the MADDF32 (MR0 to MR3) |
| MRf | CLA floating-point source register for the MADDF32 (MR0 to MR3) |
| mem32 | 32-bit memory location accessed using one of the available addressing modes. This will be the destination of the MMOV32. |
| MRa | CLA floating-point source register for the MMOV32 (MR0 to MR3) |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0101 ffee ddaa addr
```

**Description**

Perform an MADDF32 and a MMOV32 in parallel. Add MRf to the contents of MRe and store the result in MRd. In parallel move the contents of MRa to the 32-bit location mem32.

```
MRd = MRe + MRf;
[mem32] = MRa;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MADDF32 generates an underflow condition.
- LVF = 1 if MADDF32 generates an overflow condition.

**Pipeline**

Both MADDF32 and MMOV32 complete in a single cycle.

**Example**

```
; Given A, B and C are 32-bit floating-point numbers
; Calculate Y2 = (A * B)
;           Y3 = (A * B) + C
;
_Cla1Task2:
   MMOV32   MR0, @_A        ; Load MR0 with A
   MMOV32   MR1, @_B        ; Load MR1 with B
   MMPYF32  MR1, MR1, MR0   ; Multiply A*B
|| MMOV32   MR0, @_C        ;     and in parallel load MR0 with C
   MADDF32  MR1, MR1, MR0   ; Add (A*B) to C
|| MMOV32   @_Y2, MR1       ;     and in parallel store A*B
   MMOV32   @_Y3, MR1       ; Store the A*B + C
   MSTOP                    ; end of task
```

**See also**

MADDF32 MRa, #16FHi, MRb
MADDF32 MRa, MRb, #16FHi
MADDF32 MRa, MRb, MRc
MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRe, MRf
MADDF32 MRd, MRe, MRf || MMOV32 MRa, mem32

## MADDF32 MRd, MRe, MRf ||MMOV32 MRa, mem32 *32-Bit Floating-Point Addition with Parallel Move*

**Operands**

| | |
|---|---|
| MRd | CLA floating-point destination register for the MADDF32 (MR0 to MR3). MRd cannot be the same register as MRa. |
| MRe | CLA floating-point source register for the MADDF32 (MR0 to MR3) |
| MRf | CLA floating-point source register for the MADDF32 (MR0 to MR3) |
| MRa | CLA floating-point destination register for the MMOV32 (MR0 to MR3). MRa cannot be the same register as MRd. |
| mem32 | 32-bit memory location accessed using one of the available addressing modes. This is the source for the MMOV32. |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0001 ffee ddaa addr
```

**Description**

Perform an MADDF32 and a MMOV32 operation in parallel. Add MRf to the contents of MRe and store the result in MRd. In parallel move the contents of the 32-bit location mem32 to MRa.

```
MRd = MRe + MRf;
MRa = [mem32];
```

**Restrictions**

The destination register for the MADDF32 and the MMOV32 must be unique. That is, MRa and MRd cannot be the same register.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MADDF32 generates an underflow condition.
- LVF = 1 if MADDF32 generates an overflow condition.

The MMOV32 Instruction will set the NF and ZF flags as follows:

```
NF = MRa(31);
ZF = 0;
if(MRa(30:23) == 0) { ZF = 1; NF = 0; };
```

**Pipeline**

The MADDF32 and the MMOV32 both complete in a single cycle.

**Example 1**

```
; Given A, B and C are 32-bit floating-point numbers
; Calculate Y1 = A + 4B
;           Y2 = A + C
;
_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

## MAND32 MRa, MRb, MRc *Bitwise AND*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |
| MRc | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 00cc bbaa
MSW: 0111 1100 0110 0000
```

**Description**

Bitwise AND of MRb with MRc.

```
MRa(31:0) = MRb(31:0) AND MRc(31:0);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified based on the integer results of the operation.

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1; }
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ   MR0,    #0x5555  ; MR0 = 0x5555AAAA
MMOVXI   MR0,    #0xAAAA

MMOVIZ   MR1,    #0x5432  ; MR1 = 0x5432FEDC
MMOVXI   MR1,    #0xFEDC

; 0101 AND 0101 = 0101 (5)
; 0101 AND 0100 = 0100 (4)
; 0101 AND 0011 = 0001 (1)
; 0101 AND 0010 = 0000 (0)
; 1010 AND 1111 = 1010 (A)
; 1010 AND 1110 = 1010 (A)
; 1010 AND 1101 = 1000 (8)
; 1010 AND 1100 = 1000 (8)

MAND32 MR2, MR1, MR0      ; MR3 = 0x5410AA88
```

**See also**

MADD32 MRa, MRb, MRc
MASR32 MRa, #SHIFT
MLSL32 MRa, #SHIFT
MLSR32 MRa, #SHIFT
MOR32 MRa, MRb, MRc
MXOR32 MRa, MRb, MRc
MSUB32 MRa, MRb, MRc

## MASR32 MRa, #SHIFT *Arithmetic Shift Right*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point source/destination register (MR0 to MR3) |
| #SHIFT | Number of bits to shift (1 to 32) |

**Opcode**

```
LSW: 0000 0000 0shi ftaa
MSW: 0111 1011 0100 0000
```

**Description**

Arithmetic shift right of MRa by the number of bits indicated. The number of bits can be 1 to 32.

```
MARa(31:0) = Arithmetic Shift(MARa(31:0) by #SHIFT bits);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified based on the integer results of the operation.

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1; }
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given m2 = (int32)32
;       x2 = (int32)64
;       b2 = (int32)-128
;
; Calculate
;       m2 = m2/2
;       x2 = x2/4
;       b2 = b2/8
;
_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
```

**See also**

MADD32 MRa, MRb, MRc
MAND32 MRa, MRb, MRc
MLSL32 MRa, #SHIFT
MLSR32 MRa, #SHIFT
MOR32 MRa, MRb, MRc
MXOR32 MRa, MRb, MRc
MSUB32 MRa, MRb, MRc

## MBCNDD 16BitDest {, CNDF} *Branch Conditional Delayed*

**Operands**

| | |
|---|---|
| 16BitDest | 16-bit destination if condition is true |
| CNDF | Optional condition tested |

**Opcode**

```
LSW: dest dest dest dest
MSW: 0111 1001 1000 cndf
```

**Description**

If the specified condition is true, then branch by adding the signed 16BitDest value to the MPC value. Otherwise, continue without branching. If the address overflows, it wraps around. Therefore a value of "0xFFFE" will put the MPC back to the MBCNDD instruction.

Please refer to the pipeline section for important information regarding this instruction.

```
if (CNDF == TRUE) MPC += 16BitDest;
```

CNDF is one of the following conditions:

| Encode [1] | CNDF | Description | MSTF Flags Tested |
|---|---|---|---|
| 0000 | NEQ | Not equal to zero | ZF == 0 |
| 0001 | EQ | Equal to zero | ZF == 1 |
| 0010 | GT | Greater than zero | ZF == 0 AND NF == 0 |
| 0011 | GEQ | Greater than or equal to zero | NF == 0 |
| 0100 | LT | Less than zero | NF == 1 |
| 0101 | LEQ | Less than or equal to zero | ZF == 1 OR NF == 1 |
| 1010 | TF | Test flag set | TF == 1 |
| 1011 | NTF | Test flag not set | TF == 0 |
| 1100 | LU | Latched underflow | LUF == 1 |
| 1101 | LV | Latched overflow | LVF == 1 |
| 1110 | UNC | Unconditional | None |
| 1111 | UNCF [2] | Unconditional with flag modification | None |

(1) Values not shown are reserved.

(2) This is the default operation if no CNDF field is specified. This condition will allow the ZF and NF flags to be modified when a conditional operation is executed. All other conditions will not modify these flags.

**Restrictions**

The MBCNDD instruction is not allowed three instructions before or after a MBCNDD, MCCNDD or MRCNDD instruction. Refer to the pipeline section for more information.

**Flags**

This instruction does not modify flags in the MSTF register.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

The MBCNDD instruction by itself is a single-cycle instruction. As shown in Table 5-13 for each branch 6 instruction slots are executed; three before the branch instruction (I2-I4) and three after the branch instruction (I5-I7). The total number of cycles for a branch taken or not taken depends on the usage of these slots. That is, the number of cycles depends on how many slots are filled with a MNOP as well as which slots are filled. The effective number of cycles for a branch can, therefore, range from 1 to 7 cycles. The number of cycles for a branch taken may not be the same as for a branch not taken.

Referring to Table 5-13 and Table 5-14, the instructions before and after MBCNDD have the following properties:

- **I1**
  - I1 is the last instruction that can effect the CNDF flags for the MBCNDD instruction. The CNDF flags are tested in the D2 phase of the pipeline. That is, a decision is made whether to branch or not when MBCNDD is in the D2 phase.
  - There are no restrictions on the type of instruction for I1.
- **I2, I3 and I4**
  - The three instructions proceeding MBCNDD can change MSTF flags but will have no effect on whether the MBCNDD instruction branches or not. This is because the flag modification will occur after the D2 phase of the MBCNDD instruction.
  - These instructions must not be the following: MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD or MRCNDD.
- **I5, I6 and I7**
  - The three instructions following MBCNDD are always executed irrespective of whether the branch is taken or not.
  - These instructions must not be the following: MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD or MRCNDD.

```
<Instruction 1>   ; I1 Last instruction that can affect flags for
                  ; the MBCNDD operation
<Instruction 2>   ; I2 Cannot be stop, branch, call or return
<Instruction 3>   ; I3 Cannot be stop, branch, call or return
<Instruction 4>   ; I4 Cannot be stop, branch, call or return
MBCNDD _Skip, NEQ ; Branch to Skip if not 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
....
```

**Table 5-13. Pipeline Activity For MBCNDD, Branch Not Taken**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| I1 | I1 | | | | | | | |
| I2 | I2 | I1 | | | | | | |
| I3 | I3 | I2 | I1 | | | | | |
| I4 | I4 | I3 | I2 | I1 | | | | |
| MBCNDD | MBCNDD | I4 | I3 | I2 | I1 | | | |
| I5 | I5 | MBCNDD | I4 | I3 | I2 | I1 | | |
| I6 | I6 | I5 | MBCNDD | I4 | I3 | I2 | I1 | |
| I7 | I7 | I6 | I5 | MBCNDD | I4 | I3 | I2 | |
| I8 | I8 | I7 | I6 | I5 | - | I4 | I3 | |
| I9 | I9 | I8 | I7 | I6 | I5 | - | I4 | |
| I10 | I10 | I9 | I8 | I7 | I6 | I5 | - | |
| | | I10 | I9 | I8 | I7 | I6 | I5 | |
| | | | I10 | I9 | I8 | I7 | I6 | |
| | | | | I10 | I9 | I8 | I7 | |
| | | | | | I10 | I9 | I8 | |
| | | | | | | I10 | I9 | |
| | | | | | | | I10 | |

**Table 5-14. Pipeline Activity For MBCNDD, Branch Taken**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| I1 | I1 | | | | | | | |
| I2 | I2 | I1 | | | | | | |
| I3 | I3 | I2 | I1 | | | | | |
| I4 | I4 | I3 | I2 | I1 | | | | |
| MBCNDD | MBCNDD | I4 | I3 | I2 | I1 | | | |
| I5 | I5 | MBCNDD | I4 | I3 | I2 | I1 | | |
| I6 | I6 | I5 | MBCNDD | I4 | I3 | I2 | I1 | |
| I7 | I7 | I6 | I5 | MBCNDD | I4 | I3 | I2 | |
| d1 | d1 | I7 | I6 | I5 | - | I4 | I3 | |
| d2 | d2 | d1 | I7 | I6 | I5 | - | I4 | |
| d3 | d3 | d2 | d1 | I7 | I6 | I5 | - | |
| | | d3 | d2 | d1 | I7 | I6 | I5 | |
| | | | d3 | d2 | d1 | I7 | I6 | |
| | | | | d3 | d2 | d1 | I7 | |
| | | | | | d3 | d2 | d1 | |
| | | | | | | d3 | d2 | |
| | | | | | | | d3 | |

**Example 1**

```
; if (State == 0.1)
; RampState = RampState || RAMPMASK
; else if (State == 0.01)
; CoastState = CoastState || COASTMASK
; else
; SteadyState = SteadyState || STEADYMASK
;
_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
```

**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

## MCCNDD 16BitDest {, CNDF} *Call Conditional Delayed*

**Operands**

| | |
|---|---|
| 16BitDest | 16-bit destination if condition is true |
| CNDF | Optional condition to be tested |

**Opcode**

```
LSW: dest dest dest dest
MSW: 0111 1001 1001 cndf
```

**Description**

If the specified condition is true, then store the return address in the RPC field of MSTF and make the call by adding the signed 16BitDest value to the MPC value. Otherwise, continue code execution without making the call. If the address overflows, it wraps around. Therefore a value of "0xFFFE" will put the MPC back to the MCCNDD instruction.

Please refer to the pipeline section for important information regarding this instruction.

```
if (CNDF == TRUE)
{
    RPC = return address;
    MPC += 16BitDest;
};
```

CNDF is one of the following conditions:

| Encode [3] | CNDF | Description | MSTF Flags Tested |
|---|---|---|---|
| 0000 | NEQ | Not equal to zero | ZF == 0 |
| 0001 | EQ | Equal to zero | ZF == 1 |
| 0010 | GT | Greater than zero | ZF == 0 AND NF == 0 |
| 0011 | GEQ | Greater than or equal to zero | NF == 0 |
| 0100 | LT | Less than zero | NF == 1 |
| 0101 | LEQ | Less than or equal to zero | ZF == 1 OR NF == 1 |
| 1010 | TF | Test flag set | TF == 1 |
| 1011 | NTF | Test flag not set | TF == 0 |
| 1100 | LU | Latched underflow | LUF == 1 |
| 1101 | LV | Latched overflow | LVF == 1 |
| 1110 | UNC | Unconditional | None |
| 1111 | UNCF [4] | Unconditional with flag modification | None |

(3) Values not shown are reserved.

(4) This is the default operation if no CNDF field is specified. This condition will allow the ZF and NF flags to be modified when a conditional operation is executed. All other conditions will not modify these flags.

**Restrictions**

The MCCNDD instruction is not allowed three instructions before or after a MBCNDD, MCCNDD, or MRCNDD instruction. Refer to the Pipeline section for more details.

**Flags**

This instruction does not modify flags in the MSTF register.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

The MCCNDD instruction by itself is a single-cycle instruction. As shown in Table 5-15, for each call 6 instruction slots are executed; three before the call instruction (I2-I4) and three after the call instruction (I5-I7). The total number of cycles for a call taken or not taken depends on the usage of these slots. That is, the number of cycles depends on how many slots are filled with a MNOP as well as which slots are filled. The effective number of cycles for a call can, therefore, range from 1 to 7 cycles. The number of cycles for a call taken may not be the same as for a call not taken.

Referring to the following code fragment and the pipeline diagrams in Table 5-15 and Table 5-16, the instructions before and after MCCNDD have the following properties:

- **I1**
  - I1 is the last instruction that can effect the CNDF flags for the MCCNDD instruction. The CNDF flags are tested in the D2 phase of the pipeline. That is, a decision is made whether to branch or not when MCCNDD is in the D2 phase.
  - There are no restrictions on the type of instruction for I1.
- **I2, I3 and I4**
  - The three instructions proceeding MCCNDD can change MSTF flags but will have no effect on whether the MCCNDD instruction makes the call or not. This is because the flag modification will occur after the D2 phase of the MCCNDD instruction.
  - These instructions must not be the following: MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD or MRCNDD.
- **I5, I6 and I7**
  - The three instructions following MBCNDD are always executed irrespective of whether the branch is taken or not.
  - These instructions must not be the following: MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD or MRCNDD.

```
<Instruction 1>   ; I1 Last instruction that can affect flags for
                  ;    the MCCNDD operation
<Instruction 2>   ; I2 Cannot be stop, branch, call or return
<Instruction 3>   ; I3 Cannot be stop, branch, call or return
<Instruction 4>   ; I4 Cannot be stop, branch, call or return

MCCNDD _func, NEQ ; Call to func if not 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 5-15. Pipeline Activity For MCCNDD, Call Not Taken**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| I1 | I1 | | | | | | | |
| I2 | I2 | I1 | | | | | | |
| I3 | I3 | I2 | I1 | | | | | |
| I4 | I4 | I3 | I2 | I1 | | | | |
| MCCNDD | MCCNDD | I4 | I3 | I2 | I1 | | | |
| I5 | I5 | MCCNDD | I4 | I3 | I2 | I1 | | |
| I6 | I6 | I5 | MCCNDD | I4 | I3 | I2 | I1 | |
| I7 | I7 | I6 | I5 | MCCNDD | I4 | I3 | I2 | |
| I8 | I8 | I7 | I6 | I5 | - | I4 | I3 | |
| I9 | I9 | I8 | I7 | I6 | I5 | - | I4 | |
| I10 | I10 | I9 | I8 | I7 | I6 | I5 | - | |
| etc .... | | I10 | I9 | I8 | I7 | I6 | I5 | |
| .... | | | I10 | I9 | I8 | I7 | I6 | |
| .... | | | | I10 | I9 | I8 | I7 | |
| .... | | | | | I10 | I9 | I8 | |
| | | | | | | I10 | I9 | |
| | | | | | | | I10 | |

**Table 5-16. Pipeline Activity For MCCNDD, Call Taken**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| I1 | I1 | | | | | | | |
| I2 | I2 | I1 | | | | | | |
| I3 | I3 | I2 | I1 | | | | | |
| I4 | I4 | I3 | I2 | I1 | | | | |
| MCCNDD | MCCNDD | I4 | I3 | I2 | I1 | | | |
| I5 | I5 | MCCNDD | I4 | I3 | I2 | I1 | | |
| I6 | I6 | I5 | MCCNDD | I4 | I3 | I2 | I1 | |
| I7 (1) | I7 | I6 | I5 | MCCNDD | I4 | I3 | I2 | |
| d1 | d1 | I7 | I6 | I5 | - | I4 | I3 | |
| d2 | d2 | d1 | I7 | I6 | I5 | - | I4 | |
| d3 | d3 | d2 | d1 | I7 | I6 | I5 | - | |
| etc .... | | d3 | d2 | d1 | I7 | I6 | I5 | |
| .... | | | d3 | d2 | d1 | I7 | I6 | |
| .... | | | | d3 | d2 | d1 | I7 | |
| .... | | | | | d3 | d2 | d1 | |
| | | | | | | d3 | d2 | |
| | | | | | | | d3 | |

(1) The RPC value in the MSTF register will point to the instruction following I7 (instruction I8).

**Example** ;

**See also**
MBCNDD #16BitDest, CNDF
MMOV32 mem32, MSTF
MMOV32 MSTF, mem32
MRCNDD CNDF

## MCLRC BGINTM *Clear Background Task Interrupt Mask*

**Operands**

| | |
|---|---|
| None | This instruction does not have any operands |

**Opcode**

```
LSW: 0000 0000 0000 0000
MSW: 0111 1111 0111 0000
```

**Description**

This instruction will clear the background task interrupt mask (BGINTM) bit in the MSTSBGRND register, allowing any code thereafter to be interrupted by a higher priority task. This instruction clears the BGINTM bit at the end of its D2 phase.

Note: This instruction does not require the MEALLOW bit to be asserted before or deasserted after clearing BGINTM.

**Flags**

This instruction does not modify flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MCLRC BGINTM       ; Allow the background task to be
                   ; interrupted by clearing the
                   ; MSTSBGRND.BGINTM bit
```

**See also**

MSETC BGINTM

## MCMP32 MRa, MRb *32-Bit Integer Compare for Equal, Less Than or Greater Than*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point source register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1111 0010 0000
```

**Description**

Set ZF and NF flags on the result of MRa - MRb where MRa and MRb are 32-bit integers. For a floating point compare refer to MCMPF32.

**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
```

**See also**

MADD32 MRa, MRb, MRc
MSUB32 MRa, MRb, MRc

## MCMPF32 MRa, MRb *32-Bit Floating-Point Compare for Equal, Less Than or Greater Than*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point source register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1101 0000 0000
```

**Description**

Set ZF and NF flags on the result of MRa - MRb. The MCMPF32 instruction is performed as a logical compare operation. This is possible because of the IEEE format offsetting the exponent. Basically the bigger the binary number, the bigger the floating-point value.

Special cases for inputs:

- Negative zero will be treated as positive zero.
- A denormalized value will be treated as positive zero.
- Not-a-Number (NaN) will be treated as infinity.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified as follows:

```
If(MRa == MRb)   {ZF=1; NF=0;}
If(MRa > MRb) {ZF=0; NF=0;}
If(MRa < MRb) {ZF=0; NF=1;}
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Behavior of ZF and NF flags for different comparisons

    MMOVIZ    MR1, #-2.0  ; MR1 = -2.0 (0xC0000000)
    MMOVIZ    MR0, #5.0   ; MR0 = 5.0 (0x40A00000)
    MCMPF32   MR1, MR0    ; ZF = 0, NF = 1
    MCMPF32   MR0, MR1    ; ZF = 0, NF = 0
    MCMPF32   MR0, MR0    ; ZF = 1, NF = 0
```

**See also**

MCMPF32 MRa, #16FHi
MMAXF32 MRa, #16FHi
MMAXF32 MRa, MRb
MMINF32 MRa, #16FHi
MMINF32 MRa, MRb

## MCMPF32 MRa, #16FHi *32-Bit Floating-Point Compare for Equal, Less Than or Greater Than*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point source register (MR0 to MR3) |
| #16FHi | A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. |

**Opcode**

```
LSW: 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 will accept either a hex or float as the immediate value. That is, -1.5 can be represented as #-1.5 or #0xBFC0.

The MCMPF32 instruction is performed as a logical compare operation. This is possible because of the IEEE floating-point format offsets the exponent. Basically the bigger the binary number, the bigger the floating-point value.

Special cases for inputs:

- Negative zero will be treated as positive zero.
- Denormalized value will be treated as positive zero.
- Not-a-Number (NaN) will be treated as infinity.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified as follows:

```
If(MRa == #16FHi:0)   {ZF=1, NF=0;}
If(MRa > #16FHi:0) {ZF=0, NF=0;}
If(MRa < #16FHi:0) {ZF=0, NF=1;}
```

**Pipeline**

This is a single-cycle instruction

**Example 1**

```
; Behavior of ZF and NF flags for different comparisons

  MMOVIZ      MR1, #-2.0  ; MR1 = -2.0 (0xC0000000)
  MMOVIZ      MR0, #5.0   ; MR0 = 5.0 (0x40A00000)
  MCMPF32     MR1, #-2.2  ; ZF = 0, NF = 0
  MCMPF32     MR0, #6.5   ; ZF = 0, NF = 1
  MCMPF32     MR0, #5.0   ; ZF = 1, NF = 0
```

**Example 2**

```
; X is an array of 32-bit floating-point values
; and has len elements. Find the maximum value in
; the array and store it in Result
;
; Note: 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

## MDEBUGSTOP *Debug Stop Task*

**Operands**

| | |
|---|---|
| none | This instruction does not have any operands |

**Opcode**

```
LSW: 0000 0000 0000 0000
MSW: 0111 1111 0110 0000
```

**Description**

When CLA breakpoints are enabled, the MDEBUGSTOP instruction is used to halt a task so that it can be debugged. That is, MDEBUGSTOP is the CLA breakpoint. If CLA breakpoints are not enabled, the MDEBUGSTOP instruction behaves like a MNOP. Unlike the MSTOP, the MIRUN flag is not cleared and an interrupt is not issued. A single-step or run operation will continue execution of the task.

**Restrictions**

The MDEBUGSTOP instruction cannot be placed 3 instructions before or after a MBCNDD, MCCNDD or MRCNDD instruction.

**Flags**

This instruction does not modify flags in the MSTF register.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
;
```

**See also**

MSTOP, MDEBUGSTOP1

## MDEBUGSTOP1 *Software Breakpoint*

**Operands**

| | |
|---|---|
| none | This instruction does not have any operands |

**Opcode**

```
LSW: 0000 0000 0000 0000
MSW: 0111 1111 0011 0000
```

**Description**

The instruction at which a software breakpoint is placed will be replaced by the MDEBUGSTOP1 instruction. It will halt execution once it reaches the D2 phase in the pipeline; at that point the subsequent instructions that were fetched, after the halt, will be flushed from the pipeline. The replace instruction will be re-fetched after this and execution can continue normally (either in run or step mode).

See Section 5.4.3 for a detailed explanation of its operation.

**Restrictions**

The MDEBUGSTOP1 instruction cannot be placed 3 instructions before or after a MBCNDD, MCCNDD or MRCNDD instruction.

**Flags**

This instruction does not modify flags in the MSTF register.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
;
```

**See also**

MSTOP, MDEBUGSTOP

## MEALLOW — *Enable CLA Write Access to EALLOW Protected Registers*

**Operands**

| | |
|---|---|
| none | This instruction does not have any operands |

**Opcode**

```
LSW: 0000 0000 0000 0000
MSW: 0111 1111 1001 0000
```

**Description**

This instruction sets the MEALLOW bit in the CLA status register MSTF. When this bit is set, the CLA is allowed write access to EALLOW protected registers. To again protect against CLA writes to protected registers, use the MEDIS instruction.

MEALLOW and MEDIS only control CLA write access; reads are allowed even if MEALLOW has not been executed. MEALLOW and MEDIS are also independent from the main CPU's EALLOW/EDIS. This instruction does not modify the EALLOW bit in the main CPU's status register. The MEALLOW bit in MSTF only controls access for the CLA while the EALLOW bit in the ST1 register only controls access for the main CPU.

As with EALLOW, the MEALLOW bit is overridden via the JTAG port, allowing full control of register accesses during debug from Code Composer Studio.

**Flags**

This instruction does not modify flags in the MSTF register.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; C header file including definition of
; the EPwm1Regs structure
;
; The ePWM TZSEL register is EALLOW protected
;
   .cdecls C,LIST,"CLAShared.h"
   ...
_Cla1Task1:
   ...
   MEALLOW                          ; Allow CLA write access
   MMOV16 @_EPwm1Regs.TZSEL.all, MR3 ; Write to TZSEL
   MEDIS                            ; Disallow CLA write access
   ...
   ...
   MSTOP
```

**See also**

MEDIS

## MEDIS *Disable CLA Write Access to EALLOW Protected Registers*

**Operands**

| | |
|---|---|
| none | This instruction does not have any operands |

**Opcode**

```
LSW: 0000 0000 0000 0000
MSW: 0111 1111 1011 0000
```

**Description**

This instruction clears the MEALLOW bit in the CLA status register MSTF. When this bit is clear, the CLA is not allowed write access to EALLOW-protected registers. To enable CLA writes to protected registers, use the MEALLOW instruction.

MEALLOW and MEDIS only control CLA write access; reads are allowed even if MEALLOW has not been executed. MEALLOW and MEDIS are also independent from the main CPU's EALLOW/EDIS. This instruction does not modify the EALLOW bit in the main CPU's status register. The MEALLOW bit in MSTF only controls access for the CLA while the EALLOW bit in the ST1 register only controls access for the main CPU.

As with EALLOW, the MEALLOW bit is overridden via the JTAG port, allowing full control of register accesses during debug from Code Composer Studio.

**Flags**

This instruction does not modify flags in the MSTF register.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; C header file including definition of
; the EPwm1Regs structure
;
; The ePWM TZSEL register is EALLOW protected
;
   .cdecls C,LIST,"CLAShared.h"
   ...
_Cla1Task1:
   ...
   MEALLOW                          ; Allow CLA write access
   MMOV16 @_EPwm1Regs.TZSEL.all, MR3 ; Write to TZSEL
   MEDIS                            ; Disallow CLA write access
   ...
   ...
   MSTOP
```

**See also**

MEALLOW

## MEINVF32 MRa, MRb *32-Bit Floating-Point Reciprocal Approximation*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1111 0000 0000
```

**Description**

This operation generates an estimate of 1/X in 32-bit floating-point format accurate to approximately 8 bits. This value can be used in a Newton-Raphson algorithm to get a more accurate answer. That is:

```
Ye = Estimate(1/X);
Ye = Ye*(2.0 - Ye*X);
Ye = Ye*(2.0 - Ye*X);
```

After two iterations of the Newton-Raphson algorithm, you will get an exact answer accurate to the 32-bit floating-point format. On each iteration the mantissa bit accuracy approximately doubles. The MEINVF32 operation will not generate a negative zero, DeNorm or NaN value.

```
MRa = Estimate of 1/MRb;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MEINVF32 generates an underflow condition.
- LVF = 1 if MEINVF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Calculate Num/Den using a Newton-Raphson 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

## MEISQRTF32 MRa, MRb *32-Bit Floating-Point Square-Root Reciprocal Approximation*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1110 0100 0000
```

**Description**

This operation generates an estimate of 1/sqrt(X) in 32-bit floating-point format accurate to approximately 8 bits. This value can be used in a Newton-Raphson algorithm to get a more accurate answer. That is:

```
Ye = Estimate(1/sqrt(X));
Ye = Ye*(1.5 - Ye*Ye*X/2.0);
Ye = Ye*(1.5 - Ye*Ye*X/2.0);
```

After 2 iterations of the Newton-Raphson algorithm, you will get an exact answer accurate to the 32-bit floating-point format. On each iteration the mantissa bit accuracy approximately doubles. The MEISQRTF32 operation will not generate a negative zero, DeNorm or NaN value.

```
MRa = Estimate of 1/sqrt (MRb);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MEISQRTF32 generates an underflow condition.
- LVF = 1 if MEISQRTF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Y = sqrt(X)
; Ye = Estimate(1/sqrt(X));
; Ye = Ye*(1.5 - Ye*Ye*X*0.5)
; Ye = Ye*(1.5 - Ye*Ye*X*0.5)
; Y = X*Ye
;
_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

## MF32TOI16 MRa, MRb *Convert 32-Bit Floating-Point Value to 16-Bit Integer*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1101 1110 0000
```

**Description**

Convert a 32-bit floating point value in MRb to a 16-bit integer and truncate. The result will be stored in MRa.

```
MRa(15:0) = F32TOI16(MRb);
MRa(31:16) = sign extension of MRa(15);
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ      MR0, #5.0   ; MR0        = 5.0 (0x40A00000)
MF32TOI16   MR1, MR0    ; MR1(15:0)  = MF32TOI16(MR0) = 0x0005
                        ; MR1(31:16) = Sign extension of MR1(15) = 0x0000
MMOVIZ      MR2, #-5.0  ; MR2        = -5.0 (0xC0A00000)
MF32TOI16   MR3, MR2    ; MR3(15:0)  = MF32TOI16(MR2) = -5 (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

## 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

## 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
```

**See also**

MF32TOUI32 MRa, MRb
MI32TOF32 MRa, MRb
MI32TOF32 MRa, mem32
MUI32TOF32 MRa, MRb
MUI32TOF32 MRa, mem32

## MF32TOUI16 MRa, MRb *Convert 32-Bit Floating-Point Value to 16-bit Unsigned Integer*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1110 1010 0000
```

**Description**

Convert the 32-bit floating point value in MRb to an unsigned 16-bit integer value and truncate to zero. The result will be stored in MRa. To instead round the integer to the nearest even value use the MF32TOUI16R instruction.

```
MRa(15:0) = F32TOUI16(MRb);
MRa(31:16) = 0x0000;
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ      MR0, #9.0    ; MR0 = 9.0 (0x41100000)
MF32TOUI16  MR1, MR0     ; MR1(15:0) = MF32TOUI16(MR0) = 9 (0x0009)
                         ; MR1(31:16) = 0x0000
MMOVIZ      MR2, #-9.0   ; MR2 = -9.0 (0xC1100000)
MF32TOUI16  MR3, MR2     ; MR3(15:0) = MF32TOUI16(MR2) = 0 (0x0000)
                         ; MR3(31:16) = 0x0000
```

**See also**

MF32TOI16 MRa, MRb
MF32TOUI16 MRa, MRb
MF32TOUI16R MRa, MRb
MI16TOF32 MRa, MRb
MI16TOF32 MRa, mem16
MUI16TOF32 MRa, mem16
MUI16TOF32 MRa, MRb

## MF32TOUI16R MRa, MRb *Convert 32-Bit Floating-Point Value to 16-bit Unsigned Integer and Round*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1110 1100 0000
```

**Description**

Convert the 32-bit floating-point value in MRb to an unsigned 16-bit integer and round to the closest even value. The result will be stored in MRa. To instead truncate the converted value, use the MF32TOUI16 instruction.

```
MRa(15:0)  = MF32TOUI16round(MRb);
MRa(31:16) = 0x0000;
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ       MR0, #0x412C    ; MR0 = 0x412C
MMOVXI       MR0, #0xCCCD    ; MR0 = 0xCCCD ; MR0 = 10.8 (0x412CCCCD)
MF32TOUI16R  MR1, MR0        ; MR1(15:0) = MF32TOUI16round(MR0) = 11 (0x000B)
                             ; MR1(31:16) = 0x0000
MMOVF32      MR2, #-10.8     ; MR2 = -10.8 (0x0xC12CCCCD)
MF32TOUI16R  MR3, MR2        ; MR3(15:0) = MF32TOUI16round(MR2) = 0 (0x0000)
                             ; MR3(31:16) = 0x0000
```

**See also**

MF32TOI16 MRa, MRb
MF32TOI16R MRa, MRb
MF32TOUI16 MRa, MRb
MI16TOF32 MRa, MRb
MI16TOF32 MRa, mem16
MUI16TOF32 MRa, mem16
MUI16TOF32 MRa, MRb

## MF32TOUI32 MRa, MRb *Convert 32-Bit Floating-Point Value to 32-Bit Unsigned Integer*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1101 1010 0000
```

**Description**

Convert the 32-bit floating-point value in MRb to an unsigned 32-bit integer and store the result in MRa.

```
MRa = F32TOUI32(MRb);
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ      MR0, #12.5   ; MR0 = 12.5 (0x41480000)
MF32TOUI32  MR0, MR0     ; MR0 = MF32TOUI32 (MR0) = 12 (0x0000000C)
MMOVIZ      MR1, #-6.5   ; MR1 = -6.5 (0xC0D00000)
MF32TOUI32  MR2, MR1     ; MR2 = MF32TOUI32 (MR1) = 0.0 (0x00000000)
```

**See also**

MF32TOI32 MRa, MRb
MI32TOF32 MRa, MRb
MI32TOF32 MRa, mem32
MUI32TOF32 MRa, MRb
MUI32TOF32 MRa, mem32

## MFRACF32 MRa, MRb *Fractional Portion of a 32-Bit Floating-Point Value*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1110 0000 0000
```

**Description**

Returns in MRa the fractional portion of the 32-bit floating-point value in MRb

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ    MR2, #19.625 ; MR2 = 19.625 (0x419D0000)
MFRACF32  MR3, MR2     ; MR3 = MFRACF32(MR2) = 0.625 (0x3F200000)0)
```

**See also**

## MI16TOF32 MRa, MRb *Convert 16-Bit Integer to 32-Bit Floating-Point Value*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1110 1000 0000
```

**Description**

Convert the 16-bit signed integer in MRb to a 32-bit floating point value and store the result in MRa.

```
MRa = MI16TOF32(MRb);
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ      MR0, #0x0000    ; MR0(31:16) = 0.0 (0x0000)
MMOVXI      MR0, #0x0004    ; MR0(15:0) = 4.0 (0x0004)
MI16TOF32   MR1, MR0        ; MR1 = MI16TOF32 (MR0) = 4.0 (0x40800000)

MMOVIZ      MR2, #0x0000    ; MR2(31:16) = 0.0 (0x0000)
MMOVXI      MR2, #0xFFFC    ; MR2(15:0) = -4.0 (0xFFFC)
MI16TOF32   MR3, MR2        ; MR3 = MI16TOF32 (MR2) = -4.0 (0xC0800000)
MSTOP
```

**See also**

MF32TOI16 MRa, MRb
MF32TOI16R MRa, MRb
MF32TOUI16 MRa, MRb
MF32TOUI16R MRa, MRb
MI16TOF32 MRa, mem16
MUI16TOF32 MRa, mem16
MUI16TOF32 MRa, MRb

## MI16TOF32 MRa, mem16 *Convert 16-Bit Integer to 32-Bit Floating-Point Value*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| mem16 | 16-bit source memory location to be converted |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0101 00aa addr
```

**Description**

Convert the 16-bit signed integer indicated by the mem16 pointer to a 32-bit floating-point value and store the result in MRa.

```
MRa = MI16TOF32[mem16];
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction:

**Example**

```
; Assume A = 4 (0x0004)
;        B = -4 (0xFFFC)

   MI16TOF32 MR0, @_A ; MR0 = MI16TOF32(A) = 4.0 (0x40800000)
   MI16TOF32 MR1, @_B ; MR1 = MI16TOF32(B) = -4.0 (0xC0800000
```

**See also**

MF32TOI16 MRa, MRb
MF32TOI16R MRa, MRb
MF32TOUI16 MRa, MRb
MF32TOUI16R MRa, MRb
MI16TOF32 MRa, MRb
MUI16TOF32 MRa, mem16
MUI16TOF32 MRa, MRb

## MI32TOF32 MRa, mem32 *Convert 32-Bit Integer to 32-Bit Floating-Point Value*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| mem32 | 32-bit memory source for the MMOV32 operation. |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0100 01aa addr
```

**Description**

Convert the 32-bit signed integer indicated by mem32 to a 32-bit floating point value and store the result in MRa.

```
MRa = MI32TOF32[mem32];
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given X, M and B are IQ24 numbers:
; X = IQ24(+2.5) = 0x02800000
; M = IQ24(+1.5) = 0x01800000
; B = IQ24(-0.5) = 0xFF800000
;
; Calculate Y = X * M + B
;
; Convert M, X and B from IQ24 to float
;

_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

## MI32TOF32 MRa, MRb *Convert 32-Bit Integer to 32-Bit Floating-Point Value*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1101 1000 0000
```

**Description**

Convert the signed 32-bit integer in MRb to a 32-bit floating-point value and store the result in MRa.

```
MRa = MI32TOF32(MRb);
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Example1:
;
   MMOVIZ    MR2, #0x1111 ; MR2(31:16) = 4369 (0x1111)
   MMOVXI    MR2, #0x1111 ; MR2(15:0) = 4369 (0x1111)
                          ; MR2 = +286331153 (0x11111111)
   MI32TOF32 MR3, MR2     ; MR3 = MI32TOF32 (MR2) = 286331153.0 (0x4D888888)
```

**See also**

MF32TOI32 MRa, MRb
MF32TOUI32 MRa, MRb
MI32TOF32 MRa, mem32
MUI32TOF32 MRa, MRb
MUI32TOF32 MRa, mem32

## MLSL32 MRa, #SHIFT *Logical Shift Left*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point source/destination register (MR0 to MR3) |
| #SHIFT | Number of bits to shift (1 to 32) |

**Opcode**

```
LSW: 0000 0000 0shi ftaa
MSW: 0111 1011 1100 0000
```

**Description**

Logical shift left of MRa by the number of bits indicated. The number of bits can be 1 to 32.

```
MARa(31:0) = Logical Shift Left(MARa(31:0) by #SHIFT bits);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified based on the integer results of the operation.

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1; }
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given m2 = (int32)32
;       x2 = (int32)64
;       b2 = (int32)-128
;
; Calculate:
;       m2 = m2*2
;       x2 = x2*4
;       b2 = b2*8
;
_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
```

**See also**

MADD32 MRa, MRb, MRc
MASR32 MRa, #SHIFT
MAND32 MRa, MRb, MRc
MLSR32 MRa, #SHIFT
MOR32 MRa, MRb, MRc
MXOR32 MRa, MRb, MRc
MSUB32 MRa, MRb, MRc

## MLSR32 MRa, #SHIFT *Logical Shift Right*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point source/destination register (MR0 to MR3) |
| #SHIFT | Number of bits to shift (1 to 32) |

**Opcode**

```
LSW: 0000 0000 0shi ftaa
MSW: 0111 1011 1000 0000
```

**Description**

Logical shift right of MRa by the number of bits indicated. The number of bits can be 1 to 32. Unlike the arithmetic shift (MASR32), the logical shift does not preserve the number's sign bit. Every bit in the operand is moved the specified number of bit positions, and the vacant bit-positions are filled in with zeros

```
MARa(31:0) = Logical Shift Right(MARa(31:0) by #SHIFT bits);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified based on the integer results of the operation.

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1;}
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Illustrate the difference between MASR32 and MLSR32

MMOVIZ MR0, #0xAAAA   ; MR0 = 0xAAAA5555
MMOVXI MR0, #0x5555

MMOV32 MR1, MR0       ; MR1 = 0xAAAA5555
MMOV32 MR2, MR0       ; MR2 = 0xAAAA5555

MASR32 MR1, #1        ; MR1 = 0xD5552AAA
MLSR32 MR2, #1        ; MR2 = 0x55552AAA

MASR32 MR1, #1        ; MR1 = 0xEAAA9555
MLSR32 MR2, #1        ; MR2 = 0x2AAA9555

MASR32 MR1, #6        ; MR1 = 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

## 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.

**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
```

**Example 2**

```
; sum = X0*B0 + X1*B1 + X2*B2 + Y1*A1 + Y2*B2
;
;       X2 = X1
;       X1 = X0
;       Y2 = Y1 ; Y1 = sum
;
_ClaTask2:
     MMOV32     MR0, @_B2      ; MR0 = B2
     MMOV32     MR1, @_X2      ; MR1 = X2
     MMPYF32    MR2, MR1, MR0  ; MR2 = X2*B2
   || MMOV32     MR0, @_B1      ; MR0 = B1
     MMOVD32    MR1, @_X1      ; MR1 = X1, X2 = X1
     MMPYF32    MR3, MR1, MR0  ; MR3 = X1*B1
   || MMOV32     MR0, @_B0      ; MR0 = B0
     MMOVD32    MR1, @_X0      ; MR1 = X0, X1 = X0

;  MR3 = X1*B1 + X2*B2, MR2 = X0*B0
;  MR0 = A2
     MMACF32 MR3, MR2, MR2, MR1, MR0
   || MMOV32 MR0, @_A2 M

     MOV32 MR1, @_Y2           ; MR1 = Y2

;  MR3 = X0*B0 + X1*B1 + X2*B2, MR2 = Y2*A2
;  MR0 = A1
     MMACF32 MR3, MR2, MR2, MR1, MR0
   || MMOV32 MR0, @_A1

     MMOVD32 MR1,@_Y1          ; MR1 = Y1, Y2 = Y1
     MADDF32 MR3, MR3, MR2     ; MR3 = Y2*A2 + X0*B0 + X1*B1 + X2*B2
   || MMPYF32 MR2, MR1, MR0     ; MR2 = Y1*A1
     MADDF32 MR3, MR3, MR2     ; MR3 = Y1*A1 + Y2*A2 + X0*B0 + X1*B1 + X2*B2
     MMOV32 @_Y1, MR3          ; Y1 = MR3
     MSTOP                     ; end of task
```

**See also** MMPYF32 MRa, MRb, MRc || MADDF32 MRd, MRe, MRf

## MMAXF32 MRa, MRb *32-Bit Floating-Point Maximum*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point source/destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1101 0010 0000
```

**Description**

```
if(MRa < MRb) MRa = MRb;
```

Special cases for the output from the MMAXF32 operation:

- NaN output will be converted to infinity
- A denormalized output will be converted to positive zero.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The ZF and NF flags are configured on the result of the operation, not the result stored in the destination register.

```
if(MRa == MRb)  {ZF=1; NF=0;}
if(MRa > MRb) {ZF=0; NF=0;}
if(MRa < MRb) {ZF=0; NF=1;}
```

**Pipeline**

This is a single-cycle instruction.

**Example 1**

```
MMOVIZ   MR0, #5.0  ; MR0 =  5.0  (0x40A00000)
MMOVIZ   MR1, #-2.0 ; MR1 = -2.0  (0xC0000000)
MMOVIZ   MR2, #-1.5 ; MR2 = -1.5  (0xBFC00000)
MMAXF32   MR2, MR1  ; MR2 = -1.5, ZF = NF = 0
MMAXF32   MR1, MR2  ; MR1 = -1.5, ZF = 0, NF = 1
MMAXF32   MR2, MR0  ; MR2 =  5.0, ZF = 0, NF = 1
MAXF32   MR0, MR2   ; MR2 =  5.0, ZF = 1, NF = 0
```

**Example 2**

```
; X is an array of 32-bit floating-point values
; Find the maximum value in an array X
; and store it in Result
;
_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

## MMAXF32 MRa, #16FHi *32-Bit Floating-Point Maximum*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point source/destination register (MR0 to MR3) |
| #16FHi | A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. |

**Opcode**

```
LSW: IIII IIII IIII IIII
MSW: 0111 1001 0000 00aa
```

**Description**

Compare MRa with the floating-point value represented by the immediate operand. If the immediate value is larger, then load it into MRa.

```
if(MRa < #16FHi:0) MRa = #16FHi:0;
```

#16FHi is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. This addressing mode is most useful for constants where the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). The assembler will accept either a hex or float as the immediate value. That is, -1.5 can be represented as #-1.5 or #0xBFC0.

Special cases for the output from the MMAXF32 operation:

- NaN output will be converted to infinity
- A denormalized output will be converted to positive zero.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The ZF and NF flags are configured on the result of the operation, not the result stored in the destination register.

```
if(MRa == #16FHi:0)  {ZF=1; NF=0;}
if(MRa > #16FHi:0) {ZF=0; NF=0;}
if(MRa < #16FHi:0) {ZF=0; NF=1;}
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ    MR0, #5.0  ; MR0 =  5.0  (0x40A00000)
MMOVIZ    MR1, #4.0  ; MR1 =  4.0  (0x40800000)
MMOVIZ    MR2, #-1.5 ; MR2 = -1.5  (0xBFC00000)
MMAXF32   MR0, #5.5  ; MR0 =  5.5, ZF = 0, NF = 1
MMAXF32   MR1, #2.5  ; MR1 =  4.0, ZF = 0, NF = 0
MMAXF32   MR2, #-1.0 ; MR2 = -1.0, ZF = 0, NF = 1
MMAXF32   MR2, #-1.0 ; MR2 = -1.5, ZF = 1, NF = 0
```

**See also**

MMAXF32 MRa, MRb
MMINF32 MRa, MRb
MMINF32 MRa, #16FHi

## MMINF32 MRa, MRb *32-Bit Floating-Point Minimum*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point source/destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1101 0100 0000
```

**Description**

```
if(MRa > MRb) MRa = MRb;
```

Special cases for the output from the MMINF32 operation:

- NaN output will be converted to infinity
- A denormalized output will be converted to positive zero.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The ZF and NF flags are configured on the result of the operation, not the result stored in the destination register.

```
if(MRa == MRb)  {ZF=1; NF=0;}
if(MRa > MRb) {ZF=0; NF=0;}
if(MRa < MRb) {ZF=0; NF=1;}
```

**Pipeline**

This is a single-cycle instruction.

**Example 1**

```
MMOVIZ MR0, #5.0  ; MR0 = 5.0 (0x40A00000)
MMOVIZ MR1, #4.0  ; MR1 = 4.0 (0x40800000)
MMOVIZ MR2, #-1.5 ; MR2 = -1.5 (0xBFC00000)
MMINF32 MR0, MR1  ; MR0 = 4.0, ZF = 0, NF = 0
MMINF32 MR1, MR2  ; MR1 = -1.5, ZF = 0, NF = 0
MMINF32 MR2, MR1  ; MR2 = -1.5, ZF = 1, NF = 0
MMINF32 MR1, MR0  ; MR2 = -1.5, ZF = 0, NF = 1
```

**Example 2**

```
;
; X is an array of 32-bit floating-point values
; Find the minimum value in an array X
; and store it in Result
;

_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

## MMINF32 MRa, #16FHi *32-Bit Floating-Point Minimum*

**Operands**

| | |
|---|---|
| MRa | floating-point source/destination register (MR0 to MR3) |
| #16FHi | A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. |

**Opcode**

```
LSW: IIII IIII IIII IIII
MSW: 0111 1001 0100 00aa
```

**Description**

Compare MRa with the floating-point value represented by the immediate operand. If the immediate value is smaller, then load it into MRa.

```
if(MRa > #16FHi:0) MRa = #16FHi:0;
```

#16FHi is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. This addressing mode is most useful for constants where the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). The assembler will accept either a hex or float as the immediate value. That is, -1.5 can be represented as #-1.5 or #0xBFC0.

Special cases for the output from the MMINF32 operation:

- NaN output will be converted to infinity
- A denormalized output will be converted to positive zero.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The ZF and NF flags are configured on the result of the operation, not the result stored in the destination register.

```
if(MRa == #16FHi:0)   {ZF=1; NF=0;}
if(MRa > #16FHi:0) {ZF=0; NF=0;}
if(MRa < #16FHi:0) {ZF=0; NF=1;}
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ   MR0, #5.0   ; MR0 =  5.0  (0x40A00000)
MMOVIZ   MR1, #4.0   ; MR1 =  4.0  (0x40800000)
MMOVIZ   MR2, #-1.5  ; MR2 = -1.5  (0xBFC00000)
MMINF32  MR0, #5.5   ; MR0 =  5.0, ZF = 0, NF = 1
MMINF32  MR1, #2.5   ; MR1 =  2.5, ZF = 0, NF = 0
MMINF32  MR2, #-1.0  ; MR2 = -1.5, ZF = 0, NF = 1
MMINF32  MR2, #-1.5  ; MR2 = -1.5, ZF = 1, NF = 0
```

**See also**

MMAXF32 MRa, #16FHi
MMAXF32 MRa, MRb
MMINF32 MRa, MRb

## MMOV16 MARx, MRa, #16I *Load the Auxiliary Register with MRa + 16-bit Immediate Value*

**Operands**

| | |
|---|---|
| MARx | Auxiliary register MAR0 or MAR1 |
| MRa | CLA Floating-point register (MR0 to MR3) |
| #16I | 16-bit immediate value |

**Opcode**

```
LSW: 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 will occur in the EXE phase of the pipeline. Any post increment of MAR0 or MAR1 using indirect addressing will occur in the D2 phase of the pipeline. Therefore the following applies when loading the auxiliary registers:

- **I1 and I2**

  The two instructions following MMOV16 will use MAR0/MAR1 before the update occurs. Thus these two instructions will use the old value of MAR0 or MAR1.

- **I3**

  Loading of an auxiliary register occurs in the EXE phase while updates due to post-increment addressing occur in the D2 phase. Thus I3 cannot use the auxiliary register or there will be a conflict. In the case of a conflict, the update due to address-mode post increment will win and the auxiliary register will not be updated with #_X.

- **I4**

  Starting with the 4th instruction MAR0 or MAR1 will be the new value loaded with 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 Will use the old value of MAR0 (50)
  <Instruction 2>        ; I2 Will use the old value of MAR0 (50)
  <Instruction 3>        ; I3 Cannot use MAR0
  <Instruction 4>        ; I4 Will use the new value of MAR0 (30)
  <Instruction 5>        ; I5
```

**Table 5-17. Pipeline Activity For MMOV16 MARx, MRa , #16I**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| MMOV16 MAR0, MR0, #_X | MMOV16 | | | | | | | |
| I1 | I1 | MMOV16 | | | | | | |
| I2 | I2 | I1 | MMOV16 | | | | | |
| I3 | I3 | I2 | I1 | MMOV16 | | | | |
| I4 | I4 | I3 | I2 | I1 | MMOV16 | | | |
| I5 | I5 | I4 | I3 | I2 | I1 | MMOV16 | | |
| I6 | I6 | I5 | I4 | I3 | I2 | I1 | MMOV16 | |

**Example 1**

```
; Calculate an offset into a sin/cos table
;
_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
```

**Example 2**

```
; This task logs the last NUM_DATA_POINTS
; ADCRESULT1 values in the array VoltageCLA
;
; When the last element in the array has been
; filled, the task will go back to the
; the first element.
;
; Before starting the ADC conversions, force
; Task 8 to initialize the ConversionCount to zero
;
; The ADC is set to sample (acquire) for 15 SYSCLK cycles
; or 75ns. After the capacitor has captured the analog
; value, the ADC will trigger this task early.
; It takes 10.5 ADCCLKs to complete a conversion,
; the ADCCLK being SYSCLK/4
;   T_sys = 1/200MHz = 5ns
;   T_adc = 4*T_sys  = 20ns
; The ADC will take 10.5 * 4 or 42 SYSCLK cycles to complete
; a conversion. The ADC result register may be read on the
; 36th instruction after the task begins.
;
_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**

## MMOV16 MARx, mem16 *Load MAR1 with 16-bit Value*

**Operands**

| | |
|---|---|
| MARx | CLA auxiliary register MAR0 or MAR1 |
| mem16 | 16-bit destination memory accessed using one of the available addressing modes |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm (Opcode for MMOV16 MAR0, mem16)
MSW: 0111 0110 0000 addr

LSW: mmmm mmmm mmmm mmmm (Opcode for MMOV16 MAR1, mem16)
MSW: 0111 0110 0100 addr
```

**Description**

Load MAR0 or MAR1 with the 16-bit value pointed to by mem16. Refer to the pipeline section for important information regarding this instruction.

```
MAR1 = [mem16];
```

**Flags**

No flags MSTF flags are affected.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction. The load of MAR0 or MAR1 will occur in the EXE phase of the pipeline. Any post increment of MAR0 or MAR1 using indirect addressing will occur in the D2 phase of the pipeline. Therefore the following applies when loading the auxiliary registers:

- **I1 and I2**

  The two instructions following MMOV16 will use MAR0/MAR1 before the update occurs. Thus these two instructions will use the old value of MAR0 or MAR1.

- **I3**

  Loading of an auxiliary register occurs in the EXE phase while updates due to post-increment addressing occur in the D2 phase. Thus I3 cannot use the auxiliary register or there will be a conflict. In the case of a conflict, the update due to address-mode post increment will win send the auxiliary register will not be updated with #_X.

- **I4**

  Starting with the 4th instruction MAR0 or MAR1 will be the new value loaded with MMOV16.

```
; Assume MAR0 is 50 and @_X is 20

MMOV16 MAR0, @_X    ; Load MAR0 with the contents of X (20)
<Instruction 1>     ; I1 Will use the old value of MAR0 (50)
<Instruction 2>     ; I2 Will use the old value of MAR0 (50)
<Instruction 3>     ; I3 Cannot use MAR0
<Instruction 4>     ; I4 Will use the new value of MAR0 (20)
<Instruction 5>     ; I5
....
```

**Table 5-18. Pipeline Activity For MMOV16 MAR0/MAR1, mem16**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| MMOV16 MAR0, @_X | MMOV16 | | | | | | | |
| I1 | I1 | MMOV16 | | | | | | |
| I2 | I2 | I1 | MMOV16 | | | | | |
| I3 | I3 | I2 | I1 | MMOV16 | | | | |
| I4 | I4 | I3 | I2 | I1 | MMOV16 | | | |
| I5 | I5 | I4 | I3 | I2 | I1 | MMOV16 | | |
| I6 | I6 | I5 | I4 | I3 | I2 | I1 | MMOV16 | |

**Example**

```
; This task logs the last NUM_DATA_POINTS
; ADCRESULT1 values in the array VoltageCLA
;
; When the last element in the array has been
; filled, the task will go back to the
; the first element.
;
; Before starting the ADC conversions, force
; Task 8 to initialize the ConversionCount to zero
;
; The ADC is set to sample (acquire) for 15 SYSCLK cycles
; or 75ns. After the capacitor has captured the analog
; value, the ADC will trigger this task early.
; It takes 10.5 ADCCLKs to complete a conversion,
; the ADCCLK being SYSCLK/4
;   T_sys = 1/200MHz = 5ns
;   T_adc = 4*T_sys  = 20ns
; The ADC will take 10.5 * 4 or 42 SYSCLK cycles to complete
; a conversion. The ADC result register may be read on the
; 36th instruction after the task begins.
;
_Cla1Task2:
    .asg       0, N
    .loop
    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**

## MMOV16 mem16, MARx *Move 16-Bit Auxiliary Register Contents to Memory*

**Operands**

| | |
|---|---|
| mem16 | 16-bit destination memory accessed using one of the available addressing modes |
| MARx | CLA auxiliary register MAR0 or MAR1 |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm (Opcode for MMOV16 mem16, MAR0)
MSW: 0111 0110 1000 addr

LSW: mmmm mmmm mmmm mmmm (Opcode for MMOV16 mem16, MAR1)
MSW: 0111 0110 1100 addr
```

**Description**

Store the contents of MAR0 or MAR1 in the 16-bit memory location pointed to by mem16.

```
[mem16] = MAR0;
```

**Flags**

No flags MSTF flags are affected.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

**See also**

## MMOV16 mem16, MRa *Move 16-Bit Floating-Point Register Contents to Memory*

**Operands**

| | |
|---|---|
| mem16 | 16-bit destination memory accessed using one of the available addressing modes |
| MRa | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0101 11aa addr
```

**Description**

Move 16-bit value from the lower 16-bits of the floating-point register (MRa(15:0)) to the location pointed to by mem16.

```
[mem16] = MRa(15:0);
```

**Flags**

No flags MSTF flags are affected.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; This task logs the last NUM_DATA_POINTS
; ADCRESULT1 values in the array VoltageCLA
;
; When the last element in the array has been
; filled, the task will go back to the
; the first element.
;
; Before starting the ADC conversions, force
; Task 8 to initialize the ConversionCount to zero
;
; The ADC is set to sample (acquire) for 15 SYSCLK cycles
; or 75ns. After the capacitor has captured the analog
; value, the ADC will trigger this task early.
; It takes 10.5 ADCCLKs to complete a conversion,
; the ADCCLK being SYSCLK/4
;   T_sys = 1/200MHz = 5ns
;   T_adc = 4*T_sys  = 20ns
; The ADC will take 10.5 * 4 or 42 SYSCLK cycles to complete
; a conversion. The ADC result register may be read on the
; 36th instruction after the task begins.
;
_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

## 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.

**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

## MMOV32 mem32, MSTF *Move 32-Bit MSTF Register to Memory*

**Operands**

| | |
|---|---|
| MSTF | floating-point status register |
| mem32 | 32-bit destination memory |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0111 0100 addr
```

**Description** Copy the CLA's floating-point status register, MSTF, to memory.

```
[mem32] = MSTF;
```

**Flags** This instruction does not modify flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline** This is a single-cycle instruction.

**Example**

**See also** MMOV32 mem32, MRa

## MMOV32 MRa, mem32 {, CNDF} *Conditional 32-Bit Move*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| mem32 | 32-bit memory location accessed using one of the available addressing modes |
| CNDF | optional condition. |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 00cn dfaa addr
```

**Description**

If the condition is true, then move the 32-bit value referenced by mem32 to the floating-point register indicated by MRa.

```
if (CNDF == TRUE) MRa = [mem32];
```

CNDF is one of the following conditions:

| Encode [1] | CNDF | Description | MSTF Flags Tested |
|---|---|---|---|
| 0000 | NEQ | Not equal to zero | ZF == 0 |
| 0001 | EQ | Equal to zero | ZF == 1 |
| 0010 | GT | Greater than zero | ZF == 0 AND NF == 0 |
| 0011 | GEQ | Greater than or equal to zero | NF == 0 |
| 0100 | LT | Less than zero | NF == 1 |
| 0101 | LEQ | Less than or equal to zero | ZF == 1 OR NF == 1 |
| 1010 | TF | Test flag set | TF == 1 |
| 1011 | NTF | Test flag not set | TF == 0 |
| 1100 | LU | Latched underflow | LUF == 1 |
| 1101 | LV | Latched overflow | LVF == 1 |
| 1110 | UNC | Unconditional | None |
| 1111 | UNCF [2] | Unconditional with flag modification | None |

(1) Values not shown are reserved.

(2) This is the default operation if no CNDF field is specified. This condition will allow the ZF and NF flags to be modified when a conditional operation is executed. All other conditions will not modify these flags.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

```
if(CNDF == UNCF)
{
  NF = MRa(31);
  ZF = 0;
  if(MRa(30:23) == 0) { ZF = 1; NF = 0; }
}
  else No flags modified;
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given A, B, X, M1 and M2 are 32-bit floating-point
; numbers
;
; if(A == B) calculate Y = X*M1
; if(A! = B) calculate Y = X*M2
;
_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

## MMOV32 MRa, MRb {, CNDF} *Conditional 32-Bit Move*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |
| CNDF | optional condition. |

**Opcode**

```
LSW: 0000 0000 cndf bbaa
MSW: 0111 1010 1100 0000
```

**Description**

If the condition is true, then move the 32-bit value in MRb to the floating-point register indicated by MRa.

```
if (CNDF == TRUE) MRa = MRb;
```

CNDF is one of the following conditions:

| Encode [3] | CNDF | Description | MSTF Flags Tested |
|---|---|---|---|
| 0000 | NEQ | Not equal to zero | ZF == 0 |
| 0001 | EQ | Equal to zero | ZF == 1 |
| 0010 | GT | Greater than zero | ZF == 0 AND NF == 0 |
| 0011 | GEQ | Greater than or equal to zero | NF == 0 |
| 0100 | LT | Less than zero | NF == 1 |
| 0101 | LEQ | Less than or equal to zero | ZF == 1 OR NF == 1 |
| 1010 | TF | Test flag set | TF == 1 |
| 1011 | NTF | Test flag not set | TF == 0 |
| 1100 | LU | Latched underflow | LUF == 1 |
| 1101 | LV | Latched overflow | LVF == 1 |
| 1110 | UNC | Unconditional | None |
| 1111 | UNCF [4] | Unconditional with flag modification | None |

(3) Values not shown are reserved.

(4) This is the default operation if no CNDF field is specified. This condition will allow the ZF, and NF flags to be modified when a conditional operation is executed. All other conditions will not modify these flags.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

```
if(CNDF == UNCF)
 {
   NF = MRa(31); ZF = 0;
   if(MRa(30:23) == 0) {ZF = 1; NF = 0;}
}
else No flags modified;
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given: X = 8.0
;        Y = 7.0
;        A = 2.0
;        B = 5.0
; _ClaTask1
    MMOV32   MR3, @_X      ; MR3 = X = 8.0
    MMOV32   MR0, @_Y      ; MR0 = Y = 7.0
    MMAXF32  MR3, MR0      ; ZF = 0, NF = 0, MR3 = 8.0
    MMOV32   MR1, @_A, GT  ; true, MR1 = A = 2.0
    MMOV32   MR1, @_B, LT  ; false, does not load MR1
    MMOV32   MR2, MR1, GT  ; true, MR2 = MR1 = 2.0
    MMOV32   MR2, MR0, LT  ; false, does not load MR2
    MSTOP
```

**See also** MMOV32 MRa, mem32 {,CNDF}

## MMOV32 MSTF, mem32 *Move 32-Bit Value from Memory to the MSTF Register*

**Operands**

| | |
|---|---|
| MSTF | CLA status register |
| mem32 | 32-bit source memory location |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0111 0000 addr
```

**Description**

Move from memory to the CLA's status register MSTF. This instruction is most useful when nesting function calls (via MCCNDD).

```
MSTF = [mem32];
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | Yes | Yes | Yes | Yes | Yes |

Loading the status register will overwrite all flags and the RPC field. The MEALLOW field is not affected.

**Pipeline**

This is a single-cycle instruction.

**Example**

**See also**

MMOV32 mem32, MSTF

## MMOVD32 MRa, mem32 *Move 32-Bit Value from Memory with Data Copy*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point register (MR0 to MR3) |
| mem32 | 32-bit memory location accessed using one of the available addressing modes |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0100 00aa addr
```

**Description**

Move the 32-bit value referenced by mem32 to the floating-point register indicated by MRa.

```
MRa = [mem32];
[mem32+2] = [mem32];
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

```
NF = MRa(31);
ZF = 0;
if(MRa(30:23) == 0){ ZF = 1; NF = 0; }
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; sum = X0*B0 + X1*B1 + X2*B2 + Y1*A1 + Y2*B2
;
;       X2 = X1
;       X1 = X0
;       Y2 = Y1
;       Y1 = sum
;
_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}

## MMOVF32 MRa, #32F *Load the 32-Bits of a 32-Bit Floating-Point Register*

**Operands**

This instruction is an alias for MMOVIZ and MMOVXI instructions. The second operand is translated by the assembler such that the instruction becomes:

```
MMOVIZ MRa, #16FHiHex MMOVXI MRa, #16FLoHex
```

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| #32F | immediate float value represented in floating-point representation |

**Opcode**

```
LSW: IIII IIII IIII IIII (opcode of MMOVIZ MRa, #16FHiHex)
MSW: 0111 1000 0100 00aa
LSW: IIII IIII IIII IIII (opcode of MMOVXI MRa, #16FLoHex)
MSW: 0111 1000 1000 00aa
```

**Description**

Note: This instruction accepts the immediate operand only in floating-point representation. To specify the immediate value as a hex value (IEEE 32-bit floating-point format) use the MOVI32 MRa, #32FHex instruction.

Load the 32-bits of MRa with the immediate float value represented by #32F.

#32F is a float value represented in floating-point representation. The assembler will only accept a float value represented in floating-point representation. That is, 3.0 can only be represented as #3.0. #0x40400000 will result in an error.

```
MRa = #32F;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

Depending on #32FH, this instruction takes one or two cycles. If all of the lower 16-bits of the IEEE 32-bit floating-point format of #32F are zeros, then the assembler will convert MMOVF32 into only MMOVIZ instruction. If the lower 16-bits of the IEEE 32-bit floating-point format of #32F are not zeros, then the assembler will convert 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
```

**See also**

MMOVIZ MRa, #16FHi
MMOVXI MRa, #16FLoHex
MMOVI32 MRa, #32FHex

## MMOVI16 MARx, #16I *Load the Auxiliary Register with the 16-Bit Immediate Value*

**Operands**

| | |
|---|---|
| MARx | Auxiliary register MAR0 or MAR1 |
| #16I | 16-bit immediate value |

**Opcode**

```
LSW: 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 will occur in the EXE phase of the pipeline. Any post increment of MAR0 or MAR1 using indirect addressing will occur in the D2 phase of the pipeline. Therefore the following applies when loading the auxiliary registers:

- **I1 and I2**

  The two instructions following MMOVI16 will use MAR0/MAR1 before the update occurs. Thus these two instructions will use the old value of MAR0 or MAR1.

- **I3**

  Loading of an auxiliary register occurs in the EXE phase while updates due to post-increment addressing occur in the D2 phase. Thus I3 cannot use the auxiliary register or there will be a conflict. In the case of a conflict, the update due to address-mode post increment will win snd the auxiliary register will not be updated with #_X.

- **I4**

  Starting with the 4th instruction MAR0 or MAR1 will be the new value loaded with MMOVI16.

```
;  Assume MAR0 is 50 and #_X is 20

MMOVI16 MAR0, #_X             ; Load MAR0 with address of X (20)
<Instruction 1>          ; I1 Will use the old value of MAR0 (50)
<Instruction 2>          ; I2 Will use the old value of MAR0 (50)
<Instruction 3>          ; I3 Cannot use MAR0
<Instruction 4>          ; I4 Will use the new value of MAR0 (20)
<Instruction 5>          ; I5
....
```

**Table 5-19. Pipeline Activity For MMOVI16 MAR0/MAR1, #16I**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| MMOVI16 MAR0, #_X | MMOVI16 | | | | | | | |
| I1 | I1 | MMOVI16 | | | | | | |
| I2 | I2 | I1 | MMOVI16 | | | | | |
| I3 | I3 | I2 | I1 | MMOVI16 | | | | |
| I4 | I4 | I3 | I2 | I1 | MMOVI16 | | | |
| I5 | I5 | I4 | I3 | I2 | I1 | MMOVI16 | | |
| I6 | I6 | I5 | I4 | I3 | I2 | I1 | MMOVI16 | |

## MMOVI32 MRa, #32FHex *Load the 32-Bits of a 32-Bit Floating-Point Register with the Immediate*

**Operands**

| | |
|---|---|
| MRa | floating-point register (MR0 to MR3) |
| #32FHex | A 32-bit immediate value that represents an IEEE 32-bit floating-point value. |

This instruction is an alias for MMOVIZ and MMOVXI instructions. The second operand is translated by the assembler such that the instruction becomes:

```
MMOVIZ MRa, #16FHiHex
MMOVXI MRa, #16FLoHex
```

**Opcode**

```
LSW: IIII IIII IIII IIII (opcode of MMOVIZ MRa, #16FHiHex)
MSW: 0111 1000 0100 00aa

LSW: IIII IIII IIII IIII (opcode of MMOVXI MRa, #16FLoHex)
MSW: 0111 1000 1000 00aa
```

**Description**

Note: This instruction only accepts a hex value as the immediate operand. To specify the immediate value with a floating-point representation use the MMOVF32 MRa, #32F instruction.

Load the 32-bits of MRa with the immediate 32-bit hex value represented by #32Fhex.

#32Fhex is a 32-bit immediate hex value that represents the IEEE 32-bit floating-point value of a floating-point number. The assembler will only accept a hex immediate value. That is, 3.0 can only be represented as #0x40400000. #3.0 will result in an error.

```
MRa = #32FHex;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

Depending on #32FHex, this instruction takes one or two cycles. If all of the lower 16-bits of #32FHex are zeros, then assembler will convert MOVI32 to the MMOVIZ instruction. If the lower 16-bits of #32FHex are not zeros, then assembler will convert MOVI32 to a MMOVIZ and a MMOVXI instruction.

**Example**

```
MOVI32   MR1, #0x40400000 ; MR1 = 0x40400000
                          ; Assembler converts this instruction as
                          ; MMOVIZ MR1, #0x4040

MOVI32  MR2, #0x00000000  ; MR2 = 0x00000000
                          ; Assembler converts this instruction as
                          ; MMOVIZ MR2, #0x0

MOVI32  MR3, #0x40004001  ; MR3 = 0x40004001
                          ; Assembler converts this instruction as
                          ; MMOVIZ MR3, #0x4000
                          ; MMOVXI MR3, #0x4001

MOVI32  MR0, #0x00004040  ; MR0 = 0x00004040
                          ; Assembler converts this instruction as
                          ; MMOVIZ MR0, #0x0000
                          ; MMOVXI MR0, #0x4040
```

**See also**

MMOVIZ MRa, #16FHi
MMOVXI MRa, #16FLoHex
MMOVF32 MRa, #32F

## MMOVIZ MRa, #16FHi *Load the Upper 16-Bits of a 32-Bit Floating-Point Register*

**Operands**

| | |
|---|---|
| MRa | floating-point register (MR0 to MR3) |
| #16FHi | A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. |

**Opcode**

```
LSW: 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.

#16FHiHex is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. The assembler will only accept a decimal or hex immediate value. That is, -1.5 can be represented as #-1.5 or #0xBFC0.

By itself, MMOVIZ is useful for loading a floating-point register with a constant in which the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). If a constant requires all 32-bits of a floating-point register to be iniitalized, then use MMOVIZ along with the MMOVXI instruction.

```
MRa(31:16) = #16FHi;
MRa(15:0)  = 0;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Load MR0 and MR1 with -1.5 (0xBFC00000)
     MMOVIZ    MR0, #0xBFC0     ; MR0 = 0xBFC00000 (1.5)
     MMOVIZ    MR1, #-1.5       ; MR1 = -1.5 (0xBFC00000)

; Load MR2 with pi = 3.141593 (0x40490FDB)
     MMOVIZ    MR2, #0x4049     ; MR2 = 0x40490000
     MMOVXI    MR2, #0x0FDB     ; MR2 = 0x40490FDB
```

**See also**

MMOVF32 MRa, #32F
MMOVI32 MRa, #32FHex
MMOVXI MRa, #16FLoHex

## MMOVZ16 MRa, mem16 *Load MRx With 16-bit Value*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| mem16 | 16-bit source memory location |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0101 10aa addr
```

**Description**

Move the 16-bit value referenced by mem16 to the floating-point register indicated by MRa.

```
MRa(31:16) = 0;
MRa(15:0) = [mem16];
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified based on the integer results of the operation.

```
NF = 0;
if (MRa(31:0)== 0) { ZF = 1; }
```

**Pipeline**

This is a single-cycle instruction.

## MMOVXI MRa, #16FLoHex *Move Immediate to the Low 16-Bits of a Floating-Point Register*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point register (MR0 to MR3) |
| #16FLoHex | A 16-bit immediate hex value that represents the lower 16-bits of an IEEE 32-bit floating-point value. The upper 16-bits will not be modified. |

**Opcode**

```
LSW: IIII IIII IIII IIII
MSW: 0111 1000 1000 00aa
```

**Description**

Load the low 16-bits of MRa with the immediate value #16FLoHex. #16FLoHex represents the lower 16-bits of an IEEE 32-bit floating-point value. The upper 16-bits of MRa will not be modified. MMOVXI can be combined with the MMOVIZ instruction to initialize all 32-bits of a MRa register.

```
MRa(15:0) = #16FLoHex;
MRa(31:16) = Unchanged;
```

**Flags**

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Load MR0 with pi = 3.141593 (0x40490FDB)
    MMOVIZ      MR0,#0x4049   ; MR0 = 0x40490000
    MMOVXI      MR0,#0x0FDB   ; MR0 = 0x40490FDB
```

**See also**

MMOVIZ MRa, #16FHi

## MMPYF32 MRa, MRb, MRc *32-Bit Floating-Point Multiply*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |
| MRc | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 00cc bbaa
MSW: 0111 1100 0000 0000
```

**Description**

Multiply the contents of two floating-point registers.

```
MRa = MRb * MRc;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MMPYF32 generates an underflow condition.
- LVF = 1 if MMPYF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Calculate Num/Den using a Newton-Raphson 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

## MMPYF32 MRa, #16FHi, MRb *32-Bit Floating-Point Multiply*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| #16FHi | A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. |
| MRc | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: IIII IIII IIII IIII
MSW: 0111 0111 1000 baaa
```

**Description**

Multiply MRb with the floating-point value represented by the immediate operand. Store the result of the addition in MRa.

#16FHi is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. #16FHi is most useful for representing constants where the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). The assembler will accept either a hex or float as the immediate value. That is, the value -1.5 can be represented as #-1.5 or #0xBFC0.

```
MRa = MRb * #16FHi:0;
```

This instruction can also be written as MMPYF32 MRa, MRb, #16FHi.

**Flags**

This instruction modifies the following flags in the MSTF register:.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MMPYF32 generates an underflow condition.
- LVF = 1 if MMPYF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.

**Example 1**

```
; Same as example 2 but #16FHi is represented in float
    MMOVIZ    MR3, #2.0      ; MR3 = 2.0 (0x40000000)
    MMPYF32   MR0, #3.0, MR3 ; MR0 = 3.0 * MR3 = 6.0 (0x40C00000)
    MMOV32    @_X, MR0       ; Save the result in variable X
```

**Example 2**

```
; Same as example 1 but #16FHi is represented in Hex
    MMOVIZ    MR3, #2.0         ; MR3 = 2.0 (0x40000000)
    MMPYF32   MR0, #0x4040, MR3 ; MR0 = 0x4040 * MR3 = 6.0 (0x40C00000)
    MMOV32    @_X, MR0          ; Save the result in variable X
```

**Example 3**

```
; Given X, M and B are IQ24 numbers:
; X = IQ24(+2.5) = 0x02800000
; M = IQ24(+1.5) = 0x01800000
; B = IQ24(-0.5) = 0xFF800000
;
; Calculate Y = X * M + B
;
;
_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

## MMPYF32 MRa, MRb, #16FHi *32-Bit Floating-Point Multiply*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |
| #16FHi | A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. |

**Opcode**

```
LSW: IIII IIII IIII IIII
MSW: 0111 0111 1000 baaa
```

**Description**

Multiply MRb with the floating-point value represented by the immediate operand. Store the result of the addition in MRa.

#16FHi is a 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. #16FHi is most useful for representing constants where the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). The assembler will accept either a hex or float as the immediate value. That is, the value -1.5 can be represented as #-1.5 or #0xBFC0.

```
MRa = MRb * #16FHi:0;
```

This instruction can also be writen as MMPYF32 MRa, #16FHi, MRb.

**Flags**

This instruction modifies the following flags in the MSTF register:.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MMPYF32 generates an underflow condition.
- LVF = 1 if MMPYF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.

**Example 1**

```
;Same as example 2 but #16FHi is represented in float
    MMOVIZ    MR3, #2.0       ; MR3 = 2.0 (0x40000000)
    MMPYF32   MR0, MR3, #3.0  ; MR0 = MR3 * 3.0 = 6.0 (0x40C00000)
    MMOV32    @_X, MR0        ; Save the result in variable X
```

**Example 2**

```
;Same as above example but #16FHi is represented in Hex
    MMOVIZ    MR3, #2.0          ; MR3 = 2.0 (0x40000000)
    MMPYF32   MR0, MR3, #0x4040  ; MR0 = MR3 * 0x4040 = 6.0 (0x40C00000)
    MMOV32    @_X, MR0           ; Save the result in variable X
```

**Example 3**

```
; Given X, M and B are IQ24 numbers:
; X = IQ24(+2.5) = 0x02800000
; M = IQ24(+1.5) = 0x01800000
; B = IQ24(-0.5) = 0xFF800000
;
; Calculate Y = X * M + B
;
_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

## 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.

**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

## MMPYF32 MRd, MRe, MRf ||MMOV32 MRa, mem32 *32-Bit Floating-Point Multiply with Parallel Move*

**Operands**

| | |
|---|---|
| MRd | CLA floating-point destination register for the MMPYF32 (MR0 to MR3)<br>MRd cannot be the same register as MRa |
| MRe | CLA floating-point source register for the MMPYF32 (MR0 to MR3) |
| MRf | CLA floating-point source register for the MMPYF32 (MR0 to MR3) |
| MRa | CLA floating-point destination register for the MMOV32 (MR0 to MR3)<br>MRa cannot be the same register as MRd |
| mem32 | 32-bit memory location accessed using one of the available addressing modes. This will be the source of the MMOV32. |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0000 ffee ddaa addr
```

**Description**

Multiply the contents of two floating-point registers and load another.

```
MRd = MRe * MRf;
MRa = [mem32];
```

**Restrictions**

The destination register for the MMPYF32 and the MMOV32 must be unique. That is, MRa cannot be the same register as MRd.

**Flags**

This instruction modifies the following flags in the MSTF register:.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MMPYF32 generates an underflow condition.
- LVF = 1 if MMPYF32 generates an overflow condition.

The MMOV32 Instruction will set the NF and ZF flags as follows:

```
NF = MRa(31);
ZF = 0;
if(MRa(30:23) == 0) { ZF = 1; NF = 0; }
```

**Pipeline**

Both MMPYF32 and MMOV32 complete in a single cycle.

**Example 1**

```
; Given M1, X1 and B1 are 32-bit floating point
; Calculate Y1 = M1*X1+B1
;
_Cla1Task1:
    MMOV32     MR0, @M1        ; Load MR0 with M1
    MMOV32     MR1, @X1        ; Load MR1 with X1
    MMPYF32    MR1, MR1, MR0   ; Multiply M1*X1
||  MMOV32     MR0, @B1        ; and in parallel load MR0 with B1
    MADDF32    MR1, MR1, MR0   ; Add M*X1 to B1 and store in MR1
    MMOV32     @Y1, MR1        ; Store the result
    MSTOP                      ; end of task
```

**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

## MMPYF32 MRd, MRe, MRf ||MMOV32 mem32, MRa *32-Bit Floating-Point Multiply with Parallel Move*

**Operands**

| | |
|---|---|
| MRd | CLA floating-point destination register for the MMPYF32 (MR0 to MR3) |
| MRe | CLA floating-point source register for the MMPYF32 (MR0 to MR3) |
| MRf | CLA floating-point source register for the MMPYF32 (MR0 to MR3) |
| mem32 | 32-bit memory location accessed using one of the available addressing modes. This will be the destination of the MMOV32. |
| MRa | CLA floating-point source register for the MMOV32 (MR0 to MR3) |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0100 ffee ddaa addr
```

**Description**

Multiply the contents of two floating-point registers and move from memory to register.

```
MRd = MRe * MRf;
[mem32] = MRa;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MMPYF32 generates an underflow condition.
- LVF = 1 if MMPYF32 generates an overflow condition.

**Pipeline**

MMPYF32 and MMOV32 both complete in a single cycle.

**Example**

```
; Given A, B and C are 32-bit floating-point numbers
; Calculate Y2 = (A * B)
;           Y3 = (A * B) * C
;
_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

## 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
```

**See also**

MSUBF32 MRa, MRb, MRc
MSUBF32 MRd, MRe, MRf || MMOV32 MRa, mem32
MSUBF32 MRd, MRe, MRf || MMOV32 mem32, MRa

## MNEGF32 MRa, MRb{, CNDF} *Conditional Negation*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |
| CNDF | condition tested |

**Opcode**

```
LSW: 0000 0000 cndf bbaa
MSW: 0111 1010 1000 0000
```

**Description**

```
if (CNDF == true) {MRa = - MRb; }
else {MRa = MRb; }
```

CNDF is one of the following conditions:

| Encode [5] | CNDF | Description | MSTF Flags Tested |
|---|---|---|---|
| 0000 | NEQ | Not equal to zero | ZF == 0 |
| 0001 | EQ | Equal to zero | ZF == 1 |
| 0010 | GT | Greater than zero | ZF == 0 AND NF == 0 |
| 0011 | GEQ | Greater than or equal to zero | NF == 0 |
| 0100 | LT | Less than zero | NF == 1 |
| 0101 | LEQ | Less than or equal to zero | ZF == 1 OR NF == 1 |
| 1010 | TF | Test flag set | TF == 1 |
| 1011 | NTF | Test flag not set | TF == 0 |
| 1100 | LU | Latched underflow | LUF == 1 |
| 1101 | LV | Latched overflow | LVF == 1 |
| 1110 | UNC | Unconditional | None |
| 1111 | UNCF [6] | Unconditional with flag modification | None |

(5) Values not shown are reserved.

(6) This is the default operation if no CNDF field is specified. This condition will allow the ZF, and NF flags to be modified when a conditional operation is executed. All other conditions will not modify these flags.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example 1**

```
; Show the basic operation of MNEGF32
;
     MMOVIZ    MR0, #5.0      ; MR0 = 5.0 (0x40A00000)
     MMOVIZ    MR1, #4.0      ; MR1 = 4.0 (0x40800000)
     MMOVIZ    MR2, #-1.5     ; MR2 = -1.5 (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

## MNOP *No Operation*

**Operands**

| | |
|---|---|
| none | This instruction does not have any operands |

**Opcode**

```
LSW: 0000 0000 0000 0000
MSW: 0111 1111 1010 0000
```

**Description**

Do nothing. This instruction is used to fill required pipeline delay slots when other instructions are not available to fill the slots.

**Flags**

This instruction does not modify flags in the MSTF register.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; X is an array of 32-bit floating-point values
; Find the maximum value in an array X
; and store it in Result
;
_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
```

**See also**

## MOR32 MRa, MRb, MRc *Bitwise OR*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |
| MRc | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 00cc bbaa
MSW: 0111 1100 1000 0000
```

**Description**

Bitwise OR of MRb with MRc.

```
MARa(31:0) = MARb(31:0) OR MRc(31:0);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified based on the integer results of the operation.

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1; }
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ    MR0, #0x5555 ; MR0 = 0x5555AAAA
MMOVXI    MR0, #0xAAAA

MMOVIZ    MR1, #0x5432 ; MR1 = 0x5432FEDC
MMOVXI    MR1, #0xFEDC

                       ; 0101 OR 0101 = 0101 (5)
                       ; 0101 OR 0100 = 0101 (5)
                       ; 0101 OR 0011 = 0111 (7)
                       ; 0101 OR 0010 = 0111 (7)
                       ; 1010 OR 1111 = 1111 (F)
                       ; 1010 OR 1110 = 1110 (E)
                       ; 1010 OR 1101 = 1111 (F)
                       ; 1010 OR 1100 = 1110 (E)
MOR32 MR2, MR1, MR0    ; MR3 = 0x5555FEFE
```

**See also**

MAND32 MRa, MRb, MRc
MXOR32 MRa, MRb, MRc

## MRCNDD {CNDF} *Return Conditional Delayed*

**Operands**

| | |
|---|---|
| CNDF | optional condition. |

**Opcode**

```
LSW: 0000 0000 0000 0000
MSW: 0111 1001 1010 cndf
```

**Description**

If the specified condition is true, then the RPC field of MSTF is loaded into MPC and fetching continues from that location. Otherwise program fetches will continue without the return.

Please refer to the pipeline section for important information regarding this instruction.

```
if (CNDF == TRUE) MPC = RPC;
```

CNDF is one of the following conditions:

| Encode [7] | CNDF | Description | MSTF Flags Tested |
|---|---|---|---|
| 0000 | NEQ | Not equal to zero | ZF == 0 |
| 0001 | EQ | Equal to zero | ZF == 1 |
| 0010 | GT | Greater than zero | ZF == 0 AND NF == 0 |
| 0011 | GEQ | Greater than or equal to zero | NF == 0 |
| 0100 | LT | Less than zero | NF == 1 |
| 0101 | LEQ | Less than or equal to zero | ZF == 1 OR NF == 1 |
| 1010 | TF | Test flag set | TF == 1 |
| 1011 | NTF | Test flag not set | TF == 0 |
| 1100 | LU | Latched underflow | LUF == 1 |
| 1101 | LV | Latched overflow | LVF == 1 |
| 1110 | UNC | Unconditional | None |
| 1111 | UNCF [8] | Unconditional with flag modification | None |

[7] Values not shown are reserved.

[8] This is the default operation if no CNDF field is specified. This condition will allow the ZF and NF flags to be modified when a conditional operation is executed. All other conditions will not modify these flags.

**Flags**

This instruction does not modify flags in the MSTF register.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

The MRCNDD instruction by itself is a single-cycle instruction. As shown in Table 5-20, for each return 6 instruction slots are executed; three before the return instruction (d5-d7) and three after the return instruction (d8-d10). The total number of cycles for a return taken or not taken depends on the usage of these slots. That is, the number of cycles depends on how many slots are filled with a MNOP as well as which slots are filled. The effective number of cycles for a return can, therefore, range from 1 to 7 cycles. The number of cycles for a return taken may not be the same as for a return not taken.

Referring to the following code fragment and the pipeline diagrams in Table 5-20 and Table 5-21, the instructions before and after MRCNDD have the following properties:

```
;
;
<Instruction 1>    ; I1 Last instruction that can affect flags for
                   ; the MCCNDD operation
<Instruction 2>    ; I2 Cannot be stop, branch, call or return
<Instruction 3>    ; I3 Cannot be stop, branch, call or return
<Instruction 4>    ; I4 Cannot be stop, branch, call or return

MCCNDD _func, NEQ  ; Call to func if not 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 will have no effect on whether the MRCNDD instruction makes the return or not. This is

because the flag modification will occur after the D2 phase of the MRCNDD instruction.

  - These instructions must not be the following: MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD or MRCNDD.

- **d8, d9 and d10**
  - The three instructions following MRCNDD are always executed irrespective of whether the return is taken or not.
  - These instructions must not be the following: MSTOP, MDEBUGSTOP, MBCNDD, MCCNDD or MRCNDD.

**Table 5-20. Pipeline Activity For MRCNDD, Return Not Taken**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| d4 | d4 | d3 | d2 | d1 | I7 | I6 | I5 | |
| d5 | d5 | d4 | d3 | d2 | d1 | I7 | I6 | |
| d6 | d6 | d5 | d4 | d3 | d2 | d1 | i7 | |
| d7 | d7 | d6 | d5 | d4 | d3 | d2 | d1 | |
| MRCNDD | MRCNDD | d7 | d6 | d5 | d4 | d3 | d2 | |
| d8 | d8 | MRCNDD | d7 | d6 | d5 | d4 | d3 | |
| d9 | d9 | d8 | MRCNDD | d7 | d6 | d5 | d4 | |
| d10 | d10 | d9 | d8 | MRCNDD | d7 | d6 | d5 | |
| d11 | d11 | d10 | d9 | d8 | - | d7 | d6 | |
| d12 | d12 | d11 | d10 | d9 | d8 | - | d7 | |
| etc.... | .... | d12 | d11 | d10 | d9 | d8 | - | |
| .... | .... | .... | d12 | d11 | d10 | d9 | d8 | |
| .... | .... | .... | .... | d12 | d11 | d10 | d9 | |
| | | | | | d12 | d11 | d10 | |
| | | | | | | d12 | d11 | |
| | | | | | | | d12 | |

**Table 5-21. Pipeline Activity For MRCNDD, Return Taken**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| d4 | d4 | d3 | d2 | d1 | I7 | I6 | I5 | |
| d5 | d5 | d4 | d3 | d2 | d1 | I7 | I6 | |
| d6 | d6 | d5 | d4 | d3 | d2 | d1 | i7 | |
| d7 | d7 | d6 | d5 | d4 | d3 | d2 | d1 | |
| MRCNDD | MRCNDD | d7 | d6 | d5 | d4 | d3 | d2 | |
| d8 | d8 | MRCNDD | d7 | d6 | d5 | d4 | d3 | |
| d9 | d9 | d8 | MRCNDD | d7 | d6 | d5 | d4 | |
| d10 | d10 | d9 | d8 | MRCNDD | d7 | d6 | d5 | |
| I8 | I8 | d10 | d9 | d8 | - | d7 | d6 | |
| I9 | I9 | I8 | d10 | d9 | d8 | - | d7 | |
| I10 | I10 | I9 | I8 | d10 | d9 | d8 | - | |
| etc.... | .... | I10 | I9 | I8 | d10 | d9 | d8 | |
| .... | .... | | I10 | I9 | I8 | d10 | d9 | |
| .... | .... | | | I10 | I9 | I8 | d10 | |
| | | | | | I10 | I9 | I8 | |
| | | | | | | I10 | I9 | |
| | | | | | | | I10 | |

**Example**

```
;
```

**See also**

MBCNDD #16BitDest, CNDF
MCCNDD 16BitDest, CNDF
MMOV32 mem32, MSTF
MMOV32 MSTF, mem32

## MSETC BGINTM *Set Background Task Interrupt Mask*

**Operands**

| | |
|---|---|
| none | This instruction does not have any operands |

**Opcode**

```
LSW: 0000 0000 0000 0000
MSW: 0111 1111 0101 0000
```

**Description**

This instruction will set the background task interrupt mask (BGINTM) bit in the MSTSBGRND register, making any code thereafter un-interruptible. No other higher priority task will be able to interrupt the background task until the BGINTM is cleared. This instruction sets the BGINTM bit at the end of its D2 phase.

Note: This instruction does not require the MEALLOW bit to be asserted before, or de-asserted after, setting BGINTM.

**Flags**

This instruction does not modify the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MSETC BGINTM        ; Set the MSTSBGRND.BGINTM bit
                    ; to prevent any other tasks from
                    ; interrupting the background task
```

**See also**

MCLRC BGINTM

## MSETFLG FLAG, VALUE *Set or Clear Selected Floating-Point Status Flags*

**Operands**

| | |
|---|---|
| FLAG | 8 bit mask indicating which floating-point status flags to change. |
| VALUE | 8 bit mask indicating the flag value; 0 or 1. |

**Opcode**

```
LSW: FFFF FFFF VVVV VVVV
MSW: 0111 1001 1100 0000
```

**Description**

The MSETFLG instruction is used to set or clear selected floating-point status flags in the MSTF register. The FLAG field is an 11-bit value that indicates which flags will be changed. That is, if a FLAG bit is set to 1 it indicates that flag will be changed; all other flags will not be modified. The bit mapping of the FLAG field is shown below:

| RNDF32 | reserved | reserved | TF | reserved | reserved | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|---|---|---|---|
| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

The VALUE field indicates the value the flag should be set to; 0 or 1.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | Yes | Yes | Yes | Yes | Yes |

Any flag can be modified by this instruction. The MEALLOW and RPC fields cannot be modified with this instruction.

**Pipeline**

This is a single-cycle instruction.

**Example**

To make it easier and legible, the assembler accepts a FLAG=VALUE syntax for the MSTFLG operation as shown below:

```
MSETFLG RNDF32=0, TF=0, NF=1; FLAG = 11000100; VALUE = 00XXX1XX;
```

**See also**

MMOV32 mem32, MSTF
MMOV32 MSTF, mem32

## MSTOP *Stop Task*

**Operands**

| | |
|---|---|
| none | This instruction does not have any operands |

**Opcode**

```
LSW: 0000 0000 0000 0000
MSW: 0111 1111 1000 0000
```

**Description**

The MSTOP instruction must be placed to indicate the end of each task. In addition, placing MSTOP in unused memory locations within the CLA program RAM can be useful for debugging and preventing run away CLA code. When MSTOP enters the D2 phase of the pipeline, the MIRUN flag for the task is cleared and the associated interrupt is flagged in the PIE vector table.

There are three special cases that can occur when single-stepping a task such that the MPC reaches the MSTOP instruction.

1. If you are single-stepping or halted in "task A" and "task B" comes in before the MPC reaches the MSTOP, then "task B" will start if you continue to step through the MSTOP instruction. Basically if "task B" is pending before the MPC reaches MSTOP in "task A" then there is no issue in "task B" starting and no special action is required.
2. In this case you have single-stepped or halted in "task A" and the MPC has reached the MSTOP with no tasks pending. If "task B" comes in at this point, it will be flagged in the MIFR register but it may or may not start if you continue to single-step through the MSTOP instruction of "task A". It depends on exactly when the new task comes in. To reliably start "task B" perform a soft reset and reconfigure the MIER bits. Once this is done, you can start single-stepping "task B".
3. Case 2 can be handled slightly differently if there is control over when "task B" comes in (for example using the IACK instruction to start the task). In this case you have single-stepped or halted in "task A" and the MPC has reached the MSTOP with no tasks pending. Before forcing "task B", run free to force the CLA out of the debug state. Once this is done you can force "task B" and continue debugging.

**Restrictions**

The MSTOP instruction cannot be placed 3 instructions before or after a MBCNDD, MCCNDD or MRCNDD instruction.

**Flags**

This instruction does not modify flags in the MSTF register.

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction. Table 5-22 shows the pipeline behavior of the MSTOP instruction. The MSTOP instruction cannot be placed with 3 instructions of a MBCNDD, MCCNDD or MRCNDD instruction.

**Table 5-22. Pipeline Activity For MSTOP**

| Instruction | F1 | F2 | D1 | D2 | R1 | R2 | E | W |
|---|---|---|---|---|---|---|---|---|
| I1 | I1 | | | | | | | |
| I2 | I2 | I1 | | | | | | |
| I3 | I3 | I2 | I1 | | | | | |
| MSTOP | MSTOP | I3 | I2 | I1 | | | | |
| I4 | I4 | MSTOP | I3 | I2 | I1 | | | |
| I5 | I5 | I4 | MSTOP | I3 | I2 | I1 | | |
| I6 | I6 | I5 | I4 | MSTOP | I3 | I2 | I1 | |
| New Task Arbitrated and Piroitized | - | - | - | - | - | I3 | I2 | |
| New Task Arbitrated and Piroitized | - | - | - | - | - | - | I3 | |
| I1 | I1 | - | - | - | - | - | - | |
| I2 | I2 | I1 | - | - | - | - | - | |
| I3 | I3 | I2 | I1 | - | - | - | - | |
| I4 | I4 | I3 | I2 | I1 | - | - | - | |
| I5 | I5 | I4 | I3 | I2 | I1 | - | - | |
| I6 | I6 | I5 | I4 | I3 | I2 | I1 | - | |
| I7 | I7 | I6 | I5 | I4 | I3 | I2 | I1 | |
| etc .... | | | | | | | | |

**Example**

```
; Given A = (int32)1
;       B = (int32)2
;       C = (int32)-7
;
; Calculate Y2 = A - B - C
_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 , MDEBUGSTOP1

## MSUB32 MRa, MRb, MRc *32-Bit Integer Subtraction*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point destination register (MR0 to MR3) |
| MRc | CLA floating-point destination register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 00cc bbaa
MSW: 0111 1100 1110 0000
```

**Description**

32-bit integer addition of MRb and MRc.

```
MARa(31:0) = MARb(31:0) - MRc(31:0);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified as follows:

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1; }
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given A = (int32)1
;       B = (int32)2
;       C = (int32)-7
;
;
Calculate Y2 = A - B - C
;
_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

## 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

## MSUBF32 MRa, #16FHi, MRb *32-Bit Floating-Point Subtraction*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to R1) |
| #16FHi | A 16-bit immediate value that represents the upper 16-bits of an IEEE 32-bit floating-point value. The low 16-bits of the mantissa are assumed to be all 0. |
| MRb | CLA floating-point source register (MR0 to R1) |

**Opcode**

```
LSW: 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 low 16-bits of the mantissa are assumed to be all 0. #16FHi is most useful for representing constants where the lowest 16-bits of the mantissa are 0. Some examples are 2.0 (0x40000000), 4.0 (0x40800000), 0.5 (0x3F000000), and -1.5 (0xBFC00000). The assembler will accept either a hex or float as the immediate value. That is, the value -1.5 can be represented as #-1.5 or #0xBFC0.

```
MRa = #16FHi:0 - MRb;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MSUBF32 generates an underflow condition.
- LVF = 1 if MSUBF32 generates an overflow condition.

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Y = sqrt(X)
; Ye = Estimate(1/sqrt(X));
; Ye = Ye*(1.5 - Ye*Ye*X*0.5)
; Ye = Ye*(1.5 - Ye*Ye*X*0.5)
; Y = X*Ye
;
_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

## 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 will set the NF and ZF flags as follows:

**Pipeline**

Both MSUBF32 and MMOV32 complete in a single cycle.

**Example**

```
NF = MRa(31);
ZF = 0;
if(MRa(30:23) == 0) { ZF = 1; NF = 0; }
```

**See also**

MSUBF32 MRa, MRb, MRc
MSUBF32 MRa, #16FHi, MRb
MMPYF32 MRa, MRb, MRc || MSUBF32 MRd, MRe, MRf

## MSUBF32 MRd, MRe, MRf ||MMOV32 mem32, MRa *32-Bit Floating-Point Subtraction with Parallel Move*

**Operands**

| | |
|---|---|
| MRd | CLA floating-point destination register (MR0 to MR3) for the MSUBF32 operation |
| MRe | CLA floating-point source register (MR0 to MR3) for the MSUBF32 operation |
| MRf | CLA floating-point source register (MR0 to MR3) for the MSUBF32 operation |
| mem32 | 32-bit destination memory location for the MMOV32 operation |
| MRa | CLA floating-point source register (MR0 to MR3) for the MMOV32 operation |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0110 ffee ddaa addr
```

**Description**

Subtract the contents of two floating-point registers and move from a floating-point register to memory.

```
MRd = MRe - MRf;
[mem32] = MRa;
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | Yes | Yes |

The MSTF register flags are modified as follows:

- LUF = 1 if MSUBF32 generates an underflow condition.
- LVF = 1 if MSUBF32 generates an overflow condition.

**Pipeline**

Both MSUBF32 and MMOV32 complete in a single cycle.

**Example**

**See also**

MSUBF32 MRa, MRb, MRc
MSUBF32 MRa, #16FHi, MRb
MSUBF32 MRd, MRe, MRf || MMOV32 MRa, mem32
MMPYF32 MRa, MRb, MRc || MSUBF32 MRd, MRe, MRf

## MSWAPF MRa, MRb {, CNDF} *Conditional Swap*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point register (MR0 to MR3) |
| MRb | CLA floating-point register (MR0 to MR3) |
| CNDF | Optional condition tested based on the MSTF flags |

**Opcode**

```
LSW: 0000 0000 CNDF bbaa
MSW: 0111 1011 0000 0000
```

**Description**

Conditional swap of MRa and MRb.

```
if (CNDF == true) swap MRa and MRb;
```

CNDF is one of the following conditions:

| Encode [1] | CNDF | Description | MSTF Flags Tested |
|---|---|---|---|
| 0000 | NEQ | Not equal to zero | ZF == 0 |
| 0001 | EQ | Equal to zero | ZF == 1 |
| 0010 | GT | Greater than zero | ZF == 0 AND NF == 0 |
| 0011 | GEQ | Greater than or equal to zero | NF == 0 |
| 0100 | LT | Less than zero | NF == 1 |
| 0101 | LEQ | Less than or equal to zero | ZF == 1 OR NF == 1 |
| 1010 | TF | Test flag set | TF == 1 |
| 1011 | NTF | Test flag not set | TF == 0 |
| 1100 | LU | Latched underflow | LUF == 1 |
| 1101 | LV | Latched overflow | LVF == 1 |
| 1110 | UNC | Unconditional | None |
| 1111 | UNCF [2] | Unconditional with flag modification | None |

(1) Values not shown are reserved.

(2) This is the default operation if no CNDF field is specified. This condition will allow the ZF and NF flags to be modified when a conditional operation is executed. All other conditions will not modify these flags.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

No flags affected

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; X is an array of 32-bit floating-point values
; and has len elements. Find the maximum value in
; the array and store it in Result
;
; Note: 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
```

**See also**

## MTESTTF CNDF *Test MSTF Register Flag Condition*

**Operands**

| | |
|---|---|
| CNDF | condition to test based on MSTF flags |

**Opcode**

```
LSW: 0000 0000 0000 cndf
MSW: 0111 1111 0100 0000
```

**Description**

Test the CLA floating-point condition and if true, set the MSTF[TF] flag. If the condition is false, clear the MSTF[TF] flag. This is useful for temporarily storing a condition for later use.

```
if (CNDF == true) TF = 1;
else TF = 0;
```

CNDF is one of the following conditions:

| Encode [3] | CNDF | Description | MSTF Flags Tested |
|---|---|---|---|
| 0000 | NEQ | Not equal to zero | ZF == 0 |
| 0001 | EQ | Equal to zero | ZF == 1 |
| 0010 | GT | Greater than zero | ZF == 0 AND NF == 0 |
| 0011 | GEQ | Greater than or equal to zero | NF == 0 |
| 0100 | LT | Less than zero | NF == 1 |
| 0101 | LEQ | Less than or equal to zero | ZF == 1 OR NF == 1 |
| 1010 | TF | Test flag set | TF == 1 |
| 1011 | NTF | Test flag not set | TF == 0 |
| 1100 | LU | Latched underflow | LUF == 1 |
| 1101 | LV | Latched overflow | LVF == 1 |
| 1110 | UNC | Unconditional | None |
| 1111 | UNCF [4] | Unconditional with flag modification | None |

(3) Values not shown are reserved.

(4) This is the default operation if no CNDF field is specified. This condition will allow the ZF and NF flags to be modified when a conditional operation is executed. All other conditions will not modify these flags.

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | Yes | No | No | No | No |

```
TF = 0;
if (CNDF == true) TF = 1;
```

Note: If (CNDF == UNC or UNCF), the TF flag will be set to 1.

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; if (State == 0.1)
;    RampState = RampState || RAMPMASK
; else if (State == 0.01)
;    CoastState = CoastState || COASTMASK
; else
;    SteadyState = SteadyState || STEADYMASK
;
_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**

## 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 will round to nearest (even) value.

```
MRa = UI16TOF32[mem16];
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

**See also**

MF32TOI16 MRa, MRb
MF32TOI16R MRa, MRb
MF32TOUI16 MRa, MRb
MF32TOUI16R MRa, MRb
MI16TOF32 MRa, MRb
MI16TOF32 MRa, mem16
MUI16TOF32 MRa, MRb

## MUI16TOF32 MRa, MRb *Convert Unsigned 16-Bit Integer to 32-Bit Floating-Point Value*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1110 1110 0000
```

**Description**

Convert an unsigned 16-bit integer to a 32-bit floating-point value. When converting float32 to I16/UI16 data format, the MF32TOI16/UI16 operation truncates to zero while the MF32TOI16R/UI16R operation will round to nearest (even) value.

```
MRa = UI16TOF32[MRb];
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVXI MR1, #0x800F ; MR1(15:0) = 32783 (0x800F)
MUI16TOF32 MR0, MR1 ; MR0 = UI16TOF32 (MR1(15:0))
                    ; = 32783.0 (0x47000F00)
```

**See also**

MF32TOI16 MRa, MRb
MF32TOI16R MRa, MRb
MF32TOUI16 MRa, MRb
MF32TOUI16R MRa, MRb
MI16TOF32 MRa, MRb
MI16TOF32 MRa, mem16
MUI16TOF32 MRa, mem16

## MUI32TOF32 MRa, mem32 *Convert Unsigned 32-Bit Integer to 32-Bit Floating-Point Value*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| mem32 | 32-bit memory location accessed using one of the available addressing modes |

**Opcode**

```
LSW: mmmm mmmm mmmm mmmm
MSW: 0111 0100 10aa addr
```

**Description**

```
MRa = UI32TOF32[mem32];
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
; Given x2, m2 and b2 are Uint32 numbers:
;
; x2 = Uint32(2) = 0x00000002
; m2 = Uint32(1) = 0x00000001
; b2 = Uint32(3) = 0x00000003
;
; Calculate y2 = x2 * m2 + b2
;
_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

## MUI32TOF32 MRa, MRb *Convert Unsigned 32-Bit Integer to 32-Bit Floating-Point Value*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 0000 bbaa
MSW: 0111 1101 1100 0000
```

**Description**

```
MRa = UI32TOF32 [MRb];
```

**Flags**

This instruction does not affect any flags:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | No | No | No | No |

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ     MR3, #0x8000 ; MR3(31:16) = 0x8000
MMOVXI     MR3, #0x1111 ; MR3(15:0) = 0x1111
                        ; MR3 = 2147488017
MUI32TOF32 MR3, MR3     ; MR3 = MUI32TOF32 (MR3) = 2147488017.0 (0x4F000011)
```

**See also**

MF32TOI32 MRa, MRb
MF32TOUI32 MRa, MRb
MI32TOF32 MRa, mem32
MI32TOF32 MRa, MRb
MUI32TOF32 MRa, mem32

## MXOR32 MRa, MRb, MRc *Bitwise Exclusive Or*

**Operands**

| | |
|---|---|
| MRa | CLA floating-point destination register (MR0 to MR3) |
| MRb | CLA floating-point source register (MR0 to MR3) |
| MRc | CLA floating-point source register (MR0 to MR3) |

**Opcode**

```
LSW: 0000 0000 00cc bbaa
MSW: 0111 1100 1010 0000
```

**Description**

Bitwise XOR of MRb with MRc.

```
MARa(31:0) = MARb(31:0) XOR MRc(31:0);
```

**Flags**

This instruction modifies the following flags in the MSTF register:

| Flag | TF | ZF | NF | LUF | LVF |
|---|---|---|---|---|---|
| Modified | No | Yes | Yes | No | No |

The MSTF register flags are modified based on the integer results of the operation.

```
NF = MRa(31);
ZF = 0;
if(MRa(31:0) == 0) { ZF = 1; }
```

**Pipeline**

This is a single-cycle instruction.

**Example**

```
MMOVIZ MR0, #0x5555   ; MR0 = 0x5555AAAA
  MMOVXI MR0, #0xAAAA

  MMOVIZ MR1, #0x5432   ; MR1 = 0x5432FEDC
  MMOVXI MR1, #0xFEDC

  ; 0101 XOR 0101 = 0000 (0)
  ; 0101 XOR 0100 = 0001 (1)
  ; 0101 XOR 0011 = 0110 (6)
  ; 0101 XOR 0010 = 0111 (7)
  ; 1010 XOR 1111 = 0101 (5)
  ; 1010 XOR 1110 = 0100 (4)
  ; 1010 XOR 1101 = 0111 (7)
  ; 1010 XOR 1100 = 0110 (6)

  MXOR32 MR2, MR1, MR0  ; MR3 = 0x01675476
```

**See also**

MAND32 MRa, MRb, MRc
MOR32 MRa, MRb, MRc

## 5.7 Registers

### 5.7.1 Control Law Accelerator Base Addresses

**Table 5-23. CLA Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| Cla1OnlyRegs | CLA_ONLY_REGS | 0x0000_0C00 | 0x0000_0CFF |
| Cla1SoftIntRegs | CLA_SOFTINT_REGS | 0x0000_0CE0 | 0x0000_0CFF |
| Cla1Regs | CLA_REGS | 0x0000_1400 | 0x0000_147F |

### 5.7.1.1 CLA_ONLY_REGS Registers

Table 5-24 lists the memory-mapped registers for the CLA_ONLY_REGS. All register offset addresses not listed in Table 5-24 should be considered as reserved locations and the register contents should not be modified.

**Table 5-24. CLA_ONLY_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 80h | _MVECTBGRNDACTIVE | Active register for MVECTBGRND. | EALLOW | Go |
| C0h | _MPSACTL | CLA PSA Control Register | EALLOW | Go |
| C2h | _MPSA1 | CLA PSA1 Register | EALLOW | Go |
| C4h | _MPSA2 | CLA PSA2 Register | EALLOW | Go |
| E0h | SOFTINTEN | CLA Software Interrupt Enable Register | | Go |
| E2h | SOFTINTFRC | CLA Software Interrupt Force Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 5-25 shows the codes that are used for access types in this section.

**Table 5-25. CLA_ONLY_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| W=1 | 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. |

#### 5.7.1.1.1 _MVECTBGRNDACTIVE Register (Offset = 80h) [reset = 0h]

_MVECTBGRNDACTIVE is shown in Figure 5-2 and described in Table 5-26.

Return to Summary Table.

Gives the current interrupted MPC value of the background task, if the background task was running and interrupted, or reflects the MVECTBGRND value, if MCTLBGRND.BGSTART bit is 0.

**Figure 5-2. _MVECTBGRNDACTIVE Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| i16 | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 5-26. _MVECTBGRNDACTIVE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | i16 | R | 0h | Gives the current interrupted MPC value of the background task, if the background task was running and interrupted, or reflects the MVECTBGRND value, if MCTLBGRND.BGSTART bit is 0.<br>Reset type: SYSRSn |

#### 5.7.1.1.2 _MPSACTL Register (Offset = C0h) [reset = 0h]

_MPSACTL is shown in Figure 5-3 and described in Table 5-27.

Return to Summary Table.

PSA Control Register

**Figure 5-3. _MPSACTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MPSA2CFG | | MPSA2CLEAR | MPSA1CLEAR | MDWDBCYC | MDWDBSTART | MPABCYC | MPABSTART |
| R/W-0h | | R=0/W=1-0h | R=0/W=1-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 5-27. _MPSACTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-6 | MPSA2CFG | R/W | 0h | CLA PSA2 Polynomial Configuration Bits: These bits configure the type of polynomial used for PSA2. The polynomials chosen are commonly used in the industry:<br>Mode Polynomial Type<br>0,0 PSA<br>0,1 CRC32<br>1,0 CRC16<br>1,1 CRC16-CCITT<br>Note: [1] Polynomial configuration should be performed when PSA2 is stopped.<br>Reset type: SYSRSn |
| 5 | MPSA2CLEAR | R=0/W=1 | 0h | CLA PSA2 Clear Bit:<br>Writing of "1" will clear contents of PSA2 register.<br>Writes of "0" are ignored.<br>Always reads back a "0"<br>Note: Clearing operation should be performed when PSA2 is stopped.<br>Reset type: SYSRSn |
| 4 | MPSA1CLEAR | R=0/W=1 | 0h | CLA PSA1 Clear Bit:<br>Writing of "1" will clear contents of PSA1 register.<br>Writes of "0" are ignored.<br>Always reads back a "0"<br>Note: Clearing operation should be performed when PSA1 is stopped.<br>Reset type: SYSRSn |
| 3 | MDWDBCYC | R/W | 0h | CLA Data Write Data Bus PSA2 Cycle or Event Based Bit:<br>0 PSA2 calculated on every cycle<br>1 PSA2 calculated on every bus event<br>Reset type: SYSRSn |
| 2 | MDWDBSTART | R/W | 0h | CLA Data Write Data Bus PSA2 Start/Stop Bit:<br>0 PSA2 stopped<br>1 PSA2 start<br>Reset type: SYSRSn |

**Table 5-27. _MPSACTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | MPABCYC | R/W | 0h | CLA Program Address Bus PSA1 Cycle/Event Based Bit:<br>0 PSA1 calculated on every cycle<br>1 PSA1 calculated on every bus event<br>Reset type: SYSRSn |
| 0 | MPABSTART | R/W | 0h | CLA Program Address Bus PSA1 Start/Stop Bit:<br>0 PSA1 stopped<br>1 PSA1 start<br>Reset type: SYSRSn |

#### 5.7.1.1.3 _MPSA1 Register (Offset = C2h) [reset = 0h]

_MPSA1 is shown in Figure 5-4 and described in Table 5-28.

Return to Summary Table.

PSA1 Register

**Figure 5-4. _MPSA1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| i32 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 5-28. _MPSA1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | i32 | R/W | 0h | PSA1 Value: Reading this register gives the current PSA1 value. The value can be read at any time.<br><br>Writes to this register are allowed to initialize the PSA1 to a known value. Writes to this register should only be made when PSA1 is stopped.<br><br>Register value is cleared to zero by reset or by writing to the MPSA1CLEAR bit in the MPSACTL register.<br>Reset type: SYSRSn |

#### 5.7.1.1.4 _MPSA2 Register (Offset = C4h) [reset = 0h]

_MPSA2 is shown in Figure 5-5 and described in Table 5-29.

Return to Summary Table.

PSA2 Register

**Figure 5-5. _MPSA2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| i32 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 5-29. _MPSA2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | i32 | R/W | 0h | PSA2 Value: Reading this register gives the current PSA2 value. The value can be read at any time.<br><br>Writes to this register are allowed to initialize the PSA2 to a known value. Writes to this register should only be made when PSA2 is stopped.<br><br>Register value is cleared to zero by reset or by writing to the MPSA2CLEAR bit in the MPSACTL register.<br>Reset type: SYSRSn |

#### 5.7.1.1.5 SOFTINTEN Register (Offset = E0h) [reset = 0h]

SOFTINTEN is shown in Figure 5-6 and described in Table 5-30.

Return to Summary Table.

Enables the ability to generate CLA task interrupt from within the task, by writing to SOFTINTFRC register. SOFTINTFRC register can only be written from CLA.

**Figure 5-6. SOFTINTEN Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TASK8 | TASK7 | TASK6 | TASK5 | TASK4 | TASK3 | TASK2 | TASK1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 5-30. SOFTINTEN Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R/W | 0h | Reserved |
| 7 | TASK8 | R/W | 0h | 0: End-Of-Task Interrupt is fired for the respective task<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 |

**Table 5-30. SOFTINTEN Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | TASK1 | R/W | 0h | 0: End-Of-Task Interrupt is fired for the respective task<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 |

#### 5.7.1.1.6 SOFTINTFRC Register (Offset = E2h) [reset = 0h]

SOFTINTFRC is shown in Figure 5-7 and described in Table 5-31.

Return to Summary Table.

Writing a value of 1 in a bit will generate the corresponding task interrupt.

**Figure 5-7. SOFTINTFRC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TASK8 | TASK7 | TASK6 | TASK5 | TASK4 | TASK3 | TASK2 | TASK1 |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 5-31. SOFTINTFRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R/W | 0h | Reserved |
| 7 | TASK8 | R=0/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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 |

### 5.7.1.2 CLA_SOFTINT_REGS Registers

Table 5-32 lists the memory-mapped registers for the CLA_SOFTINT_REGS. All register offset addresses not listed in Table 5-32 should be considered as reserved locations and the register contents should not be modified.

**Table 5-32. CLA_SOFTINT_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | SOFTINTEN | CLA Software Interrupt Enable Register | | Go |
| 2h | SOFTINTFRC | CLA Software Interrupt Force Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 5-33 shows the codes that are used for access types in this section.

**Table 5-33. CLA_SOFTINT_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| W=1 | 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. |

#### 5.7.1.2.1 SOFTINTEN Register (Offset = 0h) [reset = 0h]

SOFTINTEN is shown in Figure 5-8 and described in Table 5-34.

Return to Summary Table.

Enables the ability to generate CLA task interrupt from within the task, by writing to SOFTINTFRC register. SOFTINTFRC register can only be written from CLA.

**Figure 5-8. SOFTINTEN Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TASK8 | TASK7 | TASK6 | TASK5 | TASK4 | TASK3 | TASK2 | TASK1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 5-34. SOFTINTEN Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R/W | 0h | Reserved |
| 7 | TASK8 | R/W | 0h | 0: End-Of-Task Interrupt is fired for the respective task<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 |

**Table 5-34. SOFTINTEN Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | TASK1 | R/W | 0h | 0: End-Of-Task Interrupt is fired for the respective task<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 |

#### 5.7.1.2.2 SOFTINTFRC Register (Offset = 2h) [reset = 0h]

SOFTINTFRC is shown in Figure 5-9 and described in Table 5-35.

Return to Summary Table.

Writing a value of 1 in a bit will generate the corresponding task interrupt.This register is only accessible by the CLA (not the CPU).

**Figure 5-9. SOFTINTFRC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TASK8 | TASK7 | TASK6 | TASK5 | TASK4 | TASK3 | TASK2 | TASK1 |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 5-35. SOFTINTFRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R/W | 0h | Reserved |
| 7 | TASK8 | R=0/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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 |

# Dual-Clock Comparator (DCC) Module

This chapter describes the dual-clock comparator (DCC) module.

**Topic** **Page**

## 6.1 Introduction

The primary purpose of a DCC module is to ensure the correct frequency range for several different device clock sources, thereby enhancing the system safety metrics.

### 6.1.1 Main Features

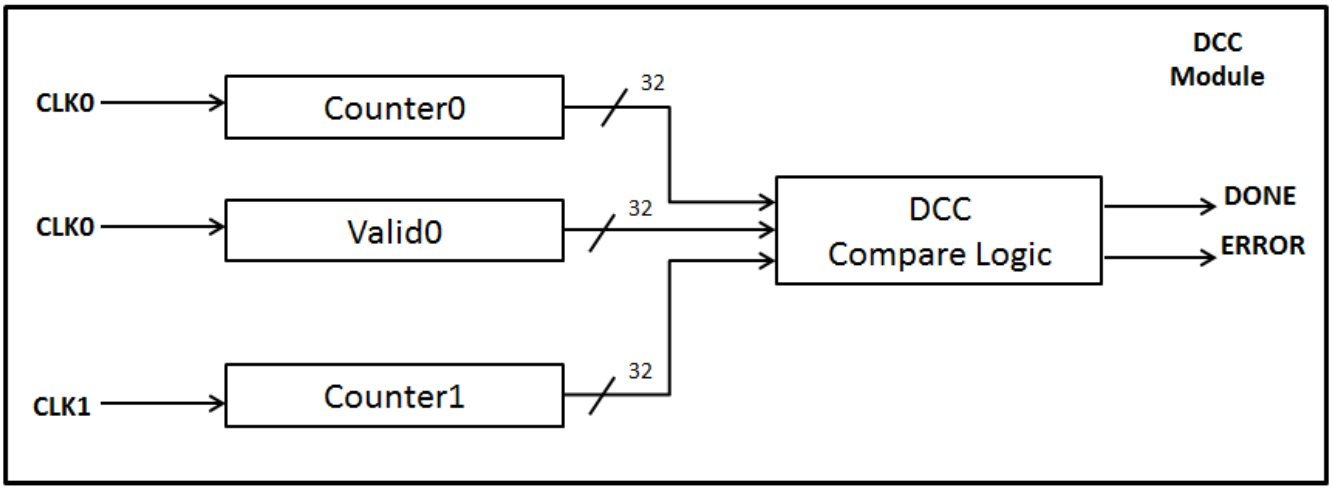
The main features of each of the DCC modules are:

- Allows application to ensure that a fixed ratio is maintained between frequencies of two clock signals
- Supports the definition of a programmable tolerance window in terms of number of reference clock cycles
- Also supports a single-sequence mode for spot measurements
- Allows selection of clock source for each of the counters resulting in several specific use cases

### 6.1.2 Block Diagram

Figure 6-1 illustrates the main concept of the DCC module.

**Figure 6-1. DCC Operation**



## 6.2 Module Operation

As shown in Figure 6-1, the DCC contains two counters – counter0 and counter1, which are driven by two signals – clock0 and clock1. The application programs the seed values for both these counters. The application also configures the tolerance window time by configuring the valid counter for clock0.

Counter0 and counter1 both start counting simultaneously once the DCC is enabled. When counter0 counts down to zero, this automatically triggers the count down of the tolerance window counter (valid0).

The DCC module can be used in two different operating modes:

### 6.2.1 Single-Shot Measurement Mode

The DCC module can be programmed to count down one time by enabling the single-shot mode. In this mode, the DCC stops operating when the down counter0 and the valid counter0 reach 0.

At the end of one sequence of counting down in this single-shot mode, the DCC gets disabled automatically, which prevents further counting. This mode is typically used for spot checking of the frequency of a signal.

**Example Usage of Single-Shot Measurement Mode: Validating PLLRAWCLK frequency**

A practical example of the usage is to validate the PLL output clock frequency using the XTAL as the reference clock. This measurement sequence would proceed as follows:

- The application sets up the seed values for counter0 and valid0 for the duration of the measurement. Suppose the XTAL frequency is 10MHz and the intended duration of the measurement is 500µs. The application needs to configure a seed value of 5000.
- These 5000 counts need to be divided between the counter0 and the valid0 counters. The minimum value for the valid0 seed is 4, so the application can configure counter0 seed value as 4996 and the valid0 seed value as 4.
- Suppose PLL is locked at 100MHz and the output of the PLL needs to be validated. In this case the application needs to set Counter1 seed value 10 times of Counter0 since the ratio between PLLRAWCLK to XTAL is 10. The application needs to configure a seed value of 50000 (5000x10).
- Once the DCC is enabled, the counters counter0 and counter1 both start counting down from their seed values.
- When counter0 reaches zero, it automatically triggers the valid0 counter.
- When valid0 reaches zero, if counter1 is not zero as well, an ERROR status flag is set and a "DCC error" is sent to the PIE. Counter1 is also frozen so that it stops counting down any further. The application can enable an interrupt to be generated from the PIE whenever this DCC error is indicated. Refer to PIE Channel Mapping table to know the channel mapping of DCC Interrupt.
- The application then needs to clear the ERROR status flag and restart the DCC module so that it is ready for the next spot measurement.

If there is no error generated at the end of the sequence, then the DONE status flag is set and a DONE interrupt is generated. The application must clear the DONE flag before restarting the DCC.

**Error Conditions:**

An error condition is generated by any one of the following:

1. Counter1 counts down to 0 before Counter0 reaches 0. This means that clock1 is faster than expected, or clock0 is slower than expected. It includes the case when clock0 is stuck at 1 or 0.
2. Counter1 does not reach 0 even when Counter0 and Valid0 have both reached 0. This means that clock1 is slower than expected. It includes the case when clock1 is stuck at 1 or 0.

   Any error freezes the counters from counting. An application may then read out the counter values to help determine what caused the error.

### 6.2.2 Error Conditions

While operating in continuous mode, the counters get reloaded with the seed values and continue counting down under the following conditions:

- The module is reset or restarted by the application, OR

- Counter0, Valid 0 and Counter1 all reach 0 without any error

**Figure 6-2. Counter Relationship**

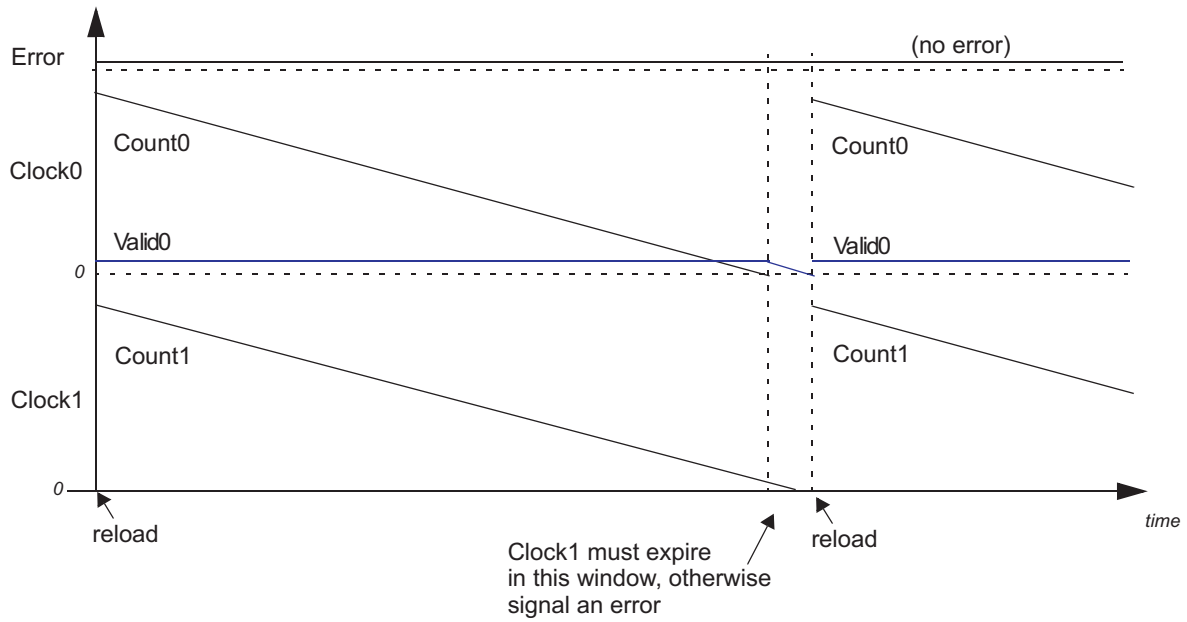



**Figure 6-3. Clock1 Slower Than Clock0 - Results in an Error and Stops Counting**

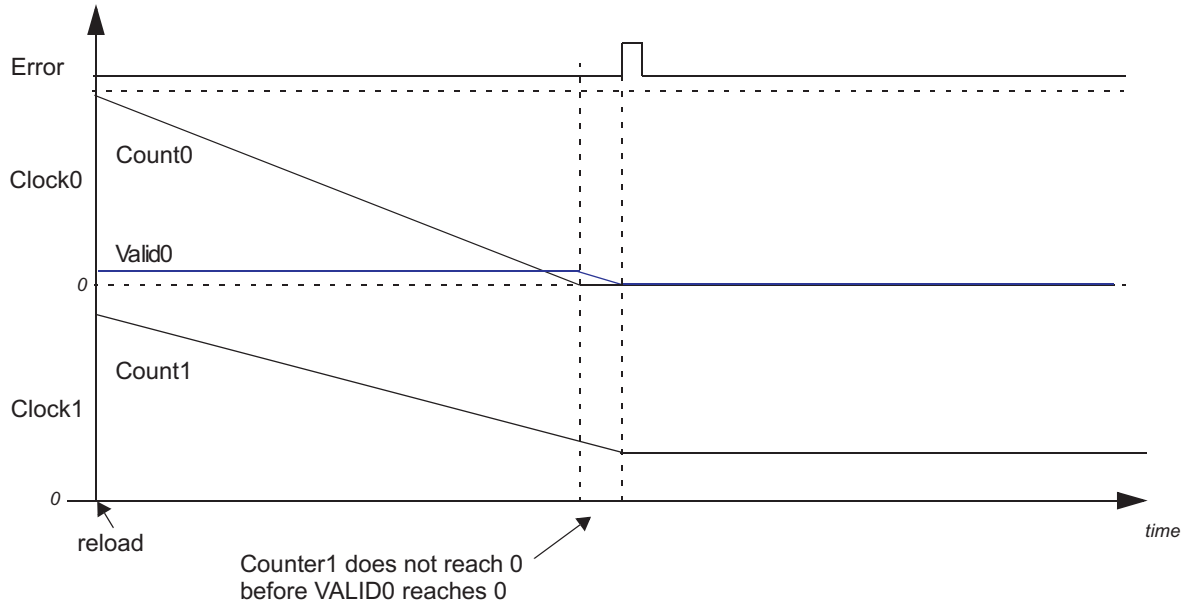


**Figure 6-4. Clock1 Faster Than Clock0 - Results in an Error and Stops Counting**

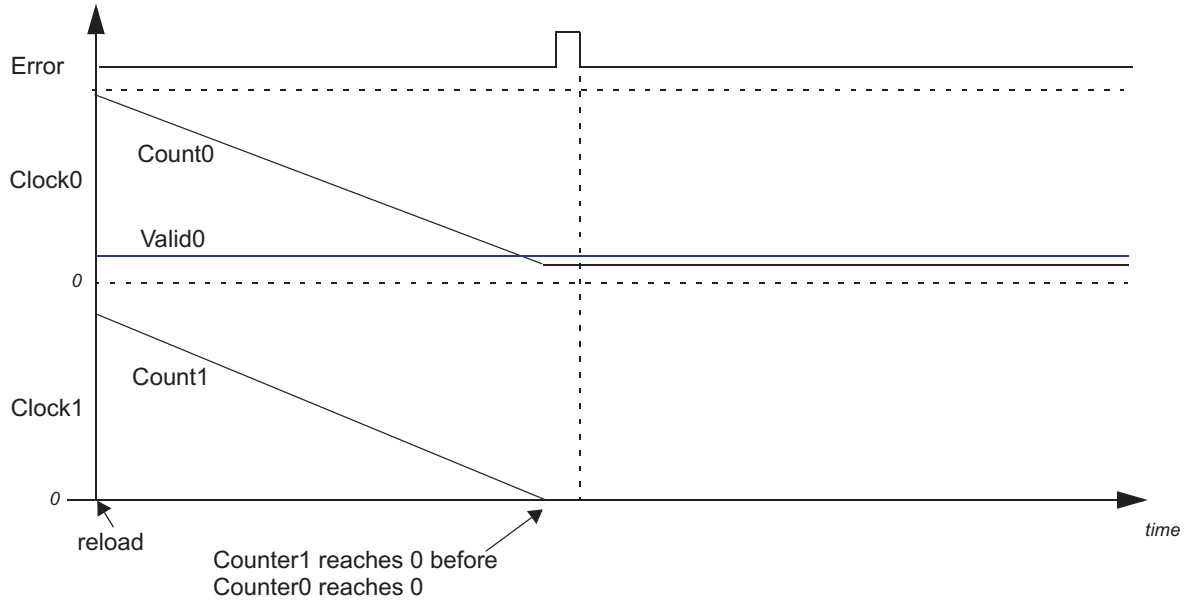



**Figure 6-5. Clock1 Not Present - Results in an Error and Stops Counting**

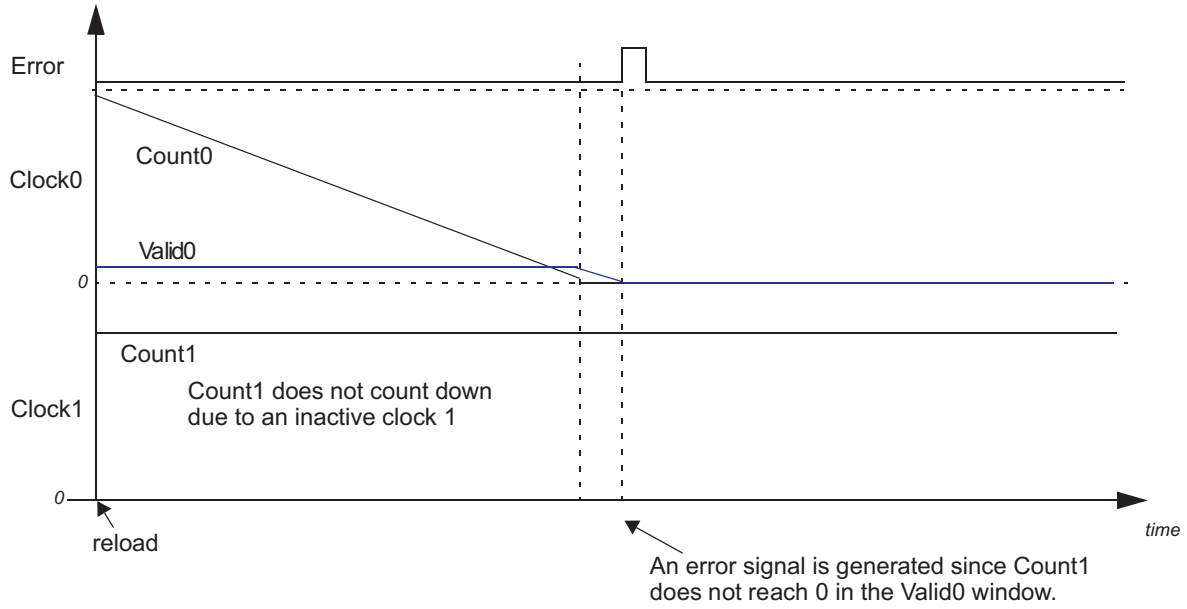
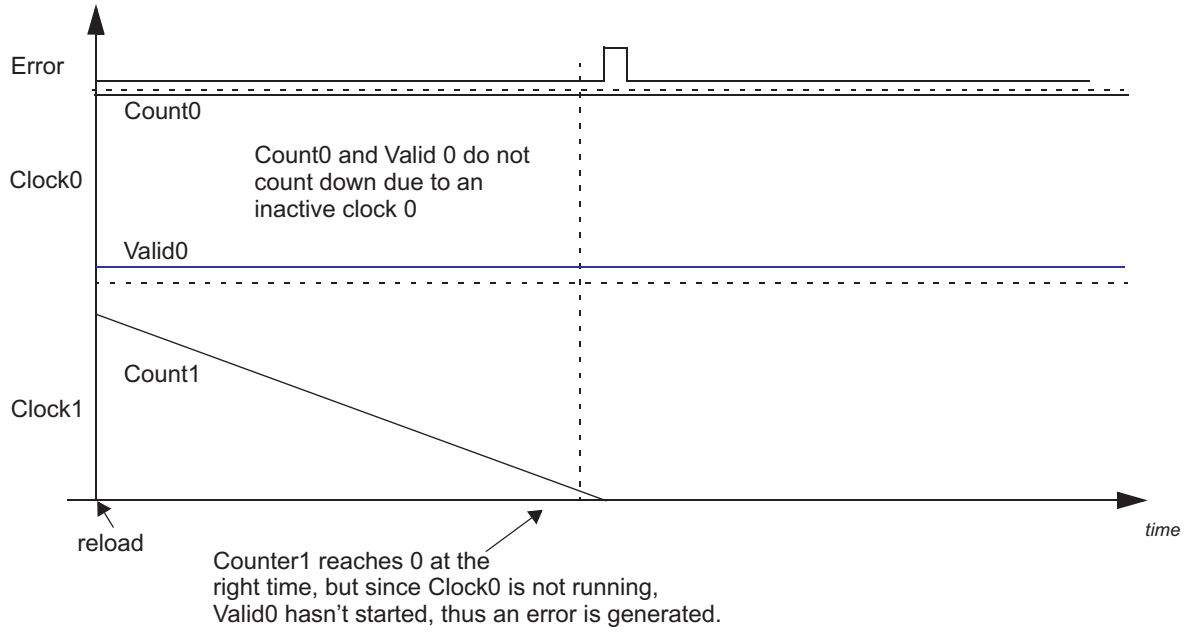


**Figure 6-6. Clock0 Not Present - Results in an Error and Stops Counting**



## 6.3 Registers

### 6.3.1 DCC Base Addresses

**Table 6-1. DCC Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| DCC0Regs | DCC_REGS | 0x0005_E700 | 0x0005_E73F |

### 6.3.1.1 DCC_REGS Registers

Table 6-2 lists the memory-mapped registers for the DCC_REGS. All register offset addresses not listed in Table 6-2 should be considered as reserved locations and the register contents should not be modified.

**Table 6-2. DCC_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | DCCGCTRL | Global Control Register | | Go |
| 4h | DCCREV | DCC Revision Register | | Go |
| 8h | DCCCNTSEED0 | Counter 0 Seed Value | | Go |
| Ch | DCCVALIDSEED0 | Valid 0 Seed Value | | Go |
| 10h | DCCCNTSEED1 | Counter 1 Seed Value | | Go |
| 14h | DCCSTATUS | DCC Status | | Go |
| 18h | DCCCNT0 | Counter 0 Value | | Go |
| 1Ch | DCCVALID0 | Valid Value 0 | | Go |
| 20h | DCCCNT1 | Counter 1 Value | | Go |
| 24h | DCCCLKSRC1 | Clock Source 1 | | Go |
| 28h | DCCCLKSRC0 | Clock Source 0 | | Go |

Complex bit access types are encoded to fit into small table cells. Table 6-3 shows the codes that are used for access types in this section.

**Table 6-3. DCC_REGS Access Type Codes**

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

#### 6.3.1.1.1 DCCGCTRL Register (Offset = 0h) [reset = 5555h]

DCCGCTRL is shown in Figure 6-7 and described in Table 6-4.

Return to Summary Table.

Global Control Register

**Figure 6-7. DCCGCTRL Register**

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

**Table 6-4. DCCGCTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R/W | 0h | Reserved |
| 15-12 | DONEENA | R/W | 5h | DONE Enable<br>Enables/disables the done interrupt signal, but has no effect on the done status flag in DCCSTAT register.<br>0101 The done signal is disabled<br>Others The done signal is enabled<br>Reset type: SYSRSn |
| 11-8 | SINGLESHOT | R/W | 5h | Single-Shot Enable<br>Enables/disables repetitive operation of the DCC.<br>1010 Stop counting when COUNTER0 and VALID0 both reach zero<br>Note: Configure this to 0xA before Enabling DCC<br>Note: All values other than 1010 are reserved<br>Reset type: SYSRSn |
| 7-4 | ERRENA | R/W | 5h | Error Enable<br>Enables/disables the error signal.<br>0101 The error signal is disabled<br>Others The error signal is enabled<br>Reset type: SYSRSn |
| 3-0 | DCCENA | R/W | 5h | DCC Enable<br>Starts and stops the operation of the DCC.<br>0101 Counters are stopped<br>Others Counters are running<br>Reset type: SYSRSn |

#### 6.3.1.1.2 DCCREV Register (Offset = 4h) [reset = 40041003h]

DCCREV is shown in Figure 6-8 and described in Table 6-5.

Return to Summary Table.

DCC Revision Register

**Figure 6-8. DCCREV Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | RESERVED | | 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 |
| RESERVED | | | | | MAJOR | | |
| R-0h | | | | | R-0h | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | MINOR | | | | | |
| R-0h | | R-3h | | | | | |

**Table 6-5. DCCREV Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | RESERVED | R | 0h | Reserved |
| 29-28 | RESERVED | R | 0h | Reserved |
| 27-16 | RESERVED | R | 0h | Reserved |
| 15-11 | RESERVED | R | 0h | Reserved |
| 10-8 | MAJOR | R | 0h | Major Revision Number<br>Represents major changes to the module (e.g. entirely new features are added/changed). The major revision number for this module.<br>Reset type: SYSRSn |
| 7-6 | RESERVED | R | 0h | Reserved |
| 5-0 | MINOR | R | 3h | Minor Revision Number<br>Represents minor changes to the module (e.g. enhancements to existing features). The minor revision number for this module.<br>Reset type: SYSRSn |

#### 6.3.1.1.3 DCCCNTSEED0 Register (Offset = 8h) [reset = 0h]

DCCCNTSEED0 is shown in Figure 6-9 and described in Table 6-6.

Return to Summary Table.

Counter 0 Seed Value

**Figure 6-9. DCCCNTSEED0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | COUNTSEED0 | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | R/W-0h | | | | | | | | | | | | | | | | | | | |

**Table 6-6. DCCCNTSEED0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | RESERVED | R | 0h | Reserved |
| 19-0 | COUNTSEED0 | R/W | 0h | Seed Value for Counter 0<br>Contains the seed value that gets loaded into Counter 0 (Clock Source 0).<br>NOTE: Operating the DCC with '0' in the COUNTSEED0 register will result in undefined operation.<br>Reset type: SYSRSn |

#### 6.3.1.1.4 DCCVALIDSEED0 Register (Offset = Ch) [reset = 0h]

DCCVALIDSEED0 is shown in Figure 6-10 and described in Table 6-7.

Return to Summary Table.

Valid 0 Seed Value

**Figure 6-10. DCCVALIDSEED0 Register**

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

**Table 6-7. DCCVALIDSEED0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-0 | VALIDSEED | R/W | 0h | Seed Value for Valid Duration Counter 0<br>Contains the seed value that gets loaded into the valid duration counter for Clock Source 0.<br>NOTE: Operating the DCC with '0' in the VALIDSEED0 register will result in undefined operation. VALID0 defines a window in which COUNT1 expires. This window is meant to be at least four cycles wide. Do not program a value less than '4' into the VALID0 register.<br>Reset type: SYSRSn |

#### 6.3.1.1.5 DCCCNTSEED1 Register (Offset = 10h) [reset = 0h]

DCCCNTSEED1 is shown in Figure 6-11 and described in Table 6-8.

Return to Summary Table.

Counter 1 Seed Value

**Figure 6-11. DCCCNTSEED1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | COUNTSEED1 | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | R/W-0h | | | | | | | | | | | | | | | | | | | |

**Table 6-8. DCCCNTSEED1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | RESERVED | R | 0h | Reserved |
| 19-0 | COUNTSEED1 | R/W | 0h | Seed Value for Counter 1<br>Contains the seed value that gets loaded into Counter 1 (Clock Source 1).<br>NOTE: Operating the DCC with '0' in the COUNTSEED1 register will result in undefined operation.<br>Reset type: SYSRSn |

#### 6.3.1.1.6 DCCSTATUS Register (Offset = 14h) [reset = 0h]

DCCSTATUS is shown in Figure 6-12 and described in Table 6-9.

Return to Summary Table.

DCC Status

**Figure 6-12. DCCSTATUS Register**

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

**Table 6-9. DCCSTATUS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-2 | RESERVED | R | 0h | Reserved |
| 1 | DONE | R/W | 0h | Single-Shot Done Flag<br>Indicates when single-shot mode is complete without error. Writing a '1' to this bit clears the flag.<br>0 Single-shot mode has not completed.<br>1 Single-shot mode has completed.<br>Reset type: SYSRSn |
| 0 | ERR | R/W | 0h | Error Flag<br>Indicates whether or not an error has occurred. Writing a '1' to this bit clears the flag.<br>0 No errors have occurred.<br>1 An error has occurred.<br>Reset type: SYSRSn |

#### 6.3.1.1.7 DCCCNT0 Register (Offset = 18h) [reset = 0h]

DCCCNT0 is shown in Figure 6-13 and described in Table 6-10.

Return to Summary Table.

Counter 0 Value

**Figure 6-13. DCCCNT0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | COUNT0 | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | | | | | |

**Table 6-10. DCCCNT0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | RESERVED | R | 0h | Reserved |
| 19-0 | COUNT0 | R | 0h | Current Value of Counter 0<br>NOTE: Reads of the counter value may not be exact since the read operation is synchronized to the register clock.<br>Reset type: SYSRSn |

#### 6.3.1.1.8 DCCVALID0 Register (Offset = 1Ch) [reset = 0h]

DCCVALID0 is shown in Figure 6-14 and described in Table 6-11.

Return to Summary Table.

Valid Value 0

**Figure 6-14. DCCVALID0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | VALID0 | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | |

**Table 6-11. DCCVALID0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-0 | VALID0 | R | 0h | Current Value of Valid 0<br>NOTE: Reads of the counter value may not be exact since the read operation is synchronized to the register clock.<br>Reset type: SYSRSn |

#### 6.3.1.1.9 DCCCNT1 Register (Offset = 20h) [reset = 0h]

DCCCNT1 is shown in Figure 6-15 and described in Table 6-12.

Return to Summary Table.

Counter 1 Value

**Figure 6-15. DCCCNT1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | COUNT1 | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | | | | | |

**Table 6-12. DCCCNT1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | RESERVED | R | 0h | Reserved |
| 19-0 | COUNT1 | R | 0h | Current Value of Counter 1<br>NOTE: Reads of the counter value may not be exact since the read operation is synchronized to the register clock.<br>Reset type: SYSRSn |

#### 6.3.1.1.10 DCCCLKSRC1 Register (Offset = 24h) [reset = 5002h]

DCCCLKSRC1 is shown in Figure 6-16 and described in Table 6-13.

Return to Summary Table.

Clock Source 1

**Figure 6-16. DCCCLKSRC1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| KEY | | | | RESERVED | | | | | | | | CLKSRC1 | | | |
| R/W-5h | | | | R-0h | | | | | | | | R/W-2h | | | |

**Table 6-13. DCCCLKSRC1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-12 | KEY | R/W | 5h | Enables or Disables Clock Source Selection for COUNT1<br>1010 The CLKSRC field selects the clock source for COUNT1.<br>Others Clock source selection is disabled. The secondary oscillator is selected for COUNT1.<br>Reset type: SYSRSn |
| 11-4 | RESERVED | R | 0h | Reserved |
| 3-0 | CLKSRC1 | R/W | 2h | Clock Source Select for Counter 1<br>Specifies the clock source for COUNT1, when the KEY field enables this feature.<br>0000 Clock Source 0 (PLLRAWCLK) is selected for COUNT1.<br>Note: Configure this value to 0x0 before enabling DCC<br>Note: All values other than 0000 are reserved.<br>Reset type: SYSRSn |

#### 6.3.1.1.11 DCCCLKSRC0 Register (Offset = 28h) [reset = 1h]

DCCCLKSRC0 is shown in Figure 6-17 and described in Table 6-14.

Return to Summary Table.

Clock Source 0

**Figure 6-17. DCCCLKSRC0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | | | | | | CLKSRC0 | | | |
| R-0h | | | | | | | | | | | | R/W-1h | | | |

**Table 6-14. DCCCLKSRC0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-4 | RESERVED | R | 0h | Reserved |
| 3-0 | CLKSRC0 | R/W | 1h | Clock Source Select for Counter 0<br>0000 Clock Source 0(XTAL) is selected for COUNT0.<br>0001 Clock Source 1(INTOSC1) is selected for COUNT0.<br>0010 Clock Source 2(INTOSC2) is selected for COUNT0.<br>Note: Reference Clock for the PLL (OSCCLKSRCSEL) and the DCCCLKSRC0 selected should be the same, meaning if OSCCLKSRCSEL=0x1 (XTAL) then DCCCLKSRC0.CLKRSC0=0x0 (XTAL).<br>Note: All other values are reserved.<br>Reset type: SYSRSn |

# CLA Program ROM (CLAPROMCRC)

The CLAPROMCRC is a feature which calculates a CRC-32 value of a configurable block of data in the CLA program ROM space.

**Topic** **Page**

## 7.1 Overview

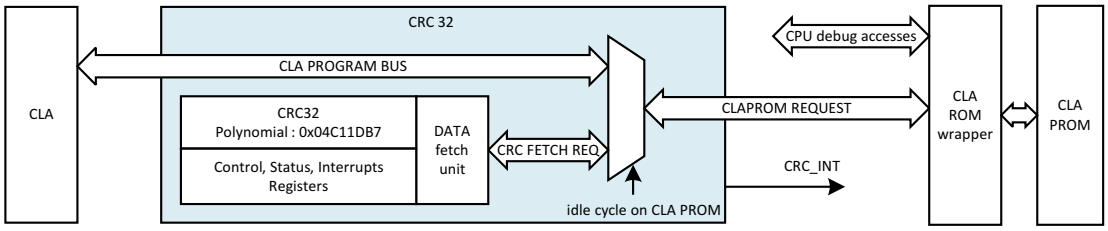
The CLAPROMCRC is a feature which calculates a CRC-32 on a configurable block of memory in the CLA program ROM space. Neither the C28x nor the CLA have the capability to compute a CRC on the CLA program ROM. The CLAPROMCRC solves this problem by calculating a CRC-32 in a non-intrusive manner. It is a hardware CRC-32 module which fetches the CLA program ROM during idle cycles (when the CLA is not accessing the ROM on the CLA program bus), and calculates the CRC-32 in order to perform a code integrity check. It then compares the result with a golden CRC-32 value and indicates a pass or fail condition.

## 7.2 Functional Description

The CLAPROMCRC module can be configured and initiated by either the C28x or the CLA. Once the CLAPROMCRC is initiated, the processing core can switch back to the application and service the CLAPROMCRC via interrupt. An interrupt can trigger the C28x and CLA and signal that the CRC-32 calculation and comparison has completed (CRCDONE). The CLAPROMCRC module waits for CLA program bus idle cycles in order to access the CLA program ROM. This ensures that the CRC-32 hardware calculation does not affect code execution and performance on the CLA.

Below is a functional diagram of the CLAPROMCRC module.

**Figure 7-1. CLAPROMCRC Functional Diagram**




Once the CLAPROMCRC is initiated by the C28x or CLA, the module will snoop for idle cycles to fetch program memory and calculate the CRC-32. It will begin fetching from the START_ADDRESS and stop after the CRC-32 has been calculated for the required number of bytes specified by BLOCK_SIZE. The CRC-32 can be calculated on as little as 1KB of data up to the size of the CLA program ROM in increments of 1KB.

The CLAPROMCRC module fetches 32 bits of program data on every fetch and takes four cycles to calculate the CRC-32 value for each fetch. For example, in order to calculate a CRC-32 on 1KB of program data, the module requires 256 idle cycles if each fetch access is spaced four cycles apart. Therefore, to calculate the CRC-32 on a 1KB block of program data requires a minimum of 1024 cycles. The CRC polynomial used in the hardware calculation is 0x04C11DB7.

### *7.2.1 Start Address*

The start address must be aligned to a 1KB boundary (512 word boundary). If a non-aligned address is programmed, then the 9 least significant bits are ignored to align the address. The start address written to the CRC32_STARTADDRESS register must be the memory address corresponding to the CLA memory map.

**NOTE:** If the start address is not within the CLA program ROM space of the CLA memory map, then the calculation will not be initiated.

### *7.2.2 Seed*

The seed is the initial value used for the CRC-32 calculation. Therefore, the result will vary with the initial seed. The seed can be written into the CRC32_SEED register.

### 7.2.3 Halt

Once the CRC calculation is triggered, it can be halted by setting the HALT bit in the configuration register (CRC32_CONTROLREG). When this bit is cleared, the CRC calculation will resume from the current address.

### 7.2.4 Result and Comparison

To protect the contents of the CLA program ROM, which includes TI proprietary code, the result of the CRC-32 must be protected. To protect the result, the actual CRC-32 value as it is calculated cycle by cycle is not made visible. However, the final result of the block is populated in the CRC32_CRCRESULT register. This is the reason for the block size requirements which must specifically be a minimum of 1KB and an increment of 1KB.

To check the correctness of the CRC-32 result, a golden CRC value should be written in the CRC32_GOLDENCRC register. After the CRC-32 calculation is completed by the module for the configured block size, the golden CRC will be compared with the final result and the module will then set a pass or fail bit accordingly in the status register (CRC32_STATUSREG).

## 7.3 Registers

### 7.3.1 CLA Program ROM Base Addresses

**Table 7-1. CLA PROM CRC Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| ClaPromCrc32Regs | CLA_PROM_CRC32_REGS | 0x000061C0 | 0x000061DF |

### 7.3.1.1 CLA_PROM_CRC32_REGS Registers

Table 7-2 lists the memory-mapped registers for the CLA_PROM_CRC32_REGS. All register offset addresses not listed in Table 7-2 should be considered as reserved locations and the register contents should not be modified.

**Table 7-2. CLA_PROM_CRC32_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | CRC32_CONTROLREG | CRC32-Control Register | EALLOW | Go |
| 2h | CRC32_STARTADDRESS | CRC32-Start address register | EALLOW | Go |
| 4h | CRC32_SEED | CRC32-Seed Register | EALLOW | Go |
| 6h | CRC32_STATUSREG | CRC32-Status Register | | Go |
| 8h | CRC32_CRCRESULT | CRC32-CRC result Register | | Go |
| Ah | CRC32_GOLDENCRC | CRC32-Golden CRC register | | Go |
| 18h | CRC32_INTEN | CRC32-Interrupt enable register | EALLOW | Go |
| 1Ah | CRC32_FLG | CRC32-Interrupt Flag Register | | Go |
| 1Ch | CRC32_CLR | CRC32-Interrupt Clear Register | | Go |
| 1Eh | CRC32_FRC | CRC32-Interrupt Force Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 7-3 shows the codes that are used for access types in this section.

**Table 7-3. CLA_PROM_CRC32_REGS Access Type Codes**

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

#### 7.3.1.1.1 CRC32_CONTROLREG Register (Offset = 0h) [reset = 0h]

CRC32_CONTROLREG is shown in Figure 7-2 and described in Table 7-4.

Return to Summary Table.

CRC32-Control Register

**Figure 7-2. CRC32_CONTROLREG Register**

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

**Table 7-4. CRC32_CONTROLREG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-23 | RESERVED | R=0 | 0h | Reserved |
| 22-16 | BLOCKSIZE | R/W | 0h | Block size:<br>0x0 : 1 KB (default)<br>0x1 : 2 KB<br>0x2 : 3 KB<br>...<br>0x7F : 128KB<br>Note : If the value written to this register is greater than size of ROM then it is internally set to max allowed block size<br>Reset type: CPU1.SYSRSn |
| 15-9 | RESERVED | R=0 | 0h | Reserved |
| 8 | HALT | R/W | 0h | Halt Bit :<br>0 : CRC calaculation will resume from where is has hanlted<br>1 : CRC calculation will be halted<br>Notes:<br>This bit has effect only after CRC calc is triggered by writing to START Bit<br>Reset type: CPU1.SYSRSn |
| 7-5 | RESERVED | R=0 | 0h | Reserved |
| 4 | FREE_SOFT | R/W | 0h | emulation control bit : This bit controls behaviour of CRC calculation during emulations<br>0 : Soft, CRC module stops on MCLA dedug suspend.<br>1 : Free, CRC calucation is not affected by DEBUG HALT of CLA<br>Reset type: CPU1.SYSRSn |
| 3-1 | RESERVED | R=0 | 0h | Reserved |

**Table 7-4. CRC32_CONTROLREG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | START | R=0/W=1 | 0h | Start Bit:<br>0: No effect<br>1: will start the CRC calulations<br>Notes:<br>Setting this anytime during the CRC caluclation will reset and re-start the CRC calulation.<br>Reset type: CPU1.SYSRSn |

#### 7.3.1.1.2 CRC32_STARTADDRESS Register (Offset = 2h) [reset = 0h]

CRC32_STARTADDRESS is shown in Figure 7-3 and described in Table 7-5.

Return to Summary Table.

CRC32-Start address register

**Figure 7-3. CRC32_STARTADDRESS Register**

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

**Table 7-5. CRC32_STARTADDRESS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | START_ADDRESS | R/W | 0h | START_ADDRESS defines starting point for CRC32 calculation. Note that it is CLA address<br>Reset type: CPU1.SYSRSn |

#### 7.3.1.1.3 CRC32_SEED Register (Offset = 4h) [reset = 0h]

CRC32_SEED is shown in Figure 7-4 and described in Table 7-6.

Return to Summary Table.

CRC32-Seed Register

**Figure 7-4. CRC32_SEED Register**

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

**Table 7-6. CRC32_SEED Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | SEED | R/W | 0h | Initial value of CRC, this value is coiped to the CRC register on triggering CRC calculation by writing to START bit.<br>Reset type: CPU1.SYSRSn |

#### 7.3.1.1.4 CRC32_STATUSREG Register (Offset = 6h) [reset = 0h]

CRC32_STATUSREG is shown in Figure 7-5 and described in Table 7-7.

Return to Summary Table.

CRC32-Status Register

**Figure 7-5. CRC32_STATUSREG Register**

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

**Table 7-7. CRC32_STATUSREG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RUNSTATUS | R | 0h | Status bit:<br>0 : CRC module is IDLE<br>1 : CRC module is Active<br>Reset type: CPU1.SYSRSn |
| 30-24 | RESERVED | R=0 | 0h | Reserved |
| 23 | CRCCHECKSTATUS | R | 0h | CRC pass/fail status bit:<br>0 : PASS<br>1 : FAIL<br>Note: Comparion is enabled only after CRC calc is completed<br>Reset type: CPU1.SYSRSn |
| 22-16 | RESERVED | R=0 | 0h | Reserved |
| 15-0 | CURRENTADDR | R | 0h | The current address of data fetch unit - this is 32 bit aligned offset address of ROM<br>Reset type: CPU1.SYSRSn |

#### 7.3.1.1.5 CRC32_CRCRESULT Register (Offset = 8h) [reset = 0h]

CRC32_CRCRESULT is shown in Figure 7-6 and described in Table 7-8.

Return to Summary Table.

CRC32-CRC result Register

**Figure 7-6. CRC32_CRCRESULT Register**

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

**Table 7-8. CRC32_CRCRESULT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | CRCRESULT | R | 0h | CRC result register<br>Reset type: CPU1.SYSRSn |

#### 7.3.1.1.6 CRC32_GOLDENCRC Register (Offset = Ah) [reset = 0h]

CRC32_GOLDENCRC is shown in Figure 7-7 and described in Table 7-9.

Return to Summary Table.

CRC32-Golden CRC register

**Figure 7-7. CRC32_GOLDENCRC Register**

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

**Table 7-9. CRC32_GOLDENCRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | GOLDENCRC | R/W | 0h | Golden CRC register:<br>Value written to this register is compared with CRCRESULT when CRCDONE bit is set.<br>After the CRC is done, GOLDENCRC will be compared with CRCRESULT and the CRCCHECKSTATUS bit will be updated.<br>Reset type: CPU1.SYSRSn |

#### 7.3.1.1.7 CRC32_INTEN Register (Offset = 18h) [reset = 0h]

CRC32_INTEN is shown in Figure 7-8 and described in Table 7-10.

Return to Summary Table.

CRC32-Interrupt enable register

**Figure 7-8. CRC32_INTEN Register**

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

**Table 7-10. CRC32_INTEN Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-2 | RESERVED | R=0 | 0h | Reserved |
| 1 | CRCDONE | R/W | 0h | 0 CRCDONE Interrupt disabled<br>1 CRCDONE Interrupt enabled<br>Reset type: CPU1.SYSRSn |
| 0 | RESERVED | R=0 | 0h | Reserved |

#### 7.3.1.1.8 CRC32_FLG Register (Offset = 1Ah) [reset = 0h]

CRC32_FLG is shown in Figure 7-9 and described in Table 7-11.

Return to Summary Table.

CRC32-Interrupt Flag Register

**Figure 7-9. CRC32_FLG Register**

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

**Table 7-11. CRC32_FLG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-2 | RESERVED | R=0 | 0h | Reserved |
| 1 | CRCDONE | R | 0h | Error Interrupt Status flag<br>0 CRC calulation is in progress or CRC module is idle.<br>1 CRC calulation is done.<br>Reset type: CPU1.SYSRSn |
| 0 | INT | R | 0h | Global Interrupt Status flag<br>0 No interrupt generated<br>1 Interrupt was generated<br>Reset type: CPU1.SYSRSn |

#### 7.3.1.1.9 CRC32_CLR Register (Offset = 1Ch) [reset = 0h]

CRC32_CLR is shown in Figure 7-10 and described in Table 7-12.

Return to Summary Table.

CRC32-Interrupt Clear Register

**Figure 7-10. CRC32_CLR Register**

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

**Table 7-12. CRC32_CLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-2 | RESERVED | R=0 | 0h | Reserved |
| 1 | CRCDONE | R=0/W=1 | 0h | Clear CRCDONE interrupt flag<br>0 No effect<br>1 Clears the CRCDONE interrupt flag<br>Reset type: CPU1.SYSRSn |
| 0 | INT | R=0/W=1 | 0h | Global Interrupt Clear<br>0 No effect<br>1 Clears the interrupt flag and enables further interrupts to be generated if an event flags is set to 1.<br>Reset type: CPU1.SYSRSn |

#### 7.3.1.1.10 CRC32_FRC Register (Offset = 1Eh) [reset = 0h]

CRC32_FRC is shown in Figure 7-11 and described in Table 7-13.

Return to Summary Table.

CRC32-Interrupt Force Register

**Figure 7-11. CRC32_FRC Register**

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

**Table 7-13. CRC32_FRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-2 | RESERVED | R=0 | 0h | Reserved |
| 1 | CRCDONE | R=0/W=1 | 0h | Force CRCDONE interrupt flag<br>0 No effect<br>1 Force CRCDONE interrupt flag<br>Reset type: CPU1.SYSRSn |
| 0 | RESERVED | R=0 | 0h | Reserved |

# General-Purpose Input/Output (GPIO)

The GPIO module controls the device's digital and analog I/O multiplexing, which uses shared pins to maximize application flexibility. The pins are named by their general-purpose I/O name (for example, GPIO0, GPIO25, GPIO58). These pins can be individually selected to operate as digital I/O (also called GPIO mode), or connected to one of several peripheral I/O signals. The input signals can be qualified to remove unwanted noise.

**Topic** **Page**

## 8.1 GPIO Overview

Up to twelve independent peripheral signals are multiplexed on a single GPIO-enabled pin in addition to the CPU-controlled I/O capability. Each pin output can be controlled by a peripheral, the CPU, or the CLA. There are two I/O ports:

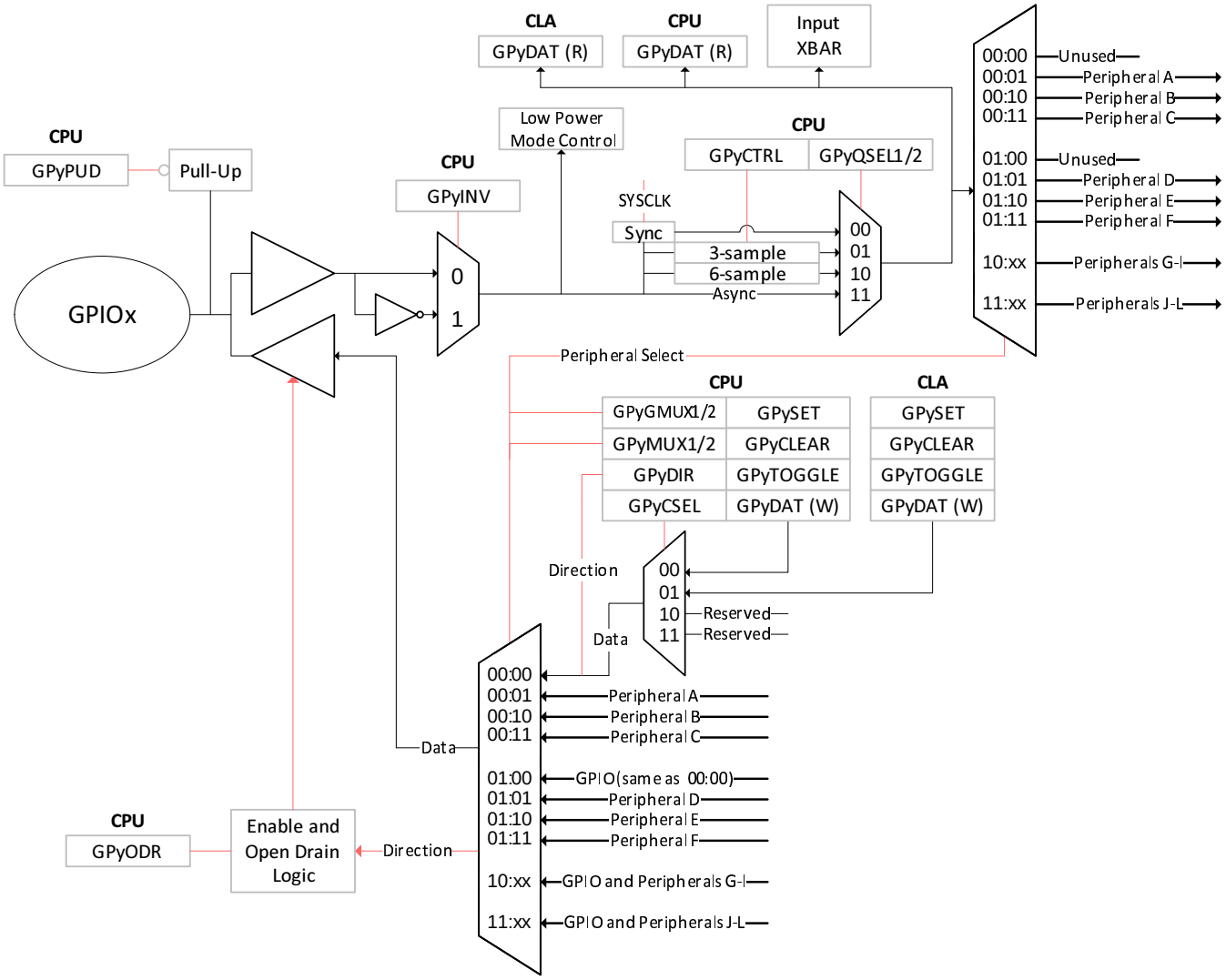
- Port A consists of GPIO0-GPIO31
- Port B consists of GPIO32-GPIO59

The analog signals on this device are multiplexed with digital inputs. These analog IO (AIO) pins do not have digital output capability. They are assigned to a single port:

- Port H consists of GPIO224-GPIO247

Figure 8-1 shows the GPIO logic for a single pin.

**Figure 8-1. GPIO Logic for a Single Pin**




There are two key features to note in this diagram. The first is that the input and output paths are entirely separate, connecting only at the pin. The second is that peripheral muxing takes place far from the pin. As a result, it is always possible for the CPU and CLA to read the physical state of the pin independent of mastering and peripheral muxing. Likewise, external interrupts can be generated from peripheral activity. All pin options (such as input qualification and pull-ups) are valid for all masters and peripherals.

**NOTE:** JTAG uses a different signal path that does not support inversion or qualification.

**NOTE:** GPIO22 and GPIO23 are in a special analog mode at reset, and must be reconfigured for GPIO use by clearing their bits in GPAAMSEL. GPIO23's maximum toggle frequency is limited. The exact limit is TBD, but estimated to be 12 MHz.

**NOTE:** GPIO18/X2 has different timings due to the load placed on it by the oscillator circuit. For information on using GPIO18/X2 as a GPIO, please see the device datasheet and the Clocking section of this document.

**NOTE:** If digital signals with sharp edges (high dv/dt) are connected to the AIOs, cross-talk can occur with adjacent analog signals. Therefore, the user should limit the edge rate of signals connected to AIOs if adjacent channels are being used for analog functions.

## 8.2 Configuration Overview

I/O pin configuration consists of several steps:

1. **Plan the device pin-out**
   Make a list of all required peripherals for the application. Using the peripheral mux information in the device data manual, choose which GPIOs to use for the peripheral signals. Decide which of the remaining GPIOs to use as inputs and outputs for the CPU and CLA.
   Once the peripheral muxing has been chosen, it should be implemented by writing the appropriate values to the GPyMUX1/2 and GPyGMUX1/2 registers. When changing the GPyGMUX value for a pin, always set the corresponding GPyMUX bits to zero first to avoid glitching in the muxes. By default, all pins are general-purpose I/Os, not peripheral signals.
2. **(Optional) Enable internal pull-up resistors**
   To enable or disable the pull-up resistors, write to the appropriate bits in the GPIO pull-up disable registers (GPyPUD). All pull-ups are disabled by default. Pull-ups can be used to keep input pins in a known state when there is no external signal driving them.
3. **Select input qualification**
   If the pin will be used as an input, specify the required input qualification, if any. The input qualification sampling period is selected in the GPyCTRL registers, while the type of qualification is selected in the GPyQSEL1 and GPyQSEL2 registers. By default, all qualification is synchronous with a sampling period equal to SYSCLK. For an explanation of input qualification, see section Section 8.4.
4. **Select the direction of any general-purpose I/O pins**
   For each pin configured as a GPIO, specify the direction of the pin as either input or output using the GPyDIR registers. By default, all GPIO pins are inputs. Before changing a pin to an output, load the output latch with the value to be driven by writing that value to the GPySET, GPyCLEAR, or GPyDAT registers. Once the latch is loaded, write to GPyDIR to change the pin direction. By default, all output latches are zero.
5. 
6. **Select low-power mode wake-up sources**
   GPIOs 0-59 can be used to wake the system up from Halt mode. To select one or more GPIOs for wake-up, write to the appropriate bits in the GPIOLPMSEL0 and GPIOLPMSEL1 registers. These registers are part of the CPU system register space. For more information on low-power modes and GPIO wake-up, see the Low Power Modes section in the *System Control and Interrupts* chapter.
7. **Select external interrupt sources**
   Configuring external interrupts is a two-step process. First, the interrupts themselves must be enabled and their polarity must be configured via the XINTnCR registers. Second, the XINT1-5 GPIO pins must be set by selecting the sources for Input X-BAR signals 4, 5, 6, 13, and 14, respectively. For more information on the Input X-BAR architecture, see the *Input Crossbar* chapter.
8. **Select pin masters**
   Once the GPIOs are configured, they can be individually assigned to the CLA via the GPyCSEL1-4

registers. Assigning an output pin to the CLA causes the CLA's GPySET, GPyCLEAR, GPyTOGGLE, and GPyDAT registers to control the state of that pin, while the CPU's data registers are ignored. Assigning an input pin to the CLA has no effect.

## 8.3 Digital General-Purpose I/O Control

The state of the pins that are configured as GPIO outputs can be changed by using the following registers:

- **Data Registers (GPyDAT)**

  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), reading the corresponding bit in the data register gives the current state of the pin after qualification.

  Writing to a data register clears or sets the corresponding output latch. If the pin is configured as a general purpose output (GPIO output) the pin will also be driven either low or high. If the pin is not configured as a GPIO output, the written value will be latched, but the pin will not be driven. If the pin is later configured as a GPIO output, the latched value be driven. Since reading the data register always returns the input state of the pins, read-modify-write operations can inadvertently change the state of the output latches. This is especially problematic for open-drain pins. Also, the CPU pipelining can cause sequential read-modify-write operations to fail, as in the code below:

```
GpioDataRegs.GPADAT.bit.GPIO1 = 1;  //Performs read-modify-write of GPADAT
GpioDataRegs.GPADAT.bit.GPIO2 = 1;  //Also performs a read-modify-write of GPADAT.
                                    //GPADAT gets the old value of GPIO1 due to the CPU pipeline
```

  For these reasons, it is better to use the GPySET, GPyCLEAR, and GPyTOGGLE registers to control the output latches. These registers always read back as zero and writes of zero have no effect, so read-modify-write operations on them are safe.

- **Set, Clear, and Toggle Registers (GPySET, GPyCLEAR, and GPyTOGGLE)**

  The set, clear, and toggle registers are used to control the state of the specified GPIO pins without disturbing other pins. Each I/O port has one set, one clear, and one toggle register and each register bit corresponds to one GPIO pin. These registers always read back as zero. If a pin is configured as an output, writing a one to the corresponding bit in the set register will set the output latch high and drive the pin high. Writing a one to the corresponding bit in the clear register will set the output latch low and drive the pin low. Writing a one to the corresponding bit in the toggle register will invert the state of the output latch and drive the pin accordingly. If the pin is not configured as a GPIO output, then the output latch will be modified but the pin will not be driven. If the pin is later configured as a GPIO output, the latched value will be driven.

## 8.4 Input Qualification

The input qualification scheme has been designed to be very flexible. You can select the type of input qualification for each GPIO pin by configuring the GPyQSEL1 and GPyQSEL2 registers. In the case of a GPIO input pin, the qualification can be specified as only synchronize to SYSCLK 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 SYSCLK or qualified by a sampling window. The remainder of this section describes the options available.

### *8.4.1 No Synchronization (Asynchronous Input)*

This mode is used for peripherals where input synchronization is not required or the peripheral itself performs the synchronization. Examples include communication ports SCI, SPI, and $I^2C$. In addition, it may be desirable to have the ePWM trip zone ($\overline{TZn}$) signals function independent of the presence of SYSCLK.

The asynchronous option is not valid if the pin is used as a general purpose digital input pin (GPIO). If the pin is configured as a GPIO input and the asynchronous option is selected then the qualification defaults to synchronization to SYSCLK as described in Section 8.4.2.

**NOTE:** Using input synchronization when the peripheral itself performs the synchronization may cause unexpected results. The user should ensure that the GPIO pin is configured for asynchronous in this case.

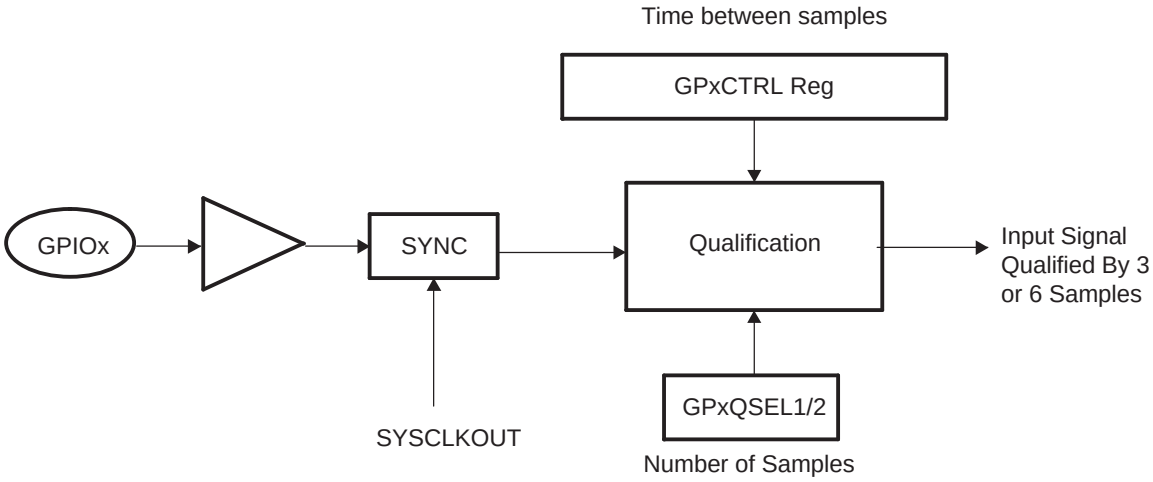
### 8.4.2 Synchronization to SYSCLK Only

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 (SYSCLK). Because the incoming signal is asynchronous, it can take up to a SYSCLK period of delay in order for the input to the MCU to be changed. No further qualification is performed on the signal.

### 8.4.3 Qualification Using a Sampling Window

In this mode, the signal is first synchronized to the system clock (SYSCLK) and then qualified by a specified number of cycles before the input is allowed to change. Figure 8-2 and Figure 8-3 show how the input qualification is performed to eliminate unwanted noise. Two parameters are specified by the user for this type of qualification: 1) the sampling period, or how often the signal is sampled, and 2) the number of samples to be taken.

**Figure 8-2. Input Qualification Using a Sampling Window**




**Time between samples (sampling period):**

To qualify the signal, the input signal is sampled at a regular period. The sampling period is specified by the user and determines the time duration between samples, or how often the signal will be sampled, relative to the CPU clock (SYSCLK).

The sampling period is specified by the qualification period (QUALPRDn) bits in the GPyCTRL 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 GPyCTRL[QUALPRDn] setting.

**Table 8-1. Sampling Period**

| | Sampling Period |
|---|---|
| If GPyCTRL[QUALPRDn] = 0 | $1 \times T_{SYSCLK}$ |
| If GPyCTRL[QUALPRDn] ≠ 0 | $2 \times$ GPyCTRL[QUALPRDn] $\times T_{SYSCLK}$ |
| | Where $T_{SYSCLK}$ is the period in time of SYSCLK |

**Table 8-2. Sampling Frequency**

| | Sampling Frequency |
|---|---|
| If GPyCTRL[QUALPRDn] = 0 | $f_{SYSCLK}$ |
| If GPyCTRL[QUALPRDn] ≠ 0 | $f_{SYSCLK} \times 1 \div (2 \times$ GPyCTRL[QUALPRDn]) |
| | Where $f_{SYSCLK}$ is the frequency of SYSCLK |

From these equations, the minimum and maximum time between samples can be calculated for a given SYSCLK frequency:

**Example: Maximum Sampling Frequency:**

If GPyCTRL[QUALPRDn] = 0

then the sampling frequency is $f_{SYSCLK}$

If, for example, $f_{SYSCLK}$ = 60 MHz

then the signal will be sampled at 60 MHz or one sample every 16.67 ns.

**Example: Minimum Sampling Frequency:**

If GPyCTRL[QUALPRDn] = 0xFF (255)

then the sampling frequency is $f_{SYSCLK} \times 1 \div (2 \times \text{GPyCTRL[QUALPRDn]})$

If, for example, $f_{SYSCLK}$ = 60 MHz

then the signal will be sampled at 60 MHz × 1 ÷ (2 × 255) or one sample every 8.5 μs.

### Number of samples:

The number of times the signal is sampled is either three samples or six samples as specified in the qualification selection (GPyQSEL1 and GPyQSEL2) registers. When three or six consecutive cycles are the same, the input change will be passed through to the MCU.

### Total Sampling Window Width:

The sampling window is the time during which the input signal will be sampled as shown in Figure 8-3. By using the equation for the sampling period along with the number of samples to be taken, the total width of the window can be determined.

For the input qualifier to detect a change in the input, the level of the signal must be stable for the duration of the sampling window width or longer.

The number of sampling periods within the window is always one less than the number of samples taken. For a three-sample window, the sampling window width is two sampling periods wide where the sampling period is defined in Table 8-1. Likewise, for a six-sample window, the sampling window width is five sampling periods wide. Table 8-3 and Table 8-4 show the calculations that can be used to determine the total sampling window width based on GPyCTRL[QUALPRDn] and the number of samples taken.

**Table 8-3. Case 1: Three-Sample Sampling Window Width**

| | Total Sampling Window Width |
|---|---|
| If GPyCTRL[QUALPRDn] = 0 | $2 \times T_{SYSCLK}$ |
| If GPyCTRL[QUALPRDn] ≠ 0 | $2 \times 2 \times \text{GPyCTRL[QUALPRDn]} \times T_{SYSCLK}$ |
| | Where $T_{SYSCLK}$ is the period in time of SYSCLK |

**Table 8-4. Case 2: Six-Sample Sampling Window Width**

| | Total Sampling Window Width |
|---|---|
| If GPyCTRL[QUALPRDn] = 0 | $5 \times T_{SYSCLK}$ |
| If GPyCTRL[QUALPRDn] ≠ 0 | $5 \times 2 \times \text{GPyCTRL[QUALPRDn]} \times T_{SYSCLK}$ |
| | Where $T_{SYSCLK}$ is the period in time of SYSCLK |

**NOTE:** The external signal change is asynchronous with respect to both the sampling period and SYSCLK. Due to the asynchronous nature of the external signal, the input should be held stable for a time greater than the sampling window width to make sure the logic detects a change in the signal. The extra time required can be up to an additional sampling period + $T_{SYSCLK}$.

The required duration for an input signal to be stable for the qualification logic to detect a change is described in the device-specific data manual.

**Example Qualification Window:**

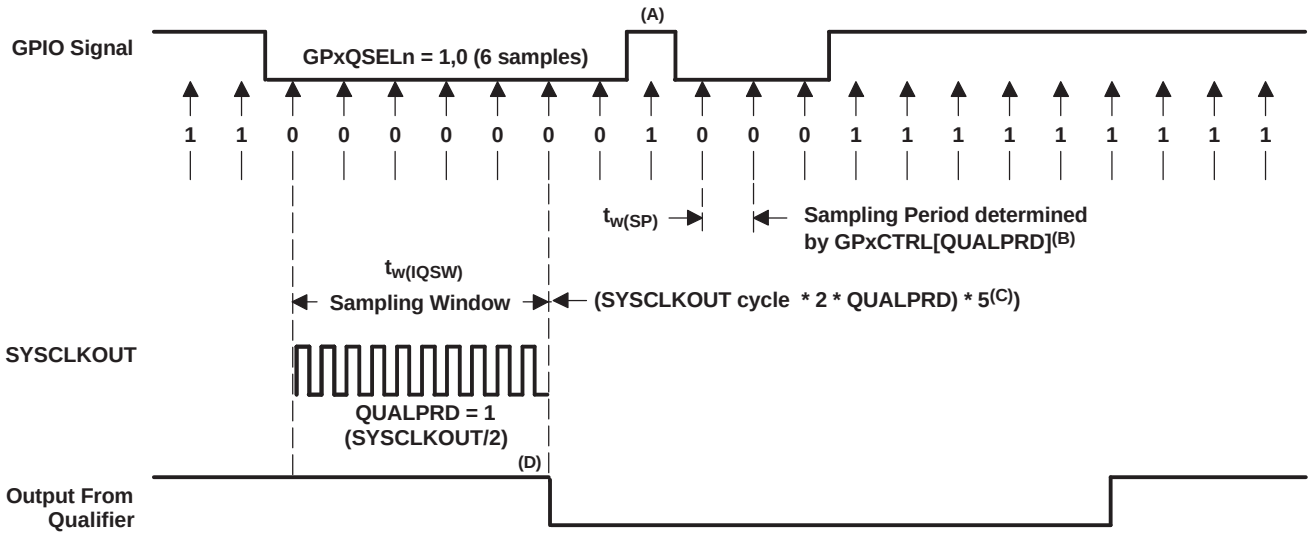
For the example shown in Figure 8-3, the input qualification has been configured as follows:

- GPyQSEL1/2 = 1,0. This indicates a six-sample qualification.
- GPyCTRL[QUALPRDn] = 1. The sampling period is $t_w(SP) = 2 \times$ GPyCTRL[QUALPRDn] $\times T_{SYSCLK}$ .

This configuration results in the following:

- The width of the sampling window is: .
  $t_w(IQSW) = 5 \times t_w(SP) = 5 \times 2 \times$ GPyCTRL[QUALPRDn] $\times T_{SYSCLK}$ or $5 \times 2 \times T_{SYSCLK}$
- If, for example, $T_{SYSCLK}$ = 16.67 ns, then the duration of the sampling window is:
  $t_w(IQSW) = 5 \times 2 \times 16.67$ ns =166.7 ns.
- To account for the asynchronous nature of the input relative to the sampling period and SYSCLK, up to an additional sampling period, $t_w(SP)$, + $T_{SYSCLK}$ may be required to detect a change in the input signal. For this example:
  $t_w(SP) + T_{SYSCLK} = 333.4$ ns + 166.67 ns = 500.1 ns
- In Figure 8-3, the glitch (A) is shorter then the qualification window and will be ignored by the input qualifier.

**Figure 8-3. Input Qualifier Clock Cycles**




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 GPxQSELn 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.

## 8.5 GPIO and Peripheral Muxing

Up to twelve different peripheral functions are multiplexed to each pin along with a general-purpose input/output (GPIO) function. This allows you to choose the peripheral mix and pinout that will work best for your particular application. Refer to the device datasheet for muxing combinations and definitions.

For example, the multiplexing for the GPIO 22 pin is controlled by writing to GPAGMUX2[13:12] and GPAMUX2[13:12]. By writing to these bits, GPIO 22 can be configured as either a general-purpose digital I/O or one of five different peripheral functions. The options are shown in Table 8-5.

**Table 8-5. GPIO and Peripheral Muxing**

| GPAGMUX1[13:12] | GPAMUX1[13:12] | Pin functionality |
|---|---|---|
| 00 | 00 | GPIO |
| 00 | 01 | EQEPS_1 |
| 00 | 11 | SCITXD_B |
| 01 | 00 | GPIO |
| 01 | 10 | SPICLK_B |
| 01 | 11 | SDD_4 |
| 10 | 00 | GPIO |
| 10 | 01 | LINTX_A |
| 11 | 00 | GPIO |
| All others | | Reserved |

Different devices have different pinouts. If a peripheral is not available on a particular devices, its associated muxing options are reserved on that device and should not be used.

Note: If you select a reserved GPIO mux configuration that is not mapped to either a peripheral or GPIO mode, the state of the pin will be undefined and the pin may be driven. Unimplemented configurations are for future expansion and must not be selected. In the device mux table (located in the datasheet), these options are indicated as Reserved or left blank.

Some peripherals can be assigned to more than one pin via the mux registers. For example, OUTPUTXBAR1 can be assigned to GPIOs 2, 24, or (on some packages) 34 or 58, depending on individual system requirements. An example of this is shown in Table 8-6.

**Table 8-6. Peripheral Muxing (multiple pins assigned)**

| GMUX Configuration | MUX Configuration | |
|---|---|---|
| Choice 1 GPIO2 | GPAGMUX1[5:4]=01 | GPAMUX1[5:4]=01 |
| or Choice 2 GPIO24 | GPAGMUX2[17:16]=00 | GPAMUX2[17:16]=01 |
| or Choice 3 GPIO34 | GPBGMUX1[5:4]=00 | GPBMUX1[5:4]=01 |
| or Choice 4 GPIO58 | GPBGMUX2[21:20]=01 | GPBMUX2[21:20]=01 |

If no pin is configured as an input to a peripheral, or if more than one pin is configured to an input for the same peripheral, then that input will be set to a hard-wired default value.

## 8.6 Internal Pull-up Configuration Requirements

On reset, GPIOs are in input mode and have their internal pull-ups disabled. An un-driven input can float to a mid-rail voltage and cause wasted shoot-through current on the input buffer. The user should always put each GPIO in one of these configurations:

- Input mode and driven on the board by another component to a level above $V_{IH}$ or below $V_{IL}$
- Input mode with GPIO internal pull-up enabled
- Output mode

On devices with fewer than 144 pins, some GPIOs are not bonded out to the package. The pull-ups for any unbonded GPIO must be enabled to prevent floating inputs. TI provides functions in C2000Ware which users can call to enable the pull-ups on all unbonded GPIOs for the package they are using. Users should take care to avoid disabling these pull-ups in their application code.

## 8.7 Registers

### 8.7.1 GPIO Base Addresses

**Table 8-7. GPIO Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| GpioCtrlRegs | GPIO_CTRL_REGS | 0x0000_7C00 | 0x0000_7EFF |
| GpioDataRegs | GPIO_DATA_REGS | 0x0000_7F00 | 0x0000_7FFF |

### 8.7.1.1 GPIO_CTRL_REGS Registers

Table 8-8 lists the memory-mapped registers for the GPIO_CTRL_REGS. All register offset addresses not listed in Table 8-8 should be considered as reserved locations and the register contents should not be modified.

**Table 8-8. GPIO_CTRL_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | GPACTRL | GPIO A Qualification Sampling Period (GPIO0 to GPIO31) | EALLOW | Go |
| 2h | GPAQSEL1 | GPIO A Qualification Type (GPIO0 to GPIO15) | EALLOW | Go |
| 4h | GPAQSEL2 | GPIO A Qualification Type (GPIO16 to GPIO31) | EALLOW | Go |
| 6h | GPAMUX1 | GPIO A Peripheral Mux (GPIO0 to GPIO15) | EALLOW | Go |
| 8h | GPAMUX2 | GPIO A Peripheral Mux (GPIO16 to GPIO31) | EALLOW | Go |
| Ah | GPADIR | GPIO A Direction (GPIO0 to GPIO31) | EALLOW | Go |
| Ch | GPAPUD | GPIO A Pull-Up Disable (GPIO0 to GPIO31) | EALLOW | Go |
| 10h | GPAINV | GPIO A Input Inversion (GPIO0 to GPIO31) | EALLOW | Go |
| 12h | GPAODR | GPIO A Open Drain Output Mode (GPIO0 to GPIO31) | EALLOW | Go |
| 14h | GPAAMSEL | GPIO A Analog Mode Select (GPIO0 to GPIO31) | EALLOW | Go |
| 20h | GPAGMUX1 | GPIO A Peripheral Group Mux (GPIO0 to GPIO15) | EALLOW | Go |
| 22h | GPAGMUX2 | GPIO A Peripheral Group Mux (GPIO16 to GPIO31) | EALLOW | Go |
| 28h | GPACSEL1 | GPIO A Master Core Select (GPIO0 to GPIO7) | EALLOW | Go |
| 2Ah | GPACSEL2 | GPIO A Master Core Select (GPIO8 to GPIO15) | EALLOW | Go |
| 2Ch | GPACSEL3 | GPIO A Master Core Select (GPIO16 to GPIO23) | EALLOW | Go |
| 2Eh | GPACSEL4 | GPIO A Master Core Select (GPIO24 to GPIO31) | EALLOW | Go |
| 3Ch | GPALOCK | GPIO A Lock Register (GPIO0 to GPIO31) | EALLOW | Go |
| 3Eh | GPACR | GPIO A Lock Commit Register (GPIO0 to GPIO31) | EALLOW | Go |
| 40h | GPBCTRL | GPIO B Qualification Sampling Period (GPIO32 to GPIO63) | EALLOW | Go |
| 42h | GPBQSEL1 | GPIO B Qualification Type (GPIO32 to GPIO47) | EALLOW | Go |
| 44h | GPBQSEL2 | GPIO B Qualification Type (GPIO48 to GPIO63) | EALLOW | Go |
| 46h | GPBMUX1 | GPIO B Peripheral Mux (GPIO32 to GPIO47) | EALLOW | Go |
| 48h | GPBMUX2 | GPIO B Peripheral Mux (GPIO48 to GPIO63) | EALLOW | Go |
| 4Ah | GPBDIR | GPIO B Direction (GPIO32 to GPIO63) | EALLOW | Go |
| 4Ch | GPBPUD | GPIO B Pull-Up Disable (GPIO32 to GPIO63) | EALLOW | Go |
| 50h | GPBINV | GPIO B Input Inversion (GPIO32 to GPIO63) | EALLOW | Go |
| 52h | GPBODR | GPIO B Open Drain Output Mode (GPIO32 to GPIO63) | EALLOW | Go |
| 60h | GPBGMUX1 | GPIO B Peripheral Group Mux (GPIO32 to GPIO47) | EALLOW | Go |
| 62h | GPBGMUX2 | GPIO B Peripheral Group Mux (GPIO48 to GPIO63) | EALLOW | Go |
| 68h | GPBCSEL1 | GPIO B Master Core Select (GPIO32 to GPIO39) | EALLOW | Go |
| 6Ah | GPBCSEL2 | GPIO B Master Core Select (GPIO40 to GPIO47) | EALLOW | Go |
| 6Ch | GPBCSEL3 | GPIO B Master Core Select (GPIO48 to GPIO55) | EALLOW | Go |
| 6Eh | GPBCSEL4 | GPIO B Master Core Select (GPIO56 to GPIO63) | EALLOW | Go |

**Table 8-8. GPIO_CTRL_REGS Registers (continued)**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 7Ch | GPBLOCK | GPIO B Lock Register (GPIO32 to GPIO63) | EALLOW | Go |
| 7Eh | GPBCR | GPIO B Lock Commit Register (GPIO32 to GPIO63) | EALLOW | Go |
| 1C0h | GPHCTRL | GPIO H Qualification Sampling Period (GPIO224 to GPIO255) | EALLOW | Go |
| 1C2h | GPHQSEL1 | GPIO H Qualification Type (GPIO224 to GPIO239) | EALLOW | Go |
| 1C4h | GPHQSEL2 | GPIO H Qualification Type (GPIO240 to GPIO255) | EALLOW | Go |
| 1D0h | GPHINV | GPIO H Input Inversion (GPIO224 to GPIO255) | EALLOW | Go |
| 1D4h | GPHAMSEL | GPIO H Analog Mode Select (GPIO224 to GPIO255) | EALLOW | Go |
| 1FCh | GPHLOCK | GPIO H Lock Register (GPIO224 to GPIO255) | EALLOW | Go |
| 1FEh | GPHCR | GPIO H Lock Commit Register (GPIO224 to GPIO255) | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 8-9 shows the codes that are used for access types in this section.

**Table 8-9. GPIO_CTRL_REGS Access Type Codes**

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

#### 8.7.1.1.1 GPACTRL Register (Offset = 0h) [reset = 0h]

GPACTRL is shown in Figure 8-4 and described in Table 8-10.

Return to Summary Table.

GPIO A Qualification Sampling Period (GPIO0 to GPIO31)

Each field in this register selects the qualification sampling period in SYSCLK cycles for eight GPIOs. The period is equal to 2 times the register field value.

0x00: Period = 0 SYSCLK cycles
0x01: Period = 2 SYSCLK cycles
0x02: Period = 4 SYSCLK cycles
...
0xFF: Period = 510 SYSCLK cycles

**Figure 8-4. GPACTRL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QUALPRD3 | | | | | | | | QUALPRD2 | | | | | | | | QUALPRD1 | | | | | | | | QUALPRD0 | | | | | | | |
| R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | |

**Table 8-10. GPACTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | QUALPRD3 | R/W | 0h | Qualification sampling period for GPIO24 to GPIO31<br>Reset type: SYSRSn |
| 23-16 | QUALPRD2 | R/W | 0h | Qualification sampling period for GPIO16 to GPIO23<br>Reset type: SYSRSn |
| 15-8 | QUALPRD1 | R/W | 0h | Qualification sampling period for GPIO8 to GPIO15<br>Reset type: SYSRSn |
| 7-0 | QUALPRD0 | R/W | 0h | Qualification sampling period for GPIO0 to GPIO7<br>Reset type: SYSRSn |

#### 8.7.1.1.2 GPAQSEL1 Register (Offset = 2h) [reset = 0h]

GPAQSEL1 is shown in Figure 8-5 and described in Table 8-11.

Return to Summary Table.

GPIO A Qualification Type (GPIO0 to GPIO15)

Each field in this register selects the input qualification type for one IO pin. The available types are:

0: Synchronous
1: 3-sample qualification
2: 6-sample qualification
3: Asynchronous

**Figure 8-5. GPAQSEL1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO15 | | GPIO14 | | GPIO13 | | GPIO12 | | GPIO11 | | GPIO10 | | GPIO9 | | GPIO8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO7 | | GPIO6 | | GPIO5 | | GPIO4 | | GPIO3 | | GPIO2 | | GPIO1 | | GPIO0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-11. GPAQSEL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO15 | R/W | 0h | Input qualification type for GPIO15<br>Reset type: SYSRSn |
| 29-28 | GPIO14 | R/W | 0h | Input qualification type for GPIO14<br>Reset type: SYSRSn |
| 27-26 | GPIO13 | R/W | 0h | Input qualification type for GPIO13<br>Reset type: SYSRSn |
| 25-24 | GPIO12 | R/W | 0h | Input qualification type for GPIO12<br>Reset type: SYSRSn |
| 23-22 | GPIO11 | R/W | 0h | Input qualification type for GPIO11<br>Reset type: SYSRSn |
| 21-20 | GPIO10 | R/W | 0h | Input qualification type for GPIO10<br>Reset type: SYSRSn |
| 19-18 | GPIO9 | R/W | 0h | Input qualification type for GPIO9<br>Reset type: SYSRSn |
| 17-16 | GPIO8 | R/W | 0h | Input qualification type for GPIO8<br>Reset type: SYSRSn |
| 15-14 | GPIO7 | R/W | 0h | Input qualification type for GPIO7<br>Reset type: SYSRSn |
| 13-12 | GPIO6 | R/W | 0h | Input qualification type for GPIO6<br>Reset type: SYSRSn |
| 11-10 | GPIO5 | R/W | 0h | Input qualification type for GPIO5<br>Reset type: SYSRSn |
| 9-8 | GPIO4 | R/W | 0h | Input qualification type for GPIO4<br>Reset type: SYSRSn |
| 7-6 | GPIO3 | R/W | 0h | Input qualification type for GPIO3<br>Reset type: SYSRSn |
| 5-4 | GPIO2 | R/W | 0h | Input qualification type for GPIO2<br>Reset type: SYSRSn |

**Table 8-11. GPAQSEL1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | GPIO1 | R/W | 0h | Input qualification type for GPIO1<br>Reset type: SYSRSn |
| 1-0 | GPIO0 | R/W | 0h | Input qualification type for GPIO0<br>Reset type: SYSRSn |

#### 8.7.1.1.3 GPAQSEL2 Register (Offset = 4h) [reset = 0h]

GPAQSEL2 is shown in Figure 8-6 and described in Table 8-12.

Return to Summary Table.

GPIO A Qualification Type (GPIO16 to GPIO31)

Each field in this register determines the input qualification type for one IO pin. The available types are:

0: Synchronous
1: 3-sample qualification
2: 6-sample qualification
3: Asynchronous

**Figure 8-6. GPAQSEL2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO31 | | GPIO30 | | GPIO29 | | GPIO28 | | GPIO27 | | GPIO26 | | GPIO25 | | GPIO24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO23 | | GPIO22 | | GPIO21 | | GPIO20 | | GPIO19 | | GPIO18 | | GPIO17 | | GPIO16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-12. GPAQSEL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO31 | R/W | 0h | Input qualification type for GPIO31<br>Reset type: SYSRSn |
| 29-28 | GPIO30 | R/W | 0h | Input qualification type for GPIO30<br>Reset type: SYSRSn |
| 27-26 | GPIO29 | R/W | 0h | Input qualification type for GPIO29<br>Reset type: SYSRSn |
| 25-24 | GPIO28 | R/W | 0h | Input qualification type for GPIO28<br>Reset type: SYSRSn |
| 23-22 | GPIO27 | R/W | 0h | Input qualification type for GPIO27<br>Reset type: SYSRSn |
| 21-20 | GPIO26 | R/W | 0h | Input qualification type for GPIO26<br>Reset type: SYSRSn |
| 19-18 | GPIO25 | R/W | 0h | Input qualification type for GPIO25<br>Reset type: SYSRSn |
| 17-16 | GPIO24 | R/W | 0h | Input qualification type for GPIO24<br>Reset type: SYSRSn |
| 15-14 | GPIO23 | R/W | 0h | Input qualification type for GPIO23<br>Reset type: SYSRSn |
| 13-12 | GPIO22 | R/W | 0h | Input qualification type for GPIO22<br>Reset type: SYSRSn |
| 11-10 | GPIO21 | R/W | 0h | Input qualification type for GPIO21<br>Reset type: SYSRSn |
| 9-8 | GPIO20 | R/W | 0h | Input qualification type for GPIO20<br>Reset type: SYSRSn |
| 7-6 | GPIO19 | R/W | 0h | Input qualification type for GPIO19<br>Reset type: SYSRSn |
| 5-4 | GPIO18 | R/W | 0h | Input qualification type for GPIO18<br>Reset type: SYSRSn |

**Table 8-12. GPAQSEL2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | GPIO17 | R/W | 0h | Input qualification type for GPIO17<br>Reset type: SYSRSn |
| 1-0 | GPIO16 | R/W | 0h | Input qualification type for GPIO16<br>Reset type: SYSRSn |

#### 8.7.1.1.4 GPAMUX1 Register (Offset = 6h) [reset = 0h]

GPAMUX1 is shown in Figure 8-7 and described in Table 8-13.

Return to Summary Table.

GPIO A Peripheral Mux (GPIO0 to GPIO15)

Each field in this register determines part of the GPIO mux configuration for one IO pin. A value of 0x0, 0x4, 0x8, or 0xC configures the pin as a general-purpose IO. All other values select a peripheral to control the pin. See the device datasheet for a table of peripheral mux options. Pins must be set to GPIO mode using this register before changing the corresponding field in the GPAGMUX1 register.

**Figure 8-7. GPAMUX1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO15 | | GPIO14 | | GPIO13 | | GPIO12 | | GPIO11 | | GPIO10 | | GPIO9 | | GPIO8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO7 | | GPIO6 | | GPIO5 | | GPIO4 | | GPIO3 | | GPIO2 | | GPIO1 | | GPIO0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-13. GPAMUX1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO15 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO15<br>Reset type: SYSRSn |
| 29-28 | GPIO14 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO14<br>Reset type: SYSRSn |
| 27-26 | GPIO13 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO13<br>Reset type: SYSRSn |
| 25-24 | GPIO12 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO12<br>Reset type: SYSRSn |
| 23-22 | GPIO11 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO11<br>Reset type: SYSRSn |
| 21-20 | GPIO10 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO10<br>Reset type: SYSRSn |
| 19-18 | GPIO9 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO9<br>Reset type: SYSRSn |
| 17-16 | GPIO8 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO8<br>Reset type: SYSRSn |
| 15-14 | GPIO7 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO7<br>Reset type: SYSRSn |
| 13-12 | GPIO6 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO6<br>Reset type: SYSRSn |
| 11-10 | GPIO5 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO5<br>Reset type: SYSRSn |
| 9-8 | GPIO4 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO4<br>Reset type: SYSRSn |
| 7-6 | GPIO3 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO3<br>Reset type: SYSRSn |
| 5-4 | GPIO2 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO2<br>Reset type: SYSRSn |
| 3-2 | GPIO1 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO1<br>Reset type: SYSRSn |

**Table 8-13. GPAMUX1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | GPIO0 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO0<br>Reset type: SYSRSn |

#### 8.7.1.1.5 GPAMUX2 Register (Offset = 8h) [reset = 0h]

GPAMUX2 is shown in Figure 8-8 and described in Table 8-14.

Return to Summary Table.

GPIO A Peripheral Mux (GPIO16 to GPIO31)

Each field in this register determines part of the GPIO mux configuration for one IO pin. A value of 0x0, 0x4, 0x8, or 0xC configures the pin as a general-purpose IO. All other values select a peripheral to control the pin. See the device datasheet for a table of peripheral mux options. Pins must be set to GPIO mode using this register before changing the corresponding field in the GPAGMUX2 register.

**Figure 8-8. GPAMUX2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO31 | | GPIO30 | | GPIO29 | | GPIO28 | | GPIO27 | | GPIO26 | | GPIO25 | | GPIO24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO23 | | GPIO22 | | GPIO21 | | GPIO20 | | GPIO19 | | GPIO18 | | GPIO17 | | GPIO16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-14. GPAMUX2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO31 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO31<br>Reset type: SYSRSn |
| 29-28 | GPIO30 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO30<br>Reset type: SYSRSn |
| 27-26 | GPIO29 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO29<br>Reset type: SYSRSn |
| 25-24 | GPIO28 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO28<br>Reset type: SYSRSn |
| 23-22 | GPIO27 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO27<br>Reset type: SYSRSn |
| 21-20 | GPIO26 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO26<br>Reset type: SYSRSn |
| 19-18 | GPIO25 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO25<br>Reset type: SYSRSn |
| 17-16 | GPIO24 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO24<br>Reset type: SYSRSn |
| 15-14 | GPIO23 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO23<br>Reset type: SYSRSn |
| 13-12 | GPIO22 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO22<br>Reset type: SYSRSn |
| 11-10 | GPIO21 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO21<br>Reset type: SYSRSn |
| 9-8 | GPIO20 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO20<br>Reset type: SYSRSn |
| 7-6 | GPIO19 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO19<br>Reset type: SYSRSn |
| 5-4 | GPIO18 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO18<br>Reset type: SYSRSn |
| 3-2 | GPIO17 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO17<br>Reset type: SYSRSn |

**Table 8-14. GPAMUX2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | GPIO16 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO16<br>Reset type: SYSRSn |

#### 8.7.1.1.6 GPADIR Register (Offset = Ah) [reset = 0h]

GPADIR is shown in Figure 8-9 and described in Table 8-15.

Return to Summary Table.

GPIO A Direction (GPIO0 to GPIO31)

Each field in this register selects the direction of one IO pin in GPIO mode. If the pin is not configured as a general-purpose IO, this register has no effect.

0: The pin is an input
1: The pin is an output

**Figure 8-9. GPADIR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO31 | GPIO30 | GPIO29 | GPIO28 | GPIO27 | GPIO26 | GPIO25 | GPIO24 |
| 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 8-15. GPADIR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO31 | R/W | 0h | Data direction for GPIO31<br>Reset type: SYSRSn |
| 30 | GPIO30 | R/W | 0h | Data direction for GPIO30<br>Reset type: SYSRSn |
| 29 | GPIO29 | R/W | 0h | Data direction for GPIO29<br>Reset type: SYSRSn |
| 28 | GPIO28 | R/W | 0h | Data direction for GPIO28<br>Reset type: SYSRSn |
| 27 | GPIO27 | R/W | 0h | Data direction for GPIO27<br>Reset type: SYSRSn |
| 26 | GPIO26 | R/W | 0h | Data direction for GPIO26<br>Reset type: SYSRSn |
| 25 | GPIO25 | R/W | 0h | Data direction for GPIO25<br>Reset type: SYSRSn |
| 24 | GPIO24 | R/W | 0h | Data direction for GPIO24<br>Reset type: SYSRSn |
| 23 | GPIO23 | R/W | 0h | Data direction for GPIO23<br>Reset type: SYSRSn |
| 22 | GPIO22 | R/W | 0h | Data direction for GPIO22<br>Reset type: SYSRSn |
| 21 | GPIO21 | R/W | 0h | Data direction for GPIO21<br>Reset type: SYSRSn |
| 20 | GPIO20 | R/W | 0h | Data direction for GPIO20<br>Reset type: SYSRSn |

**Table 8-15. GPADIR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO19 | R/W | 0h | Data direction for GPIO19<br>Reset type: SYSRSn |
| 18 | GPIO18 | R/W | 0h | Data direction for GPIO18<br>Reset type: SYSRSn |
| 17 | GPIO17 | R/W | 0h | Data direction for GPIO17<br>Reset type: SYSRSn |
| 16 | GPIO16 | R/W | 0h | Data direction for GPIO16<br>Reset type: SYSRSn |
| 15 | GPIO15 | R/W | 0h | Data direction for GPIO15<br>Reset type: SYSRSn |
| 14 | GPIO14 | R/W | 0h | Data direction for GPIO14<br>Reset type: SYSRSn |
| 13 | GPIO13 | R/W | 0h | Data direction for GPIO13<br>Reset type: SYSRSn |
| 12 | GPIO12 | R/W | 0h | Data direction for GPIO12<br>Reset type: SYSRSn |
| 11 | GPIO11 | R/W | 0h | Data direction for GPIO11<br>Reset type: SYSRSn |
| 10 | GPIO10 | R/W | 0h | Data direction for GPIO10<br>Reset type: SYSRSn |
| 9 | GPIO9 | R/W | 0h | Data direction for GPIO9<br>Reset type: SYSRSn |
| 8 | GPIO8 | R/W | 0h | Data direction for GPIO8<br>Reset type: SYSRSn |
| 7 | GPIO7 | R/W | 0h | Data direction for GPIO7<br>Reset type: SYSRSn |
| 6 | GPIO6 | R/W | 0h | Data direction for GPIO6<br>Reset type: SYSRSn |
| 5 | GPIO5 | R/W | 0h | Data direction for GPIO5<br>Reset type: SYSRSn |
| 4 | GPIO4 | R/W | 0h | Data direction for GPIO4<br>Reset type: SYSRSn |
| 3 | GPIO3 | R/W | 0h | Data direction for GPIO3<br>Reset type: SYSRSn |
| 2 | GPIO2 | R/W | 0h | Data direction for GPIO2<br>Reset type: SYSRSn |
| 1 | GPIO1 | R/W | 0h | Data direction for GPIO1<br>Reset type: SYSRSn |
| 0 | GPIO0 | R/W | 0h | Data direction for GPIO0<br>Reset type: SYSRSn |

#### 8.7.1.1.7 GPAPUD Register (Offset = Ch) [reset = FFFFFFFFh]

GPAPUD is shown in Figure 8-10 and described in Table 8-16.

Return to Summary Table.

GPIO A Pull-Up Disable (GPIO0 to GPIO31)

Each field in this register selects the state of the internal pull-up resistor for a single IO pin.

0: Pull-up enabled
1: Pull-up disabled

**Figure 8-10. GPAPUD Register**

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

**Table 8-16. GPAPUD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO31 | R/W | 1h | Pull-up disable for GPIO31<br>Reset type: SYSRSn |
| 30 | GPIO30 | R/W | 1h | Pull-up disable for GPIO30<br>Reset type: SYSRSn |
| 29 | GPIO29 | R/W | 1h | Pull-up disable for GPIO29<br>Reset type: SYSRSn |
| 28 | GPIO28 | R/W | 1h | Pull-up disable for GPIO28<br>Reset type: SYSRSn |
| 27 | GPIO27 | R/W | 1h | Pull-up disable for GPIO27<br>Reset type: SYSRSn |
| 26 | GPIO26 | R/W | 1h | Pull-up disable for GPIO26<br>Reset type: SYSRSn |
| 25 | GPIO25 | R/W | 1h | Pull-up disable for GPIO25<br>Reset type: SYSRSn |
| 24 | GPIO24 | R/W | 1h | Pull-up disable for GPIO24<br>Reset type: SYSRSn |
| 23 | GPIO23 | R/W | 1h | Pull-up disable for GPIO23<br>Reset type: SYSRSn |
| 22 | GPIO22 | R/W | 1h | Pull-up disable for GPIO22<br>Reset type: SYSRSn |
| 21 | GPIO21 | R/W | 1h | Pull-up disable for GPIO21<br>Reset type: SYSRSn |
| 20 | GPIO20 | R/W | 1h | Pull-up disable for GPIO20<br>Reset type: SYSRSn |

**Table 8-16. GPAPUD Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO19 | R/W | 1h | Pull-up disable for GPIO19<br>Reset type: SYSRSn |
| 18 | GPIO18 | R/W | 1h | Pull-up disable for GPIO18<br>Reset type: SYSRSn |
| 17 | GPIO17 | R/W | 1h | Pull-up disable for GPIO17<br>Reset type: SYSRSn |
| 16 | GPIO16 | R/W | 1h | Pull-up disable for GPIO16<br>Reset type: SYSRSn |
| 15 | GPIO15 | R/W | 1h | Pull-up disable for GPIO15<br>Reset type: SYSRSn |
| 14 | GPIO14 | R/W | 1h | Pull-up disable for GPIO14<br>Reset type: SYSRSn |
| 13 | GPIO13 | R/W | 1h | Pull-up disable for GPIO13<br>Reset type: SYSRSn |
| 12 | GPIO12 | R/W | 1h | Pull-up disable for GPIO12<br>Reset type: SYSRSn |
| 11 | GPIO11 | R/W | 1h | Pull-up disable for GPIO11<br>Reset type: SYSRSn |
| 10 | GPIO10 | R/W | 1h | Pull-up disable for GPIO10<br>Reset type: SYSRSn |
| 9 | GPIO9 | R/W | 1h | Pull-up disable for GPIO9<br>Reset type: SYSRSn |
| 8 | GPIO8 | R/W | 1h | Pull-up disable for GPIO8<br>Reset type: SYSRSn |
| 7 | GPIO7 | R/W | 1h | Pull-up disable for GPIO7<br>Reset type: SYSRSn |
| 6 | GPIO6 | R/W | 1h | Pull-up disable for GPIO6<br>Reset type: SYSRSn |
| 5 | GPIO5 | R/W | 1h | Pull-up disable for GPIO5<br>Reset type: SYSRSn |
| 4 | GPIO4 | R/W | 1h | Pull-up disable for GPIO4<br>Reset type: SYSRSn |
| 3 | GPIO3 | R/W | 1h | Pull-up disable for GPIO3<br>Reset type: SYSRSn |
| 2 | GPIO2 | R/W | 1h | Pull-up disable for GPIO2<br>Reset type: SYSRSn |
| 1 | GPIO1 | R/W | 1h | Pull-up disable for GPIO1<br>Reset type: SYSRSn |
| 0 | GPIO0 | R/W | 1h | Pull-up disable for GPIO0<br>Reset type: SYSRSn |

#### 8.7.1.1.8 GPAINV Register (Offset = 10h) [reset = 0h]

GPAINV is shown in Figure 8-11 and described in Table 8-17.

Return to Summary Table.

GPIO A Input Inversion (GPIO0 to GPIO31)

Each field in this register selects whether the input value of one IO pin passes through an inverter.

0: The input is not inverted
1: The input is inverted

**Figure 8-11. GPAINV Register**

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

**Table 8-17. GPAINV Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO31 | R/W | 0h | Input inversion for GPIO31<br>Reset type: SYSRSn |
| 30 | GPIO30 | R/W | 0h | Input inversion for GPIO30<br>Reset type: SYSRSn |
| 29 | GPIO29 | R/W | 0h | Input inversion for GPIO29<br>Reset type: SYSRSn |
| 28 | GPIO28 | R/W | 0h | Input inversion for GPIO28<br>Reset type: SYSRSn |
| 27 | GPIO27 | R/W | 0h | Input inversion for GPIO27<br>Reset type: SYSRSn |
| 26 | GPIO26 | R/W | 0h | Input inversion for GPIO26<br>Reset type: SYSRSn |
| 25 | GPIO25 | R/W | 0h | Input inversion for GPIO25<br>Reset type: SYSRSn |
| 24 | GPIO24 | R/W | 0h | Input inversion for GPIO24<br>Reset type: SYSRSn |
| 23 | GPIO23 | R/W | 0h | Input inversion for GPIO23<br>Reset type: SYSRSn |
| 22 | GPIO22 | R/W | 0h | Input inversion for GPIO22<br>Reset type: SYSRSn |
| 21 | GPIO21 | R/W | 0h | Input inversion for GPIO21<br>Reset type: SYSRSn |
| 20 | GPIO20 | R/W | 0h | Input inversion for GPIO20<br>Reset type: SYSRSn |

**Table 8-17. GPAINV Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO19 | R/W | 0h | Input inversion for GPIO19<br>Reset type: SYSRSn |
| 18 | GPIO18 | R/W | 0h | Input inversion for GPIO18<br>Reset type: SYSRSn |
| 17 | GPIO17 | R/W | 0h | Input inversion for GPIO17<br>Reset type: SYSRSn |
| 16 | GPIO16 | R/W | 0h | Input inversion for GPIO16<br>Reset type: SYSRSn |
| 15 | GPIO15 | R/W | 0h | Input inversion for GPIO15<br>Reset type: SYSRSn |
| 14 | GPIO14 | R/W | 0h | Input inversion for GPIO14<br>Reset type: SYSRSn |
| 13 | GPIO13 | R/W | 0h | Input inversion for GPIO13<br>Reset type: SYSRSn |
| 12 | GPIO12 | R/W | 0h | Input inversion for GPIO12<br>Reset type: SYSRSn |
| 11 | GPIO11 | R/W | 0h | Input inversion for GPIO11<br>Reset type: SYSRSn |
| 10 | GPIO10 | R/W | 0h | Input inversion for GPIO10<br>Reset type: SYSRSn |
| 9 | GPIO9 | R/W | 0h | Input inversion for GPIO9<br>Reset type: SYSRSn |
| 8 | GPIO8 | R/W | 0h | Input inversion for GPIO8<br>Reset type: SYSRSn |
| 7 | GPIO7 | R/W | 0h | Input inversion for GPIO7<br>Reset type: SYSRSn |
| 6 | GPIO6 | R/W | 0h | Input inversion for GPIO6<br>Reset type: SYSRSn |
| 5 | GPIO5 | R/W | 0h | Input inversion for GPIO5<br>Reset type: SYSRSn |
| 4 | GPIO4 | R/W | 0h | Input inversion for GPIO4<br>Reset type: SYSRSn |
| 3 | GPIO3 | R/W | 0h | Input inversion for GPIO3<br>Reset type: SYSRSn |
| 2 | GPIO2 | R/W | 0h | Input inversion for GPIO2<br>Reset type: SYSRSn |
| 1 | GPIO1 | R/W | 0h | Input inversion for GPIO1<br>Reset type: SYSRSn |
| 0 | GPIO0 | R/W | 0h | Input inversion for GPIO0<br>Reset type: SYSRSn |

#### 8.7.1.1.9 GPAODR Register (Offset = 12h) [reset = 0h]

GPAODR is shown in Figure 8-12 and described in Table 8-18.

Return to Summary Table.

GPIO A Open Drain Output Mode (GPIO0 to GPIO31)

Each field in this register selects between push-pull mode and open-drain mode for one general-purpose output pin. In both modes, writing a 0 to the output data latch drives the pin low. In push-pull mode, writing a 1 to the output data latch drives the pin high. In open-drain mode, it tri-states the output buffer.

0: Push-pull output
1: Open-drain output

**Figure 8-12. GPAODR Register**

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

**Table 8-18. GPAODR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO31 | R/W | 0h | Open-drain output mode for GPIO31<br>Reset type: SYSRSn |
| 30 | GPIO30 | R/W | 0h | Open-drain output mode for GPIO30<br>Reset type: SYSRSn |
| 29 | GPIO29 | R/W | 0h | Open-drain output mode for GPIO29<br>Reset type: SYSRSn |
| 28 | GPIO28 | R/W | 0h | Open-drain output mode for GPIO28<br>Reset type: SYSRSn |
| 27 | GPIO27 | R/W | 0h | Open-drain output mode for GPIO27<br>Reset type: SYSRSn |
| 26 | GPIO26 | R/W | 0h | Open-drain output mode for GPIO26<br>Reset type: SYSRSn |
| 25 | GPIO25 | R/W | 0h | Open-drain output mode for GPIO25<br>Reset type: SYSRSn |
| 24 | GPIO24 | R/W | 0h | Open-drain output mode for GPIO24<br>Reset type: SYSRSn |
| 23 | GPIO23 | R/W | 0h | Open-drain output mode for GPIO23<br>Reset type: SYSRSn |
| 22 | GPIO22 | R/W | 0h | Open-drain output mode for GPIO22<br>Reset type: SYSRSn |
| 21 | GPIO21 | R/W | 0h | Open-drain output mode for GPIO21<br>Reset type: SYSRSn |

**Table 8-18. GPAODR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO20 | R/W | 0h | Open-drain output mode for GPIO20<br>Reset type: SYSRSn |
| 19 | GPIO19 | R/W | 0h | Open-drain output mode for GPIO19<br>Reset type: SYSRSn |
| 18 | GPIO18 | R/W | 0h | Open-drain output mode for GPIO18<br>Reset type: SYSRSn |
| 17 | GPIO17 | R/W | 0h | Open-drain output mode for GPIO17<br>Reset type: SYSRSn |
| 16 | GPIO16 | R/W | 0h | Open-drain output mode for GPIO16<br>Reset type: SYSRSn |
| 15 | GPIO15 | R/W | 0h | Open-drain output mode for GPIO15<br>Reset type: SYSRSn |
| 14 | GPIO14 | R/W | 0h | Open-drain output mode for GPIO14<br>Reset type: SYSRSn |
| 13 | GPIO13 | R/W | 0h | Open-drain output mode for GPIO13<br>Reset type: SYSRSn |
| 12 | GPIO12 | R/W | 0h | Open-drain output mode for GPIO12<br>Reset type: SYSRSn |
| 11 | GPIO11 | R/W | 0h | Open-drain output mode for GPIO11<br>Reset type: SYSRSn |
| 10 | GPIO10 | R/W | 0h | Open-drain output mode for GPIO10<br>Reset type: SYSRSn |
| 9 | GPIO9 | R/W | 0h | Open-drain output mode for GPIO9<br>Reset type: SYSRSn |
| 8 | GPIO8 | R/W | 0h | Open-drain output mode for GPIO8<br>Reset type: SYSRSn |
| 7 | GPIO7 | R/W | 0h | Open-drain output mode for GPIO7<br>Reset type: SYSRSn |
| 6 | GPIO6 | R/W | 0h | Open-drain output mode for GPIO6<br>Reset type: SYSRSn |
| 5 | GPIO5 | R/W | 0h | Open-drain output mode for GPIO5<br>Reset type: SYSRSn |
| 4 | GPIO4 | R/W | 0h | Open-drain output mode for GPIO4<br>Reset type: SYSRSn |
| 3 | GPIO3 | R/W | 0h | Open-drain output mode for GPIO3<br>Reset type: SYSRSn |
| 2 | GPIO2 | R/W | 0h | Open-drain output mode for GPIO2<br>Reset type: SYSRSn |
| 1 | GPIO1 | R/W | 0h | Open-drain output mode for GPIO1<br>Reset type: SYSRSn |
| 0 | GPIO0 | R/W | 0h | Open-drain output mode for GPIO0<br>Reset type: SYSRSn |

#### 8.7.1.1.10 GPAAMSEL Register (Offset = 14h) [reset = 0h]

GPAAMSEL is shown in Figure 8-13 and described in Table 8-19.

Return to Summary Table.

GPIO A Analog Mode Select (GPIO0 to GPIO31)

Each field in this register selects between analog and digital functionality for one IO pin.

0: Digital mode
1: Analog mode

**Figure 8-13. GPAAMSEL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO23 | GPIO22 | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| R/W-0h | R/W-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 8-19. GPAAMSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RESERVED | R | 0h | Reserved |
| 30 | RESERVED | R | 0h | Reserved |
| 29 | RESERVED | R | 0h | Reserved |
| 28 | RESERVED | R | 0h | Reserved |
| 27 | RESERVED | R | 0h | Reserved |
| 26 | RESERVED | R | 0h | Reserved |
| 25 | RESERVED | R | 0h | Reserved |
| 24 | RESERVED | R | 0h | Reserved |
| 23 | GPIO23 | R/W | 0h | Analog mode select for GPIO23<br>Reset type: SYSRSn |
| 22 | GPIO22 | R/W | 0h | Analog mode select for GPIO22<br>Reset type: SYSRSn |
| 21 | RESERVED | R | 0h | Reserved |
| 20 | RESERVED | R | 0h | Reserved |
| 19 | RESERVED | R | 0h | Reserved |
| 18 | RESERVED | R | 0h | Reserved |
| 17 | RESERVED | R | 0h | Reserved |
| 16 | RESERVED | R | 0h | Reserved |
| 15 | RESERVED | R | 0h | Reserved |
| 14 | RESERVED | R | 0h | Reserved |
| 13 | RESERVED | R | 0h | Reserved |
| 12 | RESERVED | R | 0h | Reserved |
| 11 | RESERVED | R | 0h | Reserved |
| 10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |

**Table 8-19. GPAAMSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 8 | RESERVED | R | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1 | RESERVED | R | 0h | Reserved |
| 0 | RESERVED | R | 0h | Reserved |

#### 8.7.1.1.11 GPAGMUX1 Register (Offset = 20h) [reset = 0h]

GPAGMUX1 is shown in Figure 8-14 and described in Table 8-20.

Return to Summary Table.

GPIO A Peripheral Group Mux (GPIO0 to GPIO15)

Each field in this register determines part of the GPIO mux configuration for one IO pin. See the device datasheet for a table of peripheral mux options. Pins must be set to GPIO mode using the GPAMUX1 register before changing their configuration in this register.

**Figure 8-14. GPAGMUX1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO15 | | GPIO14 | | GPIO13 | | GPIO12 | | GPIO11 | | GPIO10 | | GPIO9 | | GPIO8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO7 | | GPIO6 | | GPIO5 | | GPIO4 | | GPIO3 | | GPIO2 | | GPIO1 | | GPIO0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-20. GPAGMUX1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO15 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO15<br>Reset type: SYSRSn |
| 29-28 | GPIO14 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO14<br>Reset type: SYSRSn |
| 27-26 | GPIO13 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO13<br>Reset type: SYSRSn |
| 25-24 | GPIO12 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO12<br>Reset type: SYSRSn |
| 23-22 | GPIO11 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO11<br>Reset type: SYSRSn |
| 21-20 | GPIO10 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO10<br>Reset type: SYSRSn |
| 19-18 | GPIO9 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO9<br>Reset type: SYSRSn |
| 17-16 | GPIO8 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO8<br>Reset type: SYSRSn |
| 15-14 | GPIO7 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO7<br>Reset type: SYSRSn |
| 13-12 | GPIO6 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO6<br>Reset type: SYSRSn |
| 11-10 | GPIO5 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO5<br>Reset type: SYSRSn |
| 9-8 | GPIO4 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO4<br>Reset type: SYSRSn |
| 7-6 | GPIO3 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO3<br>Reset type: SYSRSn |
| 5-4 | GPIO2 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO2<br>Reset type: SYSRSn |
| 3-2 | GPIO1 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO1<br>Reset type: SYSRSn |

**Table 8-20. GPAGMUX1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | GPIO0 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO0<br>Reset type: SYSRSn |

#### 8.7.1.1.12 GPAGMUX2 Register (Offset = 22h) [reset = 0h]

GPAGMUX2 is shown in Figure 8-15 and described in Table 8-21.

Return to Summary Table.

GPIO A Peripheral Group Mux (GPIO16 to GPIO31)

Each field in this register determines part of the GPIO mux configuration for a single IO pin. See the device datasheet for a table of peripheral mux options. Pins must be set to GPIO mode using the GPAMUX2 register before changing their configuration in this register.

**Figure 8-15. GPAGMUX2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO31 | | GPIO30 | | GPIO29 | | GPIO28 | | GPIO27 | | GPIO26 | | GPIO25 | | GPIO24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO23 | | GPIO22 | | GPIO21 | | GPIO20 | | GPIO19 | | GPIO18 | | GPIO17 | | GPIO16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-21. GPAGMUX2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO31 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO31<br>Reset type: SYSRSn |
| 29-28 | GPIO30 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO30<br>Reset type: SYSRSn |
| 27-26 | GPIO29 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO29<br>Reset type: SYSRSn |
| 25-24 | GPIO28 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO28<br>Reset type: SYSRSn |
| 23-22 | GPIO27 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO27<br>Reset type: SYSRSn |
| 21-20 | GPIO26 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO26<br>Reset type: SYSRSn |
| 19-18 | GPIO25 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO25<br>Reset type: SYSRSn |
| 17-16 | GPIO24 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO24<br>Reset type: SYSRSn |
| 15-14 | GPIO23 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO23<br>Reset type: SYSRSn |
| 13-12 | GPIO22 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO22<br>Reset type: SYSRSn |
| 11-10 | GPIO21 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO21<br>Reset type: SYSRSn |
| 9-8 | GPIO20 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO20<br>Reset type: SYSRSn |
| 7-6 | GPIO19 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO19<br>Reset type: SYSRSn |
| 5-4 | GPIO18 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO18<br>Reset type: SYSRSn |
| 3-2 | GPIO17 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO17<br>Reset type: SYSRSn |

**Table 8-21. GPAGMUX2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | GPIO16 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO16<br>Reset type: SYSRSn |

#### 8.7.1.1.13 GPACSEL1 Register (Offset = 28h) [reset = 0h]

GPACSEL1 is shown in Figure 8-16 and described in Table 8-22.

Return to Summary Table.

GPIO A Master Core Select (GPIO0 to GPIO7)

Each field in this register selects the master for one IO pin. The master controls the pin in GPIO mode via its GPADAT, GPASET, GPACLEAR, and GPATOGGLE registers.

0x0: CPU is the master
0x1: CLA is the master
0x2 - 0xF: Reserved

**Figure 8-16. GPACSEL1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO7 | | | | GPIO6 | | | | GPIO5 | | | | GPIO4 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO3 | | | | GPIO2 | | | | GPIO1 | | | | GPIO0 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-22. GPACSEL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO7 | R/W | 0h | Master core select for GPIO7<br>Reset type: SYSRSn |
| 27-24 | GPIO6 | R/W | 0h | Master core select for GPIO6<br>Reset type: SYSRSn |
| 23-20 | GPIO5 | R/W | 0h | Master core select for GPIO5<br>Reset type: SYSRSn |
| 19-16 | GPIO4 | R/W | 0h | Master core select for GPIO4<br>Reset type: SYSRSn |
| 15-12 | GPIO3 | R/W | 0h | Master core select for GPIO3<br>Reset type: SYSRSn |
| 11-8 | GPIO2 | R/W | 0h | Master core select for GPIO2<br>Reset type: SYSRSn |
| 7-4 | GPIO1 | R/W | 0h | Master core select for GPIO1<br>Reset type: SYSRSn |
| 3-0 | GPIO0 | R/W | 0h | Master core select for GPIO0<br>Reset type: SYSRSn |

#### 8.7.1.1.14 GPACSEL2 Register (Offset = 2Ah) [reset = 0h]

GPACSEL2 is shown in Figure 8-17 and described in Table 8-23.

Return to Summary Table.

GPIO A Master Core Select (GPIO8 to GPIO15)

Each field in this register selects the master for one IO pin. The master controls the pin in GPIO mode via its GPADAT, GPASET, GPACLEAR, and GPATOGGLE registers.

0x0: CPU is the master
0x1: CLA is the master
0x2 - 0xF: Reserved

**Figure 8-17. GPACSEL2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO15 | | | | GPIO14 | | | | GPIO13 | | | | GPIO12 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO11 | | | | GPIO10 | | | | GPIO9 | | | | GPIO8 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-23. GPACSEL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO15 | R/W | 0h | Master core select for GPIO15<br>Reset type: SYSRSn |
| 27-24 | GPIO14 | R/W | 0h | Master core select for GPIO14<br>Reset type: SYSRSn |
| 23-20 | GPIO13 | R/W | 0h | Master core select for GPIO13<br>Reset type: SYSRSn |
| 19-16 | GPIO12 | R/W | 0h | Master core select for GPIO12<br>Reset type: SYSRSn |
| 15-12 | GPIO11 | R/W | 0h | Master core select for GPIO11<br>Reset type: SYSRSn |
| 11-8 | GPIO10 | R/W | 0h | Master core select for GPIO10<br>Reset type: SYSRSn |
| 7-4 | GPIO9 | R/W | 0h | Master core select for GPIO9<br>Reset type: SYSRSn |
| 3-0 | GPIO8 | R/W | 0h | Master core select for GPIO8<br>Reset type: SYSRSn |

#### 8.7.1.1.15 GPACSEL3 Register (Offset = 2Ch) [reset = 0h]

GPACSEL3 is shown in Figure 8-18 and described in Table 8-24.

Return to Summary Table.

GPIO A Master Core Select (GPIO16 to GPIO23)

Each field in this register selects the master for one IO pin. The master controls the pin in GPIO mode via its GPADAT, GPASET, GPACLEAR, and GPATOGGLE registers.

0x0: CPU is the master
0x1: CLA is the master
0x2 - 0xF: Reserved

**Figure 8-18. GPACSEL3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO23 | | | | GPIO22 | | | | GPIO21 | | | | GPIO20 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO19 | | | | GPIO18 | | | | GPIO17 | | | | GPIO16 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-24. GPACSEL3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO23 | R/W | 0h | Master core select for GPIO23<br>Reset type: SYSRSn |
| 27-24 | GPIO22 | R/W | 0h | Master core select for GPIO22<br>Reset type: SYSRSn |
| 23-20 | GPIO21 | R/W | 0h | Master core select for GPIO21<br>Reset type: SYSRSn |
| 19-16 | GPIO20 | R/W | 0h | Master core select for GPIO20<br>Reset type: SYSRSn |
| 15-12 | GPIO19 | R/W | 0h | Master core select for GPIO19<br>Reset type: SYSRSn |
| 11-8 | GPIO18 | R/W | 0h | Master core select for GPIO18<br>Reset type: SYSRSn |
| 7-4 | GPIO17 | R/W | 0h | Master core select for GPIO17<br>Reset type: SYSRSn |
| 3-0 | GPIO16 | R/W | 0h | Master core select for GPIO16<br>Reset type: SYSRSn |

#### 8.7.1.1.16 GPACSEL4 Register (Offset = 2Eh) [reset = 0h]

GPACSEL4 is shown in Figure 8-19 and described in Table 8-25.

Return to Summary Table.

GPIO A Master Core Select (GPIO24 to GPIO31)

Each field in this register selects the master for one IO pin. The master controls the pin in GPIO mode via its GPADAT, GPASET, GPACLEAR, and GPATOGGLE registers.

0x0: CPU is the master
0x1: CLA is the master
0x2 - 0xF: Reserved

**Figure 8-19. GPACSEL4 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO31 | | | | GPIO30 | | | | GPIO29 | | | | GPIO28 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO27 | | | | GPIO26 | | | | GPIO25 | | | | GPIO24 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-25. GPACSEL4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO31 | R/W | 0h | Master core select for GPIO31<br>Reset type: SYSRSn |
| 27-24 | GPIO30 | R/W | 0h | Master core select for GPIO30<br>Reset type: SYSRSn |
| 23-20 | GPIO29 | R/W | 0h | Master core select for GPIO29<br>Reset type: SYSRSn |
| 19-16 | GPIO28 | R/W | 0h | Master core select for GPIO28<br>Reset type: SYSRSn |
| 15-12 | GPIO27 | R/W | 0h | Master core select for GPIO27<br>Reset type: SYSRSn |
| 11-8 | GPIO26 | R/W | 0h | Master core select for GPIO26<br>Reset type: SYSRSn |
| 7-4 | GPIO25 | R/W | 0h | Master core select for GPIO25<br>Reset type: SYSRSn |
| 3-0 | GPIO24 | R/W | 0h | Master core select for GPIO24<br>Reset type: SYSRSn |

#### 8.7.1.1.17 GPALOCK Register (Offset = 3Ch) [reset = 0h]

GPALOCK is shown in Figure 8-20 and described in Table 8-26.

Return to Summary Table.

GPIO A Lock Register (GPIO0 to GPIO31)

Each field in this register locks one IO pin's configuration. This blocks writes to the corresponding bits in the GPAMUXn, GPAIDR, GPAINV, GPAODR, GPAGMUXn, and GPACSELn registers.

0: Pin configuration is unlocked
1: Pin configuration is locked

**Figure 8-20. GPALOCK Register**

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

**Table 8-26. GPALOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO31 | R/W | 0h | Configuration lock for GPIO31<br>Reset type: SYSRSn |
| 30 | GPIO30 | R/W | 0h | Configuration lock for GPIO30<br>Reset type: SYSRSn |
| 29 | GPIO29 | R/W | 0h | Configuration lock for GPIO29<br>Reset type: SYSRSn |
| 28 | GPIO28 | R/W | 0h | Configuration lock for GPIO28<br>Reset type: SYSRSn |
| 27 | GPIO27 | R/W | 0h | Configuration lock for GPIO27<br>Reset type: SYSRSn |
| 26 | GPIO26 | R/W | 0h | Configuration lock for GPIO26<br>Reset type: SYSRSn |
| 25 | GPIO25 | R/W | 0h | Configuration lock for GPIO25<br>Reset type: SYSRSn |
| 24 | GPIO24 | R/W | 0h | Configuration lock for GPIO24<br>Reset type: SYSRSn |
| 23 | GPIO23 | R/W | 0h | Configuration lock for GPIO23<br>Reset type: SYSRSn |
| 22 | GPIO22 | R/W | 0h | Configuration lock for GPIO22<br>Reset type: SYSRSn |
| 21 | GPIO21 | R/W | 0h | Configuration lock for GPIO21<br>Reset type: SYSRSn |
| 20 | GPIO20 | R/W | 0h | Configuration lock for GPIO20<br>Reset type: SYSRSn |

**Table 8-26. GPALOCK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO19 | R/W | 0h | Configuration lock for GPIO19<br>Reset type: SYSRSn |
| 18 | GPIO18 | R/W | 0h | Configuration lock for GPIO18<br>Reset type: SYSRSn |
| 17 | GPIO17 | R/W | 0h | Configuration lock for GPIO17<br>Reset type: SYSRSn |
| 16 | GPIO16 | R/W | 0h | Configuration lock for GPIO16<br>Reset type: SYSRSn |
| 15 | GPIO15 | R/W | 0h | Configuration lock for GPIO15<br>Reset type: SYSRSn |
| 14 | GPIO14 | R/W | 0h | Configuration lock for GPIO14<br>Reset type: SYSRSn |
| 13 | GPIO13 | R/W | 0h | Configuration lock for GPIO13<br>Reset type: SYSRSn |
| 12 | GPIO12 | R/W | 0h | Configuration lock for GPIO12<br>Reset type: SYSRSn |
| 11 | GPIO11 | R/W | 0h | Configuration lock for GPIO11<br>Reset type: SYSRSn |
| 10 | GPIO10 | R/W | 0h | Configuration lock for GPIO10<br>Reset type: SYSRSn |
| 9 | GPIO9 | R/W | 0h | Configuration lock for GPIO9<br>Reset type: SYSRSn |
| 8 | GPIO8 | R/W | 0h | Configuration lock for GPIO8<br>Reset type: SYSRSn |
| 7 | GPIO7 | R/W | 0h | Configuration lock for GPIO7<br>Reset type: SYSRSn |
| 6 | GPIO6 | R/W | 0h | Configuration lock for GPIO6<br>Reset type: SYSRSn |
| 5 | GPIO5 | R/W | 0h | Configuration lock for GPIO5<br>Reset type: SYSRSn |
| 4 | GPIO4 | R/W | 0h | Configuration lock for GPIO4<br>Reset type: SYSRSn |
| 3 | GPIO3 | R/W | 0h | Configuration lock for GPIO3<br>Reset type: SYSRSn |
| 2 | GPIO2 | R/W | 0h | Configuration lock for GPIO2<br>Reset type: SYSRSn |
| 1 | GPIO1 | R/W | 0h | Configuration lock for GPIO1<br>Reset type: SYSRSn |
| 0 | GPIO0 | R/W | 0h | Configuration lock for GPIO0<br>Reset type: SYSRSn |

#### 8.7.1.1.18 GPACR Register (Offset = 3Eh) [reset = 0h]

GPACR is shown in Figure 8-21 and described in Table 8-27.

Return to Summary Table.

GPIO A Lock Commit Register (GPIO0 to GPIO31)

Each field in this register blocks writes to one IO pin's GPALOCK bit. Once set, a lock commit can only be cleared by a reset.

0: Pin configuration lock is unlocked
1: Pin configuration lock is locked

**Figure 8-21. GPACR Register**

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

**Table 8-27. GPACR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO31 | R/WSOnce | 0h | Configuration lock commit for GPIO31<br>Reset type: SYSRSn |
| 30 | GPIO30 | R/WSOnce | 0h | Configuration lock commit for GPIO30<br>Reset type: SYSRSn |
| 29 | GPIO29 | R/WSOnce | 0h | Configuration lock commit for GPIO29<br>Reset type: SYSRSn |
| 28 | GPIO28 | R/WSOnce | 0h | Configuration lock commit for GPIO28<br>Reset type: SYSRSn |
| 27 | GPIO27 | R/WSOnce | 0h | Configuration lock commit for GPIO27<br>Reset type: SYSRSn |
| 26 | GPIO26 | R/WSOnce | 0h | Configuration lock commit for GPIO26<br>Reset type: SYSRSn |
| 25 | GPIO25 | R/WSOnce | 0h | Configuration lock commit for GPIO25<br>Reset type: SYSRSn |
| 24 | GPIO24 | R/WSOnce | 0h | Configuration lock commit for GPIO24<br>Reset type: SYSRSn |
| 23 | GPIO23 | R/WSOnce | 0h | Configuration lock commit for GPIO23<br>Reset type: SYSRSn |
| 22 | GPIO22 | R/WSOnce | 0h | Configuration lock commit for GPIO22<br>Reset type: SYSRSn |
| 21 | GPIO21 | R/WSOnce | 0h | Configuration lock commit for GPIO21<br>Reset type: SYSRSn |
| 20 | GPIO20 | R/WSOnce | 0h | Configuration lock commit for GPIO20<br>Reset type: SYSRSn |

**Table 8-27. GPACR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19 | GPIO19 | R/WSOnce | 0h | Configuration lock commit for GPIO19<br>Reset type: SYSRSn |
| 18 | GPIO18 | R/WSOnce | 0h | Configuration lock commit for GPIO18<br>Reset type: SYSRSn |
| 17 | GPIO17 | R/WSOnce | 0h | Configuration lock commit for GPIO17<br>Reset type: SYSRSn |
| 16 | GPIO16 | R/WSOnce | 0h | Configuration lock commit for GPIO16<br>Reset type: SYSRSn |
| 15 | GPIO15 | R/WSOnce | 0h | Configuration lock commit for GPIO15<br>Reset type: SYSRSn |
| 14 | GPIO14 | R/WSOnce | 0h | Configuration lock commit for GPIO14<br>Reset type: SYSRSn |
| 13 | GPIO13 | R/WSOnce | 0h | Configuration lock commit for GPIO13<br>Reset type: SYSRSn |
| 12 | GPIO12 | R/WSOnce | 0h | Configuration lock commit for GPIO12<br>Reset type: SYSRSn |
| 11 | GPIO11 | R/WSOnce | 0h | Configuration lock commit for GPIO11<br>Reset type: SYSRSn |
| 10 | GPIO10 | R/WSOnce | 0h | Configuration lock commit for GPIO10<br>Reset type: SYSRSn |
| 9 | GPIO9 | R/WSOnce | 0h | Configuration lock commit for GPIO9<br>Reset type: SYSRSn |
| 8 | GPIO8 | R/WSOnce | 0h | Configuration lock commit for GPIO8<br>Reset type: SYSRSn |
| 7 | GPIO7 | R/WSOnce | 0h | Configuration lock commit for GPIO7<br>Reset type: SYSRSn |
| 6 | GPIO6 | R/WSOnce | 0h | Configuration lock commit for GPIO6<br>Reset type: SYSRSn |
| 5 | GPIO5 | R/WSOnce | 0h | Configuration lock commit for GPIO5<br>Reset type: SYSRSn |
| 4 | GPIO4 | R/WSOnce | 0h | Configuration lock commit for GPIO4<br>Reset type: SYSRSn |
| 3 | GPIO3 | R/WSOnce | 0h | Configuration lock commit for GPIO3<br>Reset type: SYSRSn |
| 2 | GPIO2 | R/WSOnce | 0h | Configuration lock commit for GPIO2<br>Reset type: SYSRSn |
| 1 | GPIO1 | R/WSOnce | 0h | Configuration lock commit for GPIO1<br>Reset type: SYSRSn |
| 0 | GPIO0 | R/WSOnce | 0h | Configuration lock commit for GPIO0<br>Reset type: SYSRSn |

#### 8.7.1.1.19 GPBCTRL Register (Offset = 40h) [reset = 0h]

GPBCTRL is shown in Figure 8-22 and described in Table 8-28.

Return to Summary Table.

GPIO B Qualification Sampling Period (GPIO32 to GPIO63)

Each field in this register selects the qualification sampling period in SYSCLK cycles for eight GPIOs. The period is equal to 2 times the register field value.

0x00: Period = 0 SYSCLK cycles
0x01: Period = 2 SYSCLK cycles
0x02: Period = 4 SYSCLK cycles
...
0xFF: Period = 510 SYSCLK cycles

**Figure 8-22. GPBCTRL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QUALPRD3 | | | | | | | | QUALPRD2 | | | | | | | | QUALPRD1 | | | | | | | | QUALPRD0 | | | | | | | |
| R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | |

**Table 8-28. GPBCTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | QUALPRD3 | R/W | 0h | Qualification sampling period for GPIO56 to GPIO63<br>Reset type: SYSRSn |
| 23-16 | QUALPRD2 | R/W | 0h | Qualification sampling period for GPIO48 to GPIO55<br>Reset type: SYSRSn |
| 15-8 | QUALPRD1 | R/W | 0h | Qualification sampling period for GPIO40 to GPIO47<br>Reset type: SYSRSn |
| 7-0 | QUALPRD0 | R/W | 0h | Qualification sampling period for GPIO32 to GPIO39<br>Reset type: SYSRSn |

#### 8.7.1.1.20 GPBQSEL1 Register (Offset = 42h) [reset = 0h]

GPBQSEL1 is shown in Figure 8-23 and described in Table 8-29.

Return to Summary Table.

GPIO B Qualification Type (GPIO32 to GPIO47)

Each field in this register selects the input qualification type for one IO pin. The available types are:

0: Synchronous
1: 3-sample qualification
2: 6-sample qualification
3: Asynchronous

**Figure 8-23. GPBQSEL1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO47 | | GPIO46 | | GPIO45 | | GPIO44 | | GPIO43 | | GPIO42 | | GPIO41 | | GPIO40 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO39 | | RESERVED | | GPIO37 | | RESERVED | | GPIO35 | | GPIO34 | | GPIO33 | | GPIO32 | |
| R/W-0h | | R-0h | | R/W-0h | | R-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-29. GPBQSEL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO47 | R/W | 0h | Input qualification type for GPIO47<br>Reset type: SYSRSn |
| 29-28 | GPIO46 | R/W | 0h | Input qualification type for GPIO46<br>Reset type: SYSRSn |
| 27-26 | GPIO45 | R/W | 0h | Input qualification type for GPIO45<br>Reset type: SYSRSn |
| 25-24 | GPIO44 | R/W | 0h | Input qualification type for GPIO44<br>Reset type: SYSRSn |
| 23-22 | GPIO43 | R/W | 0h | Input qualification type for GPIO43<br>Reset type: SYSRSn |
| 21-20 | GPIO42 | R/W | 0h | Input qualification type for GPIO42<br>Reset type: SYSRSn |
| 19-18 | GPIO41 | R/W | 0h | Input qualification type for GPIO41<br>Reset type: SYSRSn |
| 17-16 | GPIO40 | R/W | 0h | Input qualification type for GPIO40<br>Reset type: SYSRSn |
| 15-14 | GPIO39 | R/W | 0h | Input qualification type for GPIO39<br>Reset type: SYSRSn |
| 13-12 | RESERVED | R | 0h | Reserved |
| 11-10 | GPIO37 | R/W | 0h | Input qualification type for GPIO37<br>Reset type: SYSRSn |
| 9-8 | RESERVED | R | 0h | Reserved |
| 7-6 | GPIO35 | R/W | 0h | Input qualification type for GPIO35<br>Reset type: SYSRSn |
| 5-4 | GPIO34 | R/W | 0h | Input qualification type for GPIO34<br>Reset type: SYSRSn |
| 3-2 | GPIO33 | R/W | 0h | Input qualification type for GPIO33<br>Reset type: SYSRSn |

**Table 8-29. GPBQSEL1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | GPIO32 | R/W | 0h | Input qualification type for GPIO32<br>Reset type: SYSRSn |

#### 8.7.1.1.21 GPBQSEL2 Register (Offset = 44h) [reset = 0h]

GPBQSEL2 is shown in Figure 8-24 and described in Table 8-30.

Return to Summary Table.

GPIO B Qualification Type (GPIO48 to GPIO63)

Each field in this register determines the input qualification type for one IO pin. The available types are:

0: Synchronous
1: 3-sample qualification
2: 6-sample qualification
3: Asynchronous

**Figure 8-24. GPBQSEL2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | RESERVED | | RESERVED | | RESERVED | | GPIO59 | | GPIO58 | | GPIO57 | | GPIO56 | |
| R-0h | | R-0h | | R-0h | | R-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 8-30. GPBQSEL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | RESERVED | R | 0h | Reserved |
| 29-28 | RESERVED | R | 0h | Reserved |
| 27-26 | RESERVED | R | 0h | Reserved |
| 25-24 | RESERVED | R | 0h | Reserved |
| 23-22 | GPIO59 | R/W | 0h | Input qualification type for GPIO59<br>Reset type: SYSRSn |
| 21-20 | GPIO58 | R/W | 0h | Input qualification type for GPIO58<br>Reset type: SYSRSn |
| 19-18 | GPIO57 | R/W | 0h | Input qualification type for GPIO57<br>Reset type: SYSRSn |
| 17-16 | GPIO56 | R/W | 0h | Input qualification type for GPIO56<br>Reset type: SYSRSn |
| 15-14 | GPIO55 | R/W | 0h | Input qualification type for GPIO55<br>Reset type: SYSRSn |
| 13-12 | GPIO54 | R/W | 0h | Input qualification type for GPIO54<br>Reset type: SYSRSn |
| 11-10 | GPIO53 | R/W | 0h | Input qualification type for GPIO53<br>Reset type: SYSRSn |
| 9-8 | GPIO52 | R/W | 0h | Input qualification type for GPIO52<br>Reset type: SYSRSn |
| 7-6 | GPIO51 | R/W | 0h | Input qualification type for GPIO51<br>Reset type: SYSRSn |
| 5-4 | GPIO50 | R/W | 0h | Input qualification type for GPIO50<br>Reset type: SYSRSn |
| 3-2 | GPIO49 | R/W | 0h | Input qualification type for GPIO49<br>Reset type: SYSRSn |
| 1-0 | GPIO48 | R/W | 0h | Input qualification type for GPIO48<br>Reset type: SYSRSn |

#### 8.7.1.1.22 GPBMUX1 Register (Offset = 46h) [reset = CC0h]

GPBMUX1 is shown in Figure 8-25 and described in Table 8-31.

Return to Summary Table.

GPIO B Peripheral Mux (GPIO32 to GPIO47)

Each field in this register determines part of the GPIO mux configuration for one IO pin. A value of 0x0, 0x4, 0x8, or 0xC configures the pin as a general-purpose IO. All other values select a peripheral to control the pin. See the device datasheet for a table of peripheral mux options. Pins must be set to GPIO mode using this register before changing the corresponding field in the GPBGMUX1 register.

**Figure 8-25. GPBMUX1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO47 | | GPIO46 | | GPIO45 | | GPIO44 | | GPIO43 | | GPIO42 | | GPIO41 | | GPIO40 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO39 | | RESERVED | | GPIO37 | | RESERVED | | GPIO35 | | GPIO34 | | GPIO33 | | GPIO32 | |
| R/W-0h | | R-0h | | R/W-3h | | R-0h | | R/W-3h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-31. GPBMUX1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO47 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO47<br>Reset type: SYSRSn |
| 29-28 | GPIO46 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO46<br>Reset type: SYSRSn |
| 27-26 | GPIO45 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO45<br>Reset type: SYSRSn |
| 25-24 | GPIO44 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO44<br>Reset type: SYSRSn |
| 23-22 | GPIO43 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO43<br>Reset type: SYSRSn |
| 21-20 | GPIO42 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO42<br>Reset type: SYSRSn |
| 19-18 | GPIO41 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO41<br>Reset type: SYSRSn |
| 17-16 | GPIO40 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO40<br>Reset type: SYSRSn |
| 15-14 | GPIO39 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO39<br>Reset type: SYSRSn |
| 13-12 | RESERVED | R | 0h | Reserved |
| 11-10 | GPIO37 | R/W | 3h | Lower 2 bits of peripheral mux configuration for GPIO37<br>Reset type: SYSRSn |
| 9-8 | RESERVED | R | 0h | Reserved |
| 7-6 | GPIO35 | R/W | 3h | Lower 2 bits of peripheral mux configuration for GPIO35<br>Reset type: SYSRSn |
| 5-4 | GPIO34 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO34<br>Reset type: SYSRSn |
| 3-2 | GPIO33 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO33<br>Reset type: SYSRSn |
| 1-0 | GPIO32 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO32<br>Reset type: SYSRSn |

#### 8.7.1.1.23 GPBMUX2 Register (Offset = 48h) [reset = 0h]

GPBMUX2 is shown in Figure 8-26 and described in Table 8-32.

Return to Summary Table.

GPIO B Peripheral Mux (GPIO48 to GPIO63)

Each field in this register determines part of the GPIO mux configuration for one IO pin. A value of 0x0, 0x4, 0x8, or 0xC configures the pin as a general-purpose IO. All other values select a peripheral to control the pin. See the device datasheet for a table of peripheral mux options. Pins must be set to GPIO mode using this register before changing the corresponding field in the GPBGMUX2 register.

**Figure 8-26. GPBMUX2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | RESERVED | | RESERVED | | RESERVED | | GPIO59 | | GPIO58 | | GPIO57 | | GPIO56 | |
| R-0h | | R-0h | | R-0h | | R-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 8-32. GPBMUX2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | RESERVED | R | 0h | Reserved |
| 29-28 | RESERVED | R | 0h | Reserved |
| 27-26 | RESERVED | R | 0h | Reserved |
| 25-24 | RESERVED | R | 0h | Reserved |
| 23-22 | GPIO59 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO59<br>Reset type: SYSRSn |
| 21-20 | GPIO58 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO58<br>Reset type: SYSRSn |
| 19-18 | GPIO57 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO57<br>Reset type: SYSRSn |
| 17-16 | GPIO56 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO56<br>Reset type: SYSRSn |
| 15-14 | GPIO55 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO55<br>Reset type: SYSRSn |
| 13-12 | GPIO54 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO54<br>Reset type: SYSRSn |
| 11-10 | GPIO53 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO53<br>Reset type: SYSRSn |
| 9-8 | GPIO52 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO52<br>Reset type: SYSRSn |
| 7-6 | GPIO51 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO51<br>Reset type: SYSRSn |
| 5-4 | GPIO50 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO50<br>Reset type: SYSRSn |
| 3-2 | GPIO49 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO49<br>Reset type: SYSRSn |
| 1-0 | GPIO48 | R/W | 0h | Lower 2 bits of peripheral mux configuration for GPIO48<br>Reset type: SYSRSn |

#### 8.7.1.1.24 GPBDIR Register (Offset = 4Ah) [reset = 0h]

GPBDIR is shown in Figure 8-27 and described in Table 8-33.

Return to Summary Table.

GPIO B Direction (GPIO32 to GPIO63)

Each field in this register selects the direction of one IO pin in GPIO mode. If the pin is not configured as a general-purpose IO, this register has no effect.

0: The pin is an input
1: The pin is an output

**Figure 8-27. GPBDIR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | GPIO59 | GPIO58 | GPIO57 | GPIO56 |
| R-0h | R-0h | R-0h | R-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 | RESERVED | GPIO37 | RESERVED | GPIO35 | GPIO34 | GPIO33 | GPIO32 |
| R/W-0h | R-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-33. GPBDIR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RESERVED | R | 0h | Reserved |
| 30 | RESERVED | R | 0h | Reserved |
| 29 | RESERVED | R | 0h | Reserved |
| 28 | RESERVED | R | 0h | Reserved |
| 27 | GPIO59 | R/W | 0h | Data direction for GPIO59<br>Reset type: SYSRSn |
| 26 | GPIO58 | R/W | 0h | Data direction for GPIO58<br>Reset type: SYSRSn |
| 25 | GPIO57 | R/W | 0h | Data direction for GPIO57<br>Reset type: SYSRSn |
| 24 | GPIO56 | R/W | 0h | Data direction for GPIO56<br>Reset type: SYSRSn |
| 23 | GPIO55 | R/W | 0h | Data direction for GPIO55<br>Reset type: SYSRSn |
| 22 | GPIO54 | R/W | 0h | Data direction for GPIO54<br>Reset type: SYSRSn |
| 21 | GPIO53 | R/W | 0h | Data direction for GPIO53<br>Reset type: SYSRSn |
| 20 | GPIO52 | R/W | 0h | Data direction for GPIO52<br>Reset type: SYSRSn |
| 19 | GPIO51 | R/W | 0h | Data direction for GPIO51<br>Reset type: SYSRSn |
| 18 | GPIO50 | R/W | 0h | Data direction for GPIO50<br>Reset type: SYSRSn |

**Table 8-33. GPBDIR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 17 | GPIO49 | R/W | 0h | Data direction for GPIO49<br>Reset type: SYSRSn |
| 16 | GPIO48 | R/W | 0h | Data direction for GPIO48<br>Reset type: SYSRSn |
| 15 | GPIO47 | R/W | 0h | Data direction for GPIO47<br>Reset type: SYSRSn |
| 14 | GPIO46 | R/W | 0h | Data direction for GPIO46<br>Reset type: SYSRSn |
| 13 | GPIO45 | R/W | 0h | Data direction for GPIO45<br>Reset type: SYSRSn |
| 12 | GPIO44 | R/W | 0h | Data direction for GPIO44<br>Reset type: SYSRSn |
| 11 | GPIO43 | R/W | 0h | Data direction for GPIO43<br>Reset type: SYSRSn |
| 10 | GPIO42 | R/W | 0h | Data direction for GPIO42<br>Reset type: SYSRSn |
| 9 | GPIO41 | R/W | 0h | Data direction for GPIO41<br>Reset type: SYSRSn |
| 8 | GPIO40 | R/W | 0h | Data direction for GPIO40<br>Reset type: SYSRSn |
| 7 | GPIO39 | R/W | 0h | Data direction for GPIO39<br>Reset type: SYSRSn |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | GPIO37 | R/W | 0h | Data direction for GPIO37<br>Reset type: SYSRSn |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | GPIO35 | R/W | 0h | Data direction for GPIO35<br>Reset type: SYSRSn |
| 2 | GPIO34 | R/W | 0h | Data direction for GPIO34<br>Reset type: SYSRSn |
| 1 | GPIO33 | R/W | 0h | Data direction for GPIO33<br>Reset type: SYSRSn |
| 0 | GPIO32 | R/W | 0h | Data direction for GPIO32<br>Reset type: SYSRSn |

#### 8.7.1.1.25 GPBPUD Register (Offset = 4Ch) [reset = FFFFFFFFh]

GPBPUD is shown in Figure 8-28 and described in Table 8-34.

Return to Summary Table.

GPIO B Pull-Up Disable (GPIO32 to GPIO63)

Each field in this register selects the state of the internal pull-up resistor for a single IO pin.

0: Pull-up enabled
1: Pull-up disabled

**Figure 8-28. GPBPUD Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | GPIO59 | GPIO58 | GPIO57 | GPIO56 |
| R-0h | R-0h | R-0h | R-0h | 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 | RESERVED | GPIO37 | RESERVED | GPIO35 | GPIO34 | GPIO33 | GPIO32 |
| R/W-1h | R-0h | R/W-1h | R-0h | R/W-1h | R/W-1h | R/W-1h | R/W-1h |

**Table 8-34. GPBPUD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RESERVED | R | 0h | Reserved |
| 30 | RESERVED | R | 0h | Reserved |
| 29 | RESERVED | R | 0h | Reserved |
| 28 | RESERVED | R | 0h | Reserved |
| 27 | GPIO59 | R/W | 1h | Pull-up disable for GPIO59<br>Reset type: SYSRSn |
| 26 | GPIO58 | R/W | 1h | Pull-up disable for GPIO58<br>Reset type: SYSRSn |
| 25 | GPIO57 | R/W | 1h | Pull-up disable for GPIO57<br>Reset type: SYSRSn |
| 24 | GPIO56 | R/W | 1h | Pull-up disable for GPIO56<br>Reset type: SYSRSn |
| 23 | GPIO55 | R/W | 1h | Pull-up disable for GPIO55<br>Reset type: SYSRSn |
| 22 | GPIO54 | R/W | 1h | Pull-up disable for GPIO54<br>Reset type: SYSRSn |
| 21 | GPIO53 | R/W | 1h | Pull-up disable for GPIO53<br>Reset type: SYSRSn |
| 20 | GPIO52 | R/W | 1h | Pull-up disable for GPIO52<br>Reset type: SYSRSn |
| 19 | GPIO51 | R/W | 1h | Pull-up disable for GPIO51<br>Reset type: SYSRSn |
| 18 | GPIO50 | R/W | 1h | Pull-up disable for GPIO50<br>Reset type: SYSRSn |

**Table 8-34. GPBPUD Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 17 | GPIO49 | R/W | 1h | Pull-up disable for GPIO49<br>Reset type: SYSRSn |
| 16 | GPIO48 | R/W | 1h | Pull-up disable for GPIO48<br>Reset type: SYSRSn |
| 15 | GPIO47 | R/W | 1h | Pull-up disable for GPIO47<br>Reset type: SYSRSn |
| 14 | GPIO46 | R/W | 1h | Pull-up disable for GPIO46<br>Reset type: SYSRSn |
| 13 | GPIO45 | R/W | 1h | Pull-up disable for GPIO45<br>Reset type: SYSRSn |
| 12 | GPIO44 | R/W | 1h | Pull-up disable for GPIO44<br>Reset type: SYSRSn |
| 11 | GPIO43 | R/W | 1h | Pull-up disable for GPIO43<br>Reset type: SYSRSn |
| 10 | GPIO42 | R/W | 1h | Pull-up disable for GPIO42<br>Reset type: SYSRSn |
| 9 | GPIO41 | R/W | 1h | Pull-up disable for GPIO41<br>Reset type: SYSRSn |
| 8 | GPIO40 | R/W | 1h | Pull-up disable for GPIO40<br>Reset type: SYSRSn |
| 7 | GPIO39 | R/W | 1h | Pull-up disable for GPIO39<br>Reset type: SYSRSn |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | GPIO37 | R/W | 1h | Pull-up disable for GPIO37<br>Reset type: SYSRSn |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | GPIO35 | R/W | 1h | Pull-up disable for GPIO35<br>Reset type: SYSRSn |
| 2 | GPIO34 | R/W | 1h | Pull-up disable for GPIO34<br>Reset type: SYSRSn |
| 1 | GPIO33 | R/W | 1h | Pull-up disable for GPIO33<br>Reset type: SYSRSn |
| 0 | GPIO32 | R/W | 1h | Pull-up disable for GPIO32<br>Reset type: SYSRSn |

#### 8.7.1.1.26 GPBINV Register (Offset = 50h) [reset = 0h]

GPBINV is shown in Figure 8-29 and described in Table 8-35.

Return to Summary Table.

GPIO B Input Inversion (GPIO32 to GPIO63)

Each field in this register selects whether the input value of one IO pin passes through an inverter.

0: The input is not inverted
1: The input is inverted

**Figure 8-29. GPBINV Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | GPIO59 | GPIO58 | GPIO57 | GPIO56 |
| R-0h | R-0h | R-0h | R-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 | RESERVED | GPIO37 | RESERVED | GPIO35 | GPIO34 | GPIO33 | GPIO32 |
| R/W-0h | R-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-35. GPBINV Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RESERVED | R | 0h | Reserved |
| 30 | RESERVED | R | 0h | Reserved |
| 29 | RESERVED | R | 0h | Reserved |
| 28 | RESERVED | R | 0h | Reserved |
| 27 | GPIO59 | R/W | 0h | Input inversion for GPIO59<br>Reset type: SYSRSn |
| 26 | GPIO58 | R/W | 0h | Input inversion for GPIO58<br>Reset type: SYSRSn |
| 25 | GPIO57 | R/W | 0h | Input inversion for GPIO57<br>Reset type: SYSRSn |
| 24 | GPIO56 | R/W | 0h | Input inversion for GPIO56<br>Reset type: SYSRSn |
| 23 | GPIO55 | R/W | 0h | Input inversion for GPIO55<br>Reset type: SYSRSn |
| 22 | GPIO54 | R/W | 0h | Input inversion for GPIO54<br>Reset type: SYSRSn |
| 21 | GPIO53 | R/W | 0h | Input inversion for GPIO53<br>Reset type: SYSRSn |
| 20 | GPIO52 | R/W | 0h | Input inversion for GPIO52<br>Reset type: SYSRSn |
| 19 | GPIO51 | R/W | 0h | Input inversion for GPIO51<br>Reset type: SYSRSn |
| 18 | GPIO50 | R/W | 0h | Input inversion for GPIO50<br>Reset type: SYSRSn |

**Table 8-35. GPBINV Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 17 | GPIO49 | R/W | 0h | Input inversion for GPIO49<br>Reset type: SYSRSn |
| 16 | GPIO48 | R/W | 0h | Input inversion for GPIO48<br>Reset type: SYSRSn |
| 15 | GPIO47 | R/W | 0h | Input inversion for GPIO47<br>Reset type: SYSRSn |
| 14 | GPIO46 | R/W | 0h | Input inversion for GPIO46<br>Reset type: SYSRSn |
| 13 | GPIO45 | R/W | 0h | Input inversion for GPIO45<br>Reset type: SYSRSn |
| 12 | GPIO44 | R/W | 0h | Input inversion for GPIO44<br>Reset type: SYSRSn |
| 11 | GPIO43 | R/W | 0h | Input inversion for GPIO43<br>Reset type: SYSRSn |
| 10 | GPIO42 | R/W | 0h | Input inversion for GPIO42<br>Reset type: SYSRSn |
| 9 | GPIO41 | R/W | 0h | Input inversion for GPIO41<br>Reset type: SYSRSn |
| 8 | GPIO40 | R/W | 0h | Input inversion for GPIO40<br>Reset type: SYSRSn |
| 7 | GPIO39 | R/W | 0h | Input inversion for GPIO39<br>Reset type: SYSRSn |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | GPIO37 | R/W | 0h | Input inversion for GPIO37<br>Reset type: SYSRSn |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | GPIO35 | R/W | 0h | Input inversion for GPIO35<br>Reset type: SYSRSn |
| 2 | GPIO34 | R/W | 0h | Input inversion for GPIO34<br>Reset type: SYSRSn |
| 1 | GPIO33 | R/W | 0h | Input inversion for GPIO33<br>Reset type: SYSRSn |
| 0 | GPIO32 | R/W | 0h | Input inversion for GPIO32<br>Reset type: SYSRSn |

#### 8.7.1.1.27 GPBODR Register (Offset = 52h) [reset = 0h]

GPBODR is shown in Figure 8-30 and described in Table 8-36.

Return to Summary Table.

GPIO B Open Drain Output Mode (GPIO32 to GPIO63)

Each field in this register selects between push-pull mode and open-drain mode for one general-purpose output pin. In both modes, writing a 0 to the output data latch drives the pin low. In push-pull mode, writing a 1 to the output data latch drives the pin high. In open-drain mode, it tri-states the output buffer.

0: Push-pull output
1: Open-drain output

**Figure 8-30. GPBODR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | GPIO59 | GPIO58 | GPIO57 | GPIO56 |
| R-0h | R-0h | R-0h | R-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 | RESERVED | GPIO37 | RESERVED | GPIO35 | GPIO34 | GPIO33 | GPIO32 |
| R/W-0h | R-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-36. GPBODR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RESERVED | R | 0h | Reserved |
| 30 | RESERVED | R | 0h | Reserved |
| 29 | RESERVED | R | 0h | Reserved |
| 28 | RESERVED | R | 0h | Reserved |
| 27 | GPIO59 | R/W | 0h | Open-drain output mode for GPIO59<br>Reset type: SYSRSn |
| 26 | GPIO58 | R/W | 0h | Open-drain output mode for GPIO58<br>Reset type: SYSRSn |
| 25 | GPIO57 | R/W | 0h | Open-drain output mode for GPIO57<br>Reset type: SYSRSn |
| 24 | GPIO56 | R/W | 0h | Open-drain output mode for GPIO56<br>Reset type: SYSRSn |
| 23 | GPIO55 | R/W | 0h | Open-drain output mode for GPIO55<br>Reset type: SYSRSn |
| 22 | GPIO54 | R/W | 0h | Open-drain output mode for GPIO54<br>Reset type: SYSRSn |
| 21 | GPIO53 | R/W | 0h | Open-drain output mode for GPIO53<br>Reset type: SYSRSn |
| 20 | GPIO52 | R/W | 0h | Open-drain output mode for GPIO52<br>Reset type: SYSRSn |
| 19 | GPIO51 | R/W | 0h | Open-drain output mode for GPIO51<br>Reset type: SYSRSn |

**Table 8-36. GPBODR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18 | GPIO50 | R/W | 0h | Open-drain output mode for GPIO50<br>Reset type: SYSRSn |
| 17 | GPIO49 | R/W | 0h | Open-drain output mode for GPIO49<br>Reset type: SYSRSn |
| 16 | GPIO48 | R/W | 0h | Open-drain output mode for GPIO48<br>Reset type: SYSRSn |
| 15 | GPIO47 | R/W | 0h | Open-drain output mode for GPIO47<br>Reset type: SYSRSn |
| 14 | GPIO46 | R/W | 0h | Open-drain output mode for GPIO46<br>Reset type: SYSRSn |
| 13 | GPIO45 | R/W | 0h | Open-drain output mode for GPIO45<br>Reset type: SYSRSn |
| 12 | GPIO44 | R/W | 0h | Open-drain output mode for GPIO44<br>Reset type: SYSRSn |
| 11 | GPIO43 | R/W | 0h | Open-drain output mode for GPIO43<br>Reset type: SYSRSn |
| 10 | GPIO42 | R/W | 0h | Open-drain output mode for GPIO42<br>Reset type: SYSRSn |
| 9 | GPIO41 | R/W | 0h | Open-drain output mode for GPIO41<br>Reset type: SYSRSn |
| 8 | GPIO40 | R/W | 0h | Open-drain output mode for GPIO40<br>Reset type: SYSRSn |
| 7 | GPIO39 | R/W | 0h | Open-drain output mode for GPIO39<br>Reset type: SYSRSn |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | GPIO37 | R/W | 0h | Open-drain output mode for GPIO37<br>Reset type: SYSRSn |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | GPIO35 | R/W | 0h | Open-drain output mode for GPIO35<br>Reset type: SYSRSn |
| 2 | GPIO34 | R/W | 0h | Open-drain output mode for GPIO34<br>Reset type: SYSRSn |
| 1 | GPIO33 | R/W | 0h | Open-drain output mode for GPIO33<br>Reset type: SYSRSn |
| 0 | GPIO32 | R/W | 0h | Open-drain output mode for GPIO32<br>Reset type: SYSRSn |

#### 8.7.1.1.28 GPBGMUX1 Register (Offset = 60h) [reset = CC0h]

GPBGMUX1 is shown in Figure 8-31 and described in Table 8-37.

Return to Summary Table.

GPIO B Peripheral Group Mux (GPIO32 to GPIO47)

Each field in this register determines part of the GPIO mux configuration for one IO pin. See the device datasheet for a table of peripheral mux options. Pins must be set to GPIO mode using the GPBMUX1 register before changing their configuration in this register.

**Figure 8-31. GPBGMUX1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO47 | | GPIO46 | | GPIO45 | | GPIO44 | | GPIO43 | | GPIO42 | | GPIO41 | | GPIO40 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO39 | | RESERVED | | GPIO37 | | RESERVED | | GPIO35 | | GPIO34 | | GPIO33 | | GPIO32 | |
| R/W-0h | | R-0h | | R/W-3h | | R-0h | | R/W-3h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-37. GPBGMUX1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO47 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO47<br>Reset type: SYSRSn |
| 29-28 | GPIO46 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO46<br>Reset type: SYSRSn |
| 27-26 | GPIO45 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO45<br>Reset type: SYSRSn |
| 25-24 | GPIO44 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO44<br>Reset type: SYSRSn |
| 23-22 | GPIO43 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO43<br>Reset type: SYSRSn |
| 21-20 | GPIO42 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO42<br>Reset type: SYSRSn |
| 19-18 | GPIO41 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO41<br>Reset type: SYSRSn |
| 17-16 | GPIO40 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO40<br>Reset type: SYSRSn |
| 15-14 | GPIO39 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO39<br>Reset type: SYSRSn |
| 13-12 | RESERVED | R | 0h | Reserved |
| 11-10 | GPIO37 | R/W | 3h | Upper 2 bits of peripheral mux configuration for GPIO37<br>Reset type: SYSRSn |
| 9-8 | RESERVED | R | 0h | Reserved |
| 7-6 | GPIO35 | R/W | 3h | Upper 2 bits of peripheral mux configuration for GPIO35<br>Reset type: SYSRSn |
| 5-4 | GPIO34 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO34<br>Reset type: SYSRSn |
| 3-2 | GPIO33 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO33<br>Reset type: SYSRSn |
| 1-0 | GPIO32 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO32<br>Reset type: SYSRSn |

#### 8.7.1.1.29 GPBGMUX2 Register (Offset = 62h) [reset = 0h]

GPBGMUX2 is shown in Figure 8-32 and described in Table 8-38.

Return to Summary Table.

GPIO B Peripheral Group Mux (GPIO48 to GPIO63)

Each field in this register determines part of the GPIO mux configuration for a single IO pin. See the device datasheet for a table of peripheral mux options. Pins must be set to GPIO mode using the GPBMUX2 register before changing their configuration in this register.

**Figure 8-32. GPBGMUX2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | RESERVED | | RESERVED | | RESERVED | | GPIO59 | | GPIO58 | | GPIO57 | | GPIO56 | |
| R-0h | | R-0h | | R-0h | | R-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 8-38. GPBGMUX2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | RESERVED | R | 0h | Reserved |
| 29-28 | RESERVED | R | 0h | Reserved |
| 27-26 | RESERVED | R | 0h | Reserved |
| 25-24 | RESERVED | R | 0h | Reserved |
| 23-22 | GPIO59 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO59<br>Reset type: SYSRSn |
| 21-20 | GPIO58 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO58<br>Reset type: SYSRSn |
| 19-18 | GPIO57 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO57<br>Reset type: SYSRSn |
| 17-16 | GPIO56 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO56<br>Reset type: SYSRSn |
| 15-14 | GPIO55 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO55<br>Reset type: SYSRSn |
| 13-12 | GPIO54 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO54<br>Reset type: SYSRSn |
| 11-10 | GPIO53 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO53<br>Reset type: SYSRSn |
| 9-8 | GPIO52 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO52<br>Reset type: SYSRSn |
| 7-6 | GPIO51 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO51<br>Reset type: SYSRSn |
| 5-4 | GPIO50 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO50<br>Reset type: SYSRSn |
| 3-2 | GPIO49 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO49<br>Reset type: SYSRSn |
| 1-0 | GPIO48 | R/W | 0h | Upper 2 bits of peripheral mux configuration for GPIO48<br>Reset type: SYSRSn |

#### 8.7.1.1.30 GPBCSEL1 Register (Offset = 68h) [reset = 0h]

GPBCSEL1 is shown in Figure 8-33 and described in Table 8-39.

Return to Summary Table.

GPIO B Master Core Select (GPIO32 to GPIO39)

Each field in this register selects the master for one IO pin. The master controls the pin in GPIO mode via its GPBDAT, GPBSET, GPBCLEAR, and GPBTOGGLE registers.

0x0: CPU is the master
0x1: CLA is the master
0x2 - 0xF: Reserved

**Figure 8-33. GPBCSEL1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO39 | | | | RESERVED | | | | GPIO37 | | | | RESERVED | | | |
| R/W-0h | | | | R-0h | | | | R/W-0h | | | | R-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 8-39. GPBCSEL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO39 | R/W | 0h | Master core select for GPIO39<br>Reset type: SYSRSn |
| 27-24 | RESERVED | R | 0h | Reserved |
| 23-20 | GPIO37 | R/W | 0h | Master core select for GPIO37<br>Reset type: SYSRSn |
| 19-16 | RESERVED | R | 0h | Reserved |
| 15-12 | GPIO35 | R/W | 0h | Master core select for GPIO35<br>Reset type: SYSRSn |
| 11-8 | GPIO34 | R/W | 0h | Master core select for GPIO34<br>Reset type: SYSRSn |
| 7-4 | GPIO33 | R/W | 0h | Master core select for GPIO33<br>Reset type: SYSRSn |
| 3-0 | GPIO32 | R/W | 0h | Master core select for GPIO32<br>Reset type: SYSRSn |

#### 8.7.1.1.31 GPBCSEL2 Register (Offset = 6Ah) [reset = 0h]

GPBCSEL2 is shown in Figure 8-34 and described in Table 8-40.

Return to Summary Table.

GPIO B Master Core Select (GPIO40 to GPIO47)

Each field in this register selects the master for one IO pin. The master controls the pin in GPIO mode via its GPBDAT, GPBSET, GPBCLEAR, and GPBTOGGLE registers.

0x0: CPU is the master
0x1: CLA is the master
0x2 - 0xF: Reserved

**Figure 8-34. GPBCSEL2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO47 | | | | GPIO46 | | | | GPIO45 | | | | GPIO44 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO43 | | | | GPIO42 | | | | GPIO41 | | | | GPIO40 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-40. GPBCSEL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO47 | R/W | 0h | Master core select for GPIO47<br>Reset type: SYSRSn |
| 27-24 | GPIO46 | R/W | 0h | Master core select for GPIO46<br>Reset type: SYSRSn |
| 23-20 | GPIO45 | R/W | 0h | Master core select for GPIO45<br>Reset type: SYSRSn |
| 19-16 | GPIO44 | R/W | 0h | Master core select for GPIO44<br>Reset type: SYSRSn |
| 15-12 | GPIO43 | R/W | 0h | Master core select for GPIO43<br>Reset type: SYSRSn |
| 11-8 | GPIO42 | R/W | 0h | Master core select for GPIO42<br>Reset type: SYSRSn |
| 7-4 | GPIO41 | R/W | 0h | Master core select for GPIO41<br>Reset type: SYSRSn |
| 3-0 | GPIO40 | R/W | 0h | Master core select for GPIO40<br>Reset type: SYSRSn |

#### 8.7.1.1.32 GPBCSEL3 Register (Offset = 6Ch) [reset = 0h]

GPBCSEL3 is shown in Figure 8-35 and described in Table 8-41.

Return to Summary Table.

GPIO B Master Core Select (GPIO48 to GPIO55)

Each field in this register selects the master for one IO pin. The master controls the pin in GPIO mode via its GPBDAT, GPBSET, GPBCLEAR, and GPBTOGGLE registers.

0x0: CPU is the master
0x1: CLA is the master
0x2 - 0xF: Reserved

**Figure 8-35. GPBCSEL3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPIO55 | | | | GPIO54 | | | | GPIO53 | | | | GPIO52 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO51 | | | | GPIO50 | | | | GPIO49 | | | | GPIO48 | | | |
| R/W-0h | | | | R/W-0h | | | | R/W-0h | | | | R/W-0h | | | |

**Table 8-41. GPBCSEL3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GPIO55 | R/W | 0h | Master core select for GPIO55<br>Reset type: SYSRSn |
| 27-24 | GPIO54 | R/W | 0h | Master core select for GPIO54<br>Reset type: SYSRSn |
| 23-20 | GPIO53 | R/W | 0h | Master core select for GPIO53<br>Reset type: SYSRSn |
| 19-16 | GPIO52 | R/W | 0h | Master core select for GPIO52<br>Reset type: SYSRSn |
| 15-12 | GPIO51 | R/W | 0h | Master core select for GPIO51<br>Reset type: SYSRSn |
| 11-8 | GPIO50 | R/W | 0h | Master core select for GPIO50<br>Reset type: SYSRSn |
| 7-4 | GPIO49 | R/W | 0h | Master core select for GPIO49<br>Reset type: SYSRSn |
| 3-0 | GPIO48 | R/W | 0h | Master core select for GPIO48<br>Reset type: SYSRSn |

#### 8.7.1.1.33 GPBCSEL4 Register (Offset = 6Eh) [reset = 0h]

GPBCSEL4 is shown in Figure 8-36 and described in Table 8-42.

Return to Summary Table.

GPIO B Master Core Select (GPIO56 to GPIO63)

Each field in this register selects the master for one IO pin. The master controls the pin in GPIO mode via its GPBDAT, GPBSET, GPBCLEAR, and GPBTOGGLE registers.

0x0: CPU is the master
0x1: CLA is the master
0x2 - 0xF: Reserved

**Figure 8-36. GPBCSEL4 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | RESERVED | | | | RESERVED | | | | RESERVED | | | |
| R-0h | | | | R-0h | | | | R-0h | | | | R-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 8-42. GPBCSEL4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | RESERVED | R | 0h | Reserved |
| 27-24 | RESERVED | R | 0h | Reserved |
| 23-20 | RESERVED | R | 0h | Reserved |
| 19-16 | RESERVED | R | 0h | Reserved |
| 15-12 | GPIO59 | R/W | 0h | Master core select for GPIO59<br>Reset type: SYSRSn |
| 11-8 | GPIO58 | R/W | 0h | Master core select for GPIO58<br>Reset type: SYSRSn |
| 7-4 | GPIO57 | R/W | 0h | Master core select for GPIO57<br>Reset type: SYSRSn |
| 3-0 | GPIO56 | R/W | 0h | Master core select for GPIO56<br>Reset type: SYSRSn |

#### 8.7.1.1.34 GPBLOCK Register (Offset = 7Ch) [reset = 0h]

GPBLOCK is shown in Figure 8-37 and described in Table 8-43.

Return to Summary Table.

GPIO B Lock Register (GPIO32 to GPIO63)

Each field in this register locks one IO pin's configuration. This blocks writes to the corresponding bits in the GPBMUXn, GPBIDR, GPBINV, GPBODR, GPBGMUXn, and GPBCSELn registers.

0: Pin configuration is unlocked
1: Pin configuration is locked

**Figure 8-37. GPBLOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | GPIO59 | GPIO58 | GPIO57 | GPIO56 |
| R-0h | R-0h | R-0h | R-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 | RESERVED | GPIO37 | RESERVED | GPIO35 | GPIO34 | GPIO33 | GPIO32 |
| R/W-0h | R-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-43. GPBLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RESERVED | R | 0h | Reserved |
| 30 | RESERVED | R | 0h | Reserved |
| 29 | RESERVED | R | 0h | Reserved |
| 28 | RESERVED | R | 0h | Reserved |
| 27 | GPIO59 | R/W | 0h | Configuration lock for GPIO59<br>Reset type: SYSRSn |
| 26 | GPIO58 | R/W | 0h | Configuration lock for GPIO58<br>Reset type: SYSRSn |
| 25 | GPIO57 | R/W | 0h | Configuration lock for GPIO57<br>Reset type: SYSRSn |
| 24 | GPIO56 | R/W | 0h | Configuration lock for GPIO56<br>Reset type: SYSRSn |
| 23 | GPIO55 | R/W | 0h | Configuration lock for GPIO55<br>Reset type: SYSRSn |
| 22 | GPIO54 | R/W | 0h | Configuration lock for GPIO54<br>Reset type: SYSRSn |
| 21 | GPIO53 | R/W | 0h | Configuration lock for GPIO53<br>Reset type: SYSRSn |
| 20 | GPIO52 | R/W | 0h | Configuration lock for GPIO52<br>Reset type: SYSRSn |
| 19 | GPIO51 | R/W | 0h | Configuration lock for GPIO51<br>Reset type: SYSRSn |
| 18 | GPIO50 | R/W | 0h | Configuration lock for GPIO50<br>Reset type: SYSRSn |

**Table 8-43. GPBLOCK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 17 | GPIO49 | R/W | 0h | Configuration lock for GPIO49<br>Reset type: SYSRSn |
| 16 | GPIO48 | R/W | 0h | Configuration lock for GPIO48<br>Reset type: SYSRSn |
| 15 | GPIO47 | R/W | 0h | Configuration lock for GPIO47<br>Reset type: SYSRSn |
| 14 | GPIO46 | R/W | 0h | Configuration lock for GPIO46<br>Reset type: SYSRSn |
| 13 | GPIO45 | R/W | 0h | Configuration lock for GPIO45<br>Reset type: SYSRSn |
| 12 | GPIO44 | R/W | 0h | Configuration lock for GPIO44<br>Reset type: SYSRSn |
| 11 | GPIO43 | R/W | 0h | Configuration lock for GPIO43<br>Reset type: SYSRSn |
| 10 | GPIO42 | R/W | 0h | Configuration lock for GPIO42<br>Reset type: SYSRSn |
| 9 | GPIO41 | R/W | 0h | Configuration lock for GPIO41<br>Reset type: SYSRSn |
| 8 | GPIO40 | R/W | 0h | Configuration lock for GPIO40<br>Reset type: SYSRSn |
| 7 | GPIO39 | R/W | 0h | Configuration lock for GPIO39<br>Reset type: SYSRSn |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | GPIO37 | R/W | 0h | Configuration lock for GPIO37<br>Reset type: SYSRSn |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | GPIO35 | R/W | 0h | Configuration lock for GPIO35<br>Reset type: SYSRSn |
| 2 | GPIO34 | R/W | 0h | Configuration lock for GPIO34<br>Reset type: SYSRSn |
| 1 | GPIO33 | R/W | 0h | Configuration lock for GPIO33<br>Reset type: SYSRSn |
| 0 | GPIO32 | R/W | 0h | Configuration lock for GPIO32<br>Reset type: SYSRSn |

#### 8.7.1.1.35 GPBCR Register (Offset = 7Eh) [reset = 0h]

GPBCR is shown in Figure 8-38 and described in Table 8-44.

Return to Summary Table.

GPIO B Lock Commit Register (GPIO32 to GPIO63)

Each field in this register blocks writes to one IO pin's GPBLOCK bit. Once set, a lock commit can only be cleared by a reset.

0: Pin configuration lock is unlocked
1: Pin configuration lock is locked

**Figure 8-38. GPBCR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | GPIO59 | GPIO58 | GPIO57 | GPIO56 |
| R-0h | R-0h | R-0h | R-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO55 | GPIO54 | GPIO53 | GPIO52 | GPIO51 | GPIO50 | GPIO49 | GPIO48 |
| 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 |
| GPIO47 | GPIO46 | GPIO45 | GPIO44 | GPIO43 | GPIO42 | GPIO41 | GPIO40 |
| 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 |
| GPIO39 | RESERVED | GPIO37 | RESERVED | GPIO35 | GPIO34 | GPIO33 | GPIO32 |
| R/WSOnce-0h | R-0h | R/WSOnce-0h | R-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h |

**Table 8-44. GPBCR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RESERVED | R | 0h | Reserved |
| 30 | RESERVED | R | 0h | Reserved |
| 29 | RESERVED | R | 0h | Reserved |
| 28 | RESERVED | R | 0h | Reserved |
| 27 | GPIO59 | R/WSOnce | 0h | Configuration lock commit for GPIO59<br>Reset type: SYSRSn |
| 26 | GPIO58 | R/WSOnce | 0h | Configuration lock commit for GPIO58<br>Reset type: SYSRSn |
| 25 | GPIO57 | R/WSOnce | 0h | Configuration lock commit for GPIO57<br>Reset type: SYSRSn |
| 24 | GPIO56 | R/WSOnce | 0h | Configuration lock commit for GPIO56<br>Reset type: SYSRSn |
| 23 | GPIO55 | R/WSOnce | 0h | Configuration lock commit for GPIO55<br>Reset type: SYSRSn |
| 22 | GPIO54 | R/WSOnce | 0h | Configuration lock commit for GPIO54<br>Reset type: SYSRSn |
| 21 | GPIO53 | R/WSOnce | 0h | Configuration lock commit for GPIO53<br>Reset type: SYSRSn |
| 20 | GPIO52 | R/WSOnce | 0h | Configuration lock commit for GPIO52<br>Reset type: SYSRSn |
| 19 | GPIO51 | R/WSOnce | 0h | Configuration lock commit for GPIO51<br>Reset type: SYSRSn |
| 18 | GPIO50 | R/WSOnce | 0h | Configuration lock commit for GPIO50<br>Reset type: SYSRSn |

**Table 8-44. GPBCR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 17 | GPIO49 | R/WSOnce | 0h | Configuration lock commit for GPIO49<br>Reset type: SYSRSn |
| 16 | GPIO48 | R/WSOnce | 0h | Configuration lock commit for GPIO48<br>Reset type: SYSRSn |
| 15 | GPIO47 | R/WSOnce | 0h | Configuration lock commit for GPIO47<br>Reset type: SYSRSn |
| 14 | GPIO46 | R/WSOnce | 0h | Configuration lock commit for GPIO46<br>Reset type: SYSRSn |
| 13 | GPIO45 | R/WSOnce | 0h | Configuration lock commit for GPIO45<br>Reset type: SYSRSn |
| 12 | GPIO44 | R/WSOnce | 0h | Configuration lock commit for GPIO44<br>Reset type: SYSRSn |
| 11 | GPIO43 | R/WSOnce | 0h | Configuration lock commit for GPIO43<br>Reset type: SYSRSn |
| 10 | GPIO42 | R/WSOnce | 0h | Configuration lock commit for GPIO42<br>Reset type: SYSRSn |
| 9 | GPIO41 | R/WSOnce | 0h | Configuration lock commit for GPIO41<br>Reset type: SYSRSn |
| 8 | GPIO40 | R/WSOnce | 0h | Configuration lock commit for GPIO40<br>Reset type: SYSRSn |
| 7 | GPIO39 | R/WSOnce | 0h | Configuration lock commit for GPIO39<br>Reset type: SYSRSn |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | GPIO37 | R/WSOnce | 0h | Configuration lock commit for GPIO37<br>Reset type: SYSRSn |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | GPIO35 | R/WSOnce | 0h | Configuration lock commit for GPIO35<br>Reset type: SYSRSn |
| 2 | GPIO34 | R/WSOnce | 0h | Configuration lock commit for GPIO34<br>Reset type: SYSRSn |
| 1 | GPIO33 | R/WSOnce | 0h | Configuration lock commit for GPIO33<br>Reset type: SYSRSn |
| 0 | GPIO32 | R/WSOnce | 0h | Configuration lock commit for GPIO32<br>Reset type: SYSRSn |

#### 8.7.1.1.36 GPHCTRL Register (Offset = 1C0h) [reset = 0h]

GPHCTRL is shown in Figure 8-39 and described in Table 8-45.

Return to Summary Table.

GPIO H Qualification Sampling Period (GPIO224 to GPIO255)

Each field in this register selects the qualification sampling period in SYSCLK cycles for eight GPIOs. The period is equal to 2 times the register field value.

0x00: Period = 0 SYSCLK cycles
0x01: Period = 2 SYSCLK cycles
0x02: Period = 4 SYSCLK cycles
...
0xFF: Period = 510 SYSCLK cycles

**Figure 8-39. GPHCTRL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | QUALPRD2 | | | | | | | | QUALPRD1 | | | | | | | | QUALPRD0 | | | | | | | |
| R-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | |

**Table 8-45. GPHCTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | RESERVED | R | 0h | Reserved |
| 23-16 | QUALPRD2 | R/W | 0h | Qualification sampling period for GPIO240 to GPIO247<br>Reset type: SYSRSn |
| 15-8 | QUALPRD1 | R/W | 0h | Qualification sampling period for GPIO232 to GPIO239<br>Reset type: SYSRSn |
| 7-0 | QUALPRD0 | R/W | 0h | Qualification sampling period for GPIO224 to GPIO231<br>Reset type: SYSRSn |

#### 8.7.1.1.37 GPHQSEL1 Register (Offset = 1C2h) [reset = 0h]

GPHQSEL1 is shown in Figure 8-40 and described in Table 8-46.

Return to Summary Table.

GPIO H Qualification Type (GPIO224 to GPIO239)

Each field in this register selects the input qualification type for one IO pin. The available types are:

0: Synchronous
1: 3-sample qualification
2: 6-sample qualification
3: Asynchronous

**Figure 8-40. GPHQSEL1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| GPIO239 | | GPIO238 | | GPIO237 | | GPIO236 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO235 | | GPIO234 | | GPIO233 | | GPIO232 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO231 | | GPIO230 | | GPIO229 | | GPIO228 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO227 | | GPIO226 | | GPIO225 | | GPIO224 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-46. GPHQSEL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | GPIO239 | R/W | 0h | Input qualification type for GPIO239<br>Reset type: SYSRSn |
| 29-28 | GPIO238 | R/W | 0h | Input qualification type for GPIO238<br>Reset type: SYSRSn |
| 27-26 | GPIO237 | R/W | 0h | Input qualification type for GPIO237<br>Reset type: SYSRSn |
| 25-24 | GPIO236 | R/W | 0h | Input qualification type for GPIO236<br>Reset type: SYSRSn |
| 23-22 | GPIO235 | R/W | 0h | Input qualification type for GPIO235<br>Reset type: SYSRSn |
| 21-20 | GPIO234 | R/W | 0h | Input qualification type for GPIO234<br>Reset type: SYSRSn |
| 19-18 | GPIO233 | R/W | 0h | Input qualification type for GPIO233<br>Reset type: SYSRSn |
| 17-16 | GPIO232 | R/W | 0h | Input qualification type for GPIO232<br>Reset type: SYSRSn |
| 15-14 | GPIO231 | R/W | 0h | Input qualification type for GPIO231<br>Reset type: SYSRSn |
| 13-12 | GPIO230 | R/W | 0h | Input qualification type for GPIO230<br>Reset type: SYSRSn |
| 11-10 | GPIO229 | R/W | 0h | Input qualification type for GPIO229<br>Reset type: SYSRSn |

**Table 8-46. GPHQSEL1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9-8 | GPIO228 | R/W | 0h | Input qualification type for GPIO228<br>Reset type: SYSRSn |
| 7-6 | GPIO227 | R/W | 0h | Input qualification type for GPIO227<br>Reset type: SYSRSn |
| 5-4 | GPIO226 | R/W | 0h | Input qualification type for GPIO226<br>Reset type: SYSRSn |
| 3-2 | GPIO225 | R/W | 0h | Input qualification type for GPIO225<br>Reset type: SYSRSn |
| 1-0 | GPIO224 | R/W | 0h | Input qualification type for GPIO224<br>Reset type: SYSRSn |

### 8.7.1.1.38 GPHQSEL2 Register (Offset = 1C4h) [reset = 0h]

GPHQSEL2 is shown in Figure 8-41 and described in Table 8-47.

Return to Summary Table.

GPIO H Qualification Type (GPIO240 to GPIO255)

Each field in this register determines the input qualification type for one IO pin. The available types are:

0: Synchronous
1: 3-sample qualification
2: 6-sample qualification
3: Asynchronous

**Figure 8-41. GPHQSEL2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | RESERVED | | RESERVED | | RESERVED | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | RESERVED | | RESERVED | | RESERVED | |
| R-0h | | R-0h | | R-0h | | R-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO247 | | GPIO246 | | GPIO245 | | GPIO244 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO243 | | GPIO242 | | GPIO241 | | GPIO240 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 8-47. GPHQSEL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | RESERVED | R | 0h | Reserved |
| 29-28 | RESERVED | R | 0h | Reserved |
| 27-26 | RESERVED | R | 0h | Reserved |
| 25-24 | RESERVED | R | 0h | Reserved |
| 23-22 | RESERVED | R | 0h | Reserved |
| 21-20 | RESERVED | R | 0h | Reserved |
| 19-18 | RESERVED | R | 0h | Reserved |
| 17-16 | RESERVED | R | 0h | Reserved |
| 15-14 | GPIO247 | R/W | 0h | Input qualification type for GPIO247<br>Reset type: SYSRSn |
| 13-12 | GPIO246 | R/W | 0h | Input qualification type for GPIO246<br>Reset type: SYSRSn |
| 11-10 | GPIO245 | R/W | 0h | Input qualification type for GPIO245<br>Reset type: SYSRSn |
| 9-8 | GPIO244 | R/W | 0h | Input qualification type for GPIO244<br>Reset type: SYSRSn |
| 7-6 | GPIO243 | R/W | 0h | Input qualification type for GPIO243<br>Reset type: SYSRSn |
| 5-4 | GPIO242 | R/W | 0h | Input qualification type for GPIO242<br>Reset type: SYSRSn |
| 3-2 | GPIO241 | R/W | 0h | Input qualification type for GPIO241<br>Reset type: SYSRSn |

**Table 8-47. GPHQSEL2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | GPIO240 | R/W | 0h | Input qualification type for GPIO240<br>Reset type: SYSRSn |

#### 8.7.1.1.39 GPHINV Register (Offset = 1D0h) [reset = 0h]

GPHINV is shown in Figure 8-42 and described in Table 8-48.

Return to Summary Table.

GPIO H Input Inversion (GPIO224 to GPIO255)

Each field in this register selects whether the input value of one IO pin passes through an inverter.

0: The input is not inverted
1: The input is inverted

**Figure 8-42. GPHINV Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO247 | GPIO246 | GPIO245 | GPIO244 | GPIO243 | GPIO242 | GPIO241 | GPIO240 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO239 | GPIO238 | GPIO237 | GPIO236 | GPIO235 | GPIO234 | GPIO233 | GPIO232 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO231 | GPIO230 | GPIO229 | GPIO228 | GPIO227 | GPIO226 | GPIO225 | GPIO224 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-48. GPHINV Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RESERVED | R | 0h | Reserved |
| 30 | RESERVED | R | 0h | Reserved |
| 29 | RESERVED | R | 0h | Reserved |
| 28 | RESERVED | R | 0h | Reserved |
| 27 | RESERVED | R | 0h | Reserved |
| 26 | RESERVED | R | 0h | Reserved |
| 25 | RESERVED | R | 0h | Reserved |
| 24 | RESERVED | R | 0h | Reserved |
| 23 | GPIO247 | R/W | 0h | Input inversion for GPIO247<br>Reset type: SYSRSn |
| 22 | GPIO246 | R/W | 0h | Input inversion for GPIO246<br>Reset type: SYSRSn |
| 21 | GPIO245 | R/W | 0h | Input inversion for GPIO245<br>Reset type: SYSRSn |
| 20 | GPIO244 | R/W | 0h | Input inversion for GPIO244<br>Reset type: SYSRSn |
| 19 | GPIO243 | R/W | 0h | Input inversion for GPIO243<br>Reset type: SYSRSn |
| 18 | GPIO242 | R/W | 0h | Input inversion for GPIO242<br>Reset type: SYSRSn |
| 17 | GPIO241 | R/W | 0h | Input inversion for GPIO241<br>Reset type: SYSRSn |
| 16 | GPIO240 | R/W | 0h | Input inversion for GPIO240<br>Reset type: SYSRSn |

**Table 8-48. GPHINV Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | GPIO239 | R/W | 0h | Input inversion for GPIO239<br>Reset type: SYSRSn |
| 14 | GPIO238 | R/W | 0h | Input inversion for GPIO238<br>Reset type: SYSRSn |
| 13 | GPIO237 | R/W | 0h | Input inversion for GPIO237<br>Reset type: SYSRSn |
| 12 | GPIO236 | R/W | 0h | Input inversion for GPIO236<br>Reset type: SYSRSn |
| 11 | GPIO235 | R/W | 0h | Input inversion for GPIO235<br>Reset type: SYSRSn |
| 10 | GPIO234 | R/W | 0h | Input inversion for GPIO234<br>Reset type: SYSRSn |
| 9 | GPIO233 | R/W | 0h | Input inversion for GPIO233<br>Reset type: SYSRSn |
| 8 | GPIO232 | R/W | 0h | Input inversion for GPIO232<br>Reset type: SYSRSn |
| 7 | GPIO231 | R/W | 0h | Input inversion for GPIO231<br>Reset type: SYSRSn |
| 6 | GPIO230 | R/W | 0h | Input inversion for GPIO230<br>Reset type: SYSRSn |
| 5 | GPIO229 | R/W | 0h | Input inversion for GPIO229<br>Reset type: SYSRSn |
| 4 | GPIO228 | R/W | 0h | Input inversion for GPIO228<br>Reset type: SYSRSn |
| 3 | GPIO227 | R/W | 0h | Input inversion for GPIO227<br>Reset type: SYSRSn |
| 2 | GPIO226 | R/W | 0h | Input inversion for GPIO226<br>Reset type: SYSRSn |
| 1 | GPIO225 | R/W | 0h | Input inversion for GPIO225<br>Reset type: SYSRSn |
| 0 | GPIO224 | R/W | 0h | Input inversion for GPIO224<br>Reset type: SYSRSn |

### 8.7.1.1.40 GPHAMSEL Register (Offset = 1D4h) [reset = FFFFFFFFh]

GPHAMSEL is shown in Figure 8-43 and described in Table 8-49.

Return to Summary Table.

GPIO H Analog Mode Select (GPIO224 to GPIO255)

Each field in this register selects between analog and digital functionality for one IO pin.

0: Digital mode
1: Analog mode

**Figure 8-43. GPHAMSEL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO247 | GPIO246 | GPIO245 | GPIO244 | GPIO243 | GPIO242 | GPIO241 | GPIO240 |
| R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO239 | GPIO238 | GPIO237 | GPIO236 | GPIO235 | GPIO234 | GPIO233 | GPIO232 |
| R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO231 | GPIO230 | GPIO229 | GPIO228 | GPIO227 | GPIO226 | GPIO225 | GPIO224 |
| R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h |

**Table 8-49. GPHAMSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RESERVED | R | 0h | Reserved |
| 30 | RESERVED | R | 0h | Reserved |
| 29 | RESERVED | R | 0h | Reserved |
| 28 | RESERVED | R | 0h | Reserved |
| 27 | RESERVED | R | 0h | Reserved |
| 26 | RESERVED | R | 0h | Reserved |
| 25 | RESERVED | R | 0h | Reserved |
| 24 | RESERVED | R | 0h | Reserved |
| 23 | GPIO247 | R/W | 1h | Analog mode select for GPIO247<br>Reset type: SYSRSn |
| 22 | GPIO246 | R/W | 1h | Analog mode select for GPIO246<br>Reset type: SYSRSn |
| 21 | GPIO245 | R/W | 1h | Analog mode select for GPIO245<br>Reset type: SYSRSn |
| 20 | GPIO244 | R/W | 1h | Analog mode select for GPIO244<br>Reset type: SYSRSn |
| 19 | GPIO243 | R/W | 1h | Analog mode select for GPIO243<br>Reset type: SYSRSn |
| 18 | GPIO242 | R/W | 1h | Analog mode select for GPIO242<br>Reset type: SYSRSn |
| 17 | GPIO241 | R/W | 1h | Analog mode select for GPIO241<br>Reset type: SYSRSn |
| 16 | GPIO240 | R/W | 1h | Analog mode select for GPIO240<br>Reset type: SYSRSn |

**Table 8-49. GPHAMSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | GPIO239 | R/W | 1h | Analog mode select for GPIO239<br>Reset type: SYSRSn |
| 14 | GPIO238 | R/W | 1h | Analog mode select for GPIO238<br>Reset type: SYSRSn |
| 13 | GPIO237 | R/W | 1h | Analog mode select for GPIO237<br>Reset type: SYSRSn |
| 12 | GPIO236 | R/W | 1h | Analog mode select for GPIO236<br>Reset type: SYSRSn |
| 11 | GPIO235 | R/W | 1h | Analog mode select for GPIO235<br>Reset type: SYSRSn |
| 10 | GPIO234 | R/W | 1h | Analog mode select for GPIO234<br>Reset type: SYSRSn |
| 9 | GPIO233 | R/W | 1h | Analog mode select for GPIO233<br>Reset type: SYSRSn |
| 8 | GPIO232 | R/W | 1h | Analog mode select for GPIO232<br>Reset type: SYSRSn |
| 7 | GPIO231 | R/W | 1h | Analog mode select for GPIO231<br>Reset type: SYSRSn |
| 6 | GPIO230 | R/W | 1h | Analog mode select for GPIO230<br>Reset type: SYSRSn |
| 5 | GPIO229 | R/W | 1h | Analog mode select for GPIO229<br>Reset type: SYSRSn |
| 4 | GPIO228 | R/W | 1h | Analog mode select for GPIO228<br>Reset type: SYSRSn |
| 3 | GPIO227 | R/W | 1h | Analog mode select for GPIO227<br>Reset type: SYSRSn |
| 2 | GPIO226 | R/W | 1h | Analog mode select for GPIO226<br>Reset type: SYSRSn |
| 1 | GPIO225 | R/W | 1h | Analog mode select for GPIO225<br>Reset type: SYSRSn |
| 0 | GPIO224 | R/W | 1h | Analog mode select for GPIO224<br>Reset type: SYSRSn |

#### 8.7.1.1.41 GPHLOCK Register (Offset = 1FCh) [reset = 0h]

GPHLOCK is shown in Figure 8-44 and described in Table 8-50.

Return to Summary Table.

GPIO H Lock Register (GPIO224 to GPIO255)

Each field in this register locks one IO pin's configuration. This blocks writes to the corresponding bits in the GPHINV register.

0: Pin configuration is unlocked
1: Pin configuration is locked

**Figure 8-44. GPHLOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO247 | GPIO246 | GPIO245 | GPIO244 | GPIO243 | GPIO242 | GPIO241 | GPIO240 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO239 | GPIO238 | GPIO237 | GPIO236 | GPIO235 | GPIO234 | GPIO233 | GPIO232 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO231 | GPIO230 | GPIO229 | GPIO228 | GPIO227 | GPIO226 | GPIO225 | GPIO224 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-50. GPHLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RESERVED | R | 0h | Reserved |
| 30 | RESERVED | R | 0h | Reserved |
| 29 | RESERVED | R | 0h | Reserved |
| 28 | RESERVED | R | 0h | Reserved |
| 27 | RESERVED | R | 0h | Reserved |
| 26 | RESERVED | R | 0h | Reserved |
| 25 | RESERVED | R | 0h | Reserved |
| 24 | RESERVED | R | 0h | Reserved |
| 23 | GPIO247 | R/W | 0h | Configuration lock for GPIO247<br>Reset type: SYSRSn |
| 22 | GPIO246 | R/W | 0h | Configuration lock for GPIO246<br>Reset type: SYSRSn |
| 21 | GPIO245 | R/W | 0h | Configuration lock for GPIO245<br>Reset type: SYSRSn |
| 20 | GPIO244 | R/W | 0h | Configuration lock for GPIO244<br>Reset type: SYSRSn |
| 19 | GPIO243 | R/W | 0h | Configuration lock for GPIO243<br>Reset type: SYSRSn |
| 18 | GPIO242 | R/W | 0h | Configuration lock for GPIO242<br>Reset type: SYSRSn |
| 17 | GPIO241 | R/W | 0h | Configuration lock for GPIO241<br>Reset type: SYSRSn |
| 16 | GPIO240 | R/W | 0h | Configuration lock for GPIO240<br>Reset type: SYSRSn |

**Table 8-50. GPHLOCK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | GPIO239 | R/W | 0h | Configuration lock for GPIO239<br>Reset type: SYSRSn |
| 14 | GPIO238 | R/W | 0h | Configuration lock for GPIO238<br>Reset type: SYSRSn |
| 13 | GPIO237 | R/W | 0h | Configuration lock for GPIO237<br>Reset type: SYSRSn |
| 12 | GPIO236 | R/W | 0h | Configuration lock for GPIO236<br>Reset type: SYSRSn |
| 11 | GPIO235 | R/W | 0h | Configuration lock for GPIO235<br>Reset type: SYSRSn |
| 10 | GPIO234 | R/W | 0h | Configuration lock for GPIO234<br>Reset type: SYSRSn |
| 9 | GPIO233 | R/W | 0h | Configuration lock for GPIO233<br>Reset type: SYSRSn |
| 8 | GPIO232 | R/W | 0h | Configuration lock for GPIO232<br>Reset type: SYSRSn |
| 7 | GPIO231 | R/W | 0h | Configuration lock for GPIO231<br>Reset type: SYSRSn |
| 6 | GPIO230 | R/W | 0h | Configuration lock for GPIO230<br>Reset type: SYSRSn |
| 5 | GPIO229 | R/W | 0h | Configuration lock for GPIO229<br>Reset type: SYSRSn |
| 4 | GPIO228 | R/W | 0h | Configuration lock for GPIO228<br>Reset type: SYSRSn |
| 3 | GPIO227 | R/W | 0h | Configuration lock for GPIO227<br>Reset type: SYSRSn |
| 2 | GPIO226 | R/W | 0h | Configuration lock for GPIO226<br>Reset type: SYSRSn |
| 1 | GPIO225 | R/W | 0h | Configuration lock for GPIO225<br>Reset type: SYSRSn |
| 0 | GPIO224 | R/W | 0h | Configuration lock for GPIO224<br>Reset type: SYSRSn |

#### 8.7.1.1.42 GPHCR Register (Offset = 1FEh) [reset = 0h]

GPHCR is shown in Figure 8-45 and described in Table 8-51.

Return to Summary Table.

GPIO H Lock Commit Register (GPIO224 to GPIO255)

Each field in this register blocks writes to one IO pin's GPHLOCK bit. Once set, a lock commit can only be cleared by a reset.

0: Pin configuration lock is unlocked
1: Pin configuration lock is locked

**Figure 8-45. GPHCR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO247 | GPIO246 | GPIO245 | GPIO244 | GPIO243 | GPIO242 | GPIO241 | GPIO240 |
| 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 |
| GPIO239 | GPIO238 | GPIO237 | GPIO236 | GPIO235 | GPIO234 | GPIO233 | GPIO232 |
| 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 |
| GPIO231 | GPIO230 | GPIO229 | GPIO228 | GPIO227 | GPIO226 | GPIO225 | GPIO224 |
| R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h |

**Table 8-51. GPHCR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RESERVED | R | 0h | Reserved |
| 30 | RESERVED | R | 0h | Reserved |
| 29 | RESERVED | R | 0h | Reserved |
| 28 | RESERVED | R | 0h | Reserved |
| 27 | RESERVED | R | 0h | Reserved |
| 26 | RESERVED | R | 0h | Reserved |
| 25 | RESERVED | R | 0h | Reserved |
| 24 | RESERVED | R | 0h | Reserved |
| 23 | GPIO247 | R/WSOnce | 0h | Configuration lock commit for GPIO247<br>Reset type: SYSRSn |
| 22 | GPIO246 | R/WSOnce | 0h | Configuration lock commit for GPIO246<br>Reset type: SYSRSn |
| 21 | GPIO245 | R/WSOnce | 0h | Configuration lock commit for GPIO245<br>Reset type: SYSRSn |
| 20 | GPIO244 | R/WSOnce | 0h | Configuration lock commit for GPIO244<br>Reset type: SYSRSn |
| 19 | GPIO243 | R/WSOnce | 0h | Configuration lock commit for GPIO243<br>Reset type: SYSRSn |
| 18 | GPIO242 | R/WSOnce | 0h | Configuration lock commit for GPIO242<br>Reset type: SYSRSn |
| 17 | GPIO241 | R/WSOnce | 0h | Configuration lock commit for GPIO241<br>Reset type: SYSRSn |
| 16 | GPIO240 | R/WSOnce | 0h | Configuration lock commit for GPIO240<br>Reset type: SYSRSn |

**Table 8-51. GPHCR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | GPIO239 | R/WSOnce | 0h | Configuration lock commit for GPIO239<br>Reset type: SYSRSn |
| 14 | GPIO238 | R/WSOnce | 0h | Configuration lock commit for GPIO238<br>Reset type: SYSRSn |
| 13 | GPIO237 | R/WSOnce | 0h | Configuration lock commit for GPIO237<br>Reset type: SYSRSn |
| 12 | GPIO236 | R/WSOnce | 0h | Configuration lock commit for GPIO236<br>Reset type: SYSRSn |
| 11 | GPIO235 | R/WSOnce | 0h | Configuration lock commit for GPIO235<br>Reset type: SYSRSn |
| 10 | GPIO234 | R/WSOnce | 0h | Configuration lock commit for GPIO234<br>Reset type: SYSRSn |
| 9 | GPIO233 | R/WSOnce | 0h | Configuration lock commit for GPIO233<br>Reset type: SYSRSn |
| 8 | GPIO232 | R/WSOnce | 0h | Configuration lock commit for GPIO232<br>Reset type: SYSRSn |
| 7 | GPIO231 | R/WSOnce | 0h | Configuration lock commit for GPIO231<br>Reset type: SYSRSn |
| 6 | GPIO230 | R/WSOnce | 0h | Configuration lock commit for GPIO230<br>Reset type: SYSRSn |
| 5 | GPIO229 | R/WSOnce | 0h | Configuration lock commit for GPIO229<br>Reset type: SYSRSn |
| 4 | GPIO228 | R/WSOnce | 0h | Configuration lock commit for GPIO228<br>Reset type: SYSRSn |
| 3 | GPIO227 | R/WSOnce | 0h | Configuration lock commit for GPIO227<br>Reset type: SYSRSn |
| 2 | GPIO226 | R/WSOnce | 0h | Configuration lock commit for GPIO226<br>Reset type: SYSRSn |
| 1 | GPIO225 | R/WSOnce | 0h | Configuration lock commit for GPIO225<br>Reset type: SYSRSn |
| 0 | GPIO224 | R/WSOnce | 0h | Configuration lock commit for GPIO224<br>Reset type: SYSRSn |

### 8.7.1.2 GPIO_DATA_REGS Registers

Table 8-52 lists the memory-mapped registers for the GPIO_DATA_REGS. All register offset addresses not listed in Table 8-52 should be considered as reserved locations and the register contents should not be modified.

**Table 8-52. GPIO_DATA_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | GPADAT | GPIO A Data Register (GPIO0 to GPIO31) | | Go |
| 2h | GPASET | GPIO A Output Set (GPIO0 to GPIO31) | | Go |
| 4h | GPACLEAR | GPIO A Output Clear (GPIO0 to GPIO31) | | Go |
| 6h | GPATOGGLE | GPIO A Output Toggle (GPIO0 to GPIO31) | | Go |
| 8h | GPBDAT | GPIO B Data Register (GPIO32 to GPIO64) | | Go |
| Ah | GPBSET | GPIO B Output Set (GPIO32 to GPIO64) | | Go |
| Ch | GPBCLEAR | GPIO B Output Clear (GPIO32 to GPIO64) | | Go |
| Eh | GPBTOGGLE | GPIO B Output Toggle (GPIO32 to GPIO64) | | Go |
| 38h | GPHDAT | GPIO H Data Register (GPIO0 to GPIO255) | | Go |

Complex bit access types are encoded to fit into small table cells. Table 8-53 shows the codes that are used for access types in this section.

**Table 8-53. 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. |
| y | | When this variable is used in a register name, an offset, or an address it refers to the value of a register array. |

#### 8.7.1.2.1 GPADAT Register (Offset = 0h) [reset = 0h]

GPADAT is shown in Figure 8-46 and described in Table 8-54.

Return to Summary Table.

GPIO A Data (GPIO0 to GPIO31)

Reading a field in this register returns the input level of the corresponding IO pin after qualification and (optionally) inversion. The input value is always readable even when the pin is configured as a GPIO output or peripheral signal.

Writing to a field in this register selects the value of the output data latch for the corresponding IO pin. If the pin is configured as a GPIO output, this value will be driven onto the pin. Otherwise, the value will be latched and ignored unless the pin is reconfigured to be an output. A system reset will clear all output latches.

Due to the difference between the read and write values, sequential read-modify-write operations on this register may corrupt the state of the output latches. The GPASET, GPACLEAR, and GPATOGGLE registers should be used to safely control the output latches.

**Figure 8-46. 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 8-54. GPADAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO31 | R/W | 0h | Read: Input value, Write: Output latch for GPIO31<br>Reset type: SYSRSn |
| 30 | GPIO30 | R/W | 0h | Read: Input value, Write: Output latch for GPIO30<br>Reset type: SYSRSn |
| 29 | GPIO29 | R/W | 0h | Read: Input value, Write: Output latch for GPIO29<br>Reset type: SYSRSn |
| 28 | GPIO28 | R/W | 0h | Read: Input value, Write: Output latch for GPIO28<br>Reset type: SYSRSn |
| 27 | GPIO27 | R/W | 0h | Read: Input value, Write: Output latch for GPIO27<br>Reset type: SYSRSn |
| 26 | GPIO26 | R/W | 0h | Read: Input value, Write: Output latch for GPIO26<br>Reset type: SYSRSn |
| 25 | GPIO25 | R/W | 0h | Read: Input value, Write: Output latch for GPIO25<br>Reset type: SYSRSn |
| 24 | GPIO24 | R/W | 0h | Read: Input value, Write: Output latch for GPIO24<br>Reset type: SYSRSn |
| 23 | GPIO23 | R/W | 0h | Read: Input value, Write: Output latch for GPIO23<br>Reset type: SYSRSn |

**Table 8-54. GPADAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 22 | GPIO22 | R/W | 0h | Read: Input value, Write: Output latch for GPIO22<br>Reset type: SYSRSn |
| 21 | GPIO21 | R/W | 0h | Read: Input value, Write: Output latch for GPIO21<br>Reset type: SYSRSn |
| 20 | GPIO20 | R/W | 0h | Read: Input value, Write: Output latch for GPIO20<br>Reset type: SYSRSn |
| 19 | GPIO19 | R/W | 0h | Read: Input value, Write: Output latch for GPIO19<br>Reset type: SYSRSn |
| 18 | GPIO18 | R/W | 0h | Read: Input value, Write: Output latch for GPIO18<br>Reset type: SYSRSn |
| 17 | GPIO17 | R/W | 0h | Read: Input value, Write: Output latch for GPIO17<br>Reset type: SYSRSn |
| 16 | GPIO16 | R/W | 0h | Read: Input value, Write: Output latch for GPIO16<br>Reset type: SYSRSn |
| 15 | GPIO15 | R/W | 0h | Read: Input value, Write: Output latch for GPIO15<br>Reset type: SYSRSn |
| 14 | GPIO14 | R/W | 0h | Read: Input value, Write: Output latch for GPIO14<br>Reset type: SYSRSn |
| 13 | GPIO13 | R/W | 0h | Read: Input value, Write: Output latch for GPIO13<br>Reset type: SYSRSn |
| 12 | GPIO12 | R/W | 0h | Read: Input value, Write: Output latch for GPIO12<br>Reset type: SYSRSn |
| 11 | GPIO11 | R/W | 0h | Read: Input value, Write: Output latch for GPIO11<br>Reset type: SYSRSn |
| 10 | GPIO10 | R/W | 0h | Read: Input value, Write: Output latch for GPIO10<br>Reset type: SYSRSn |
| 9 | GPIO9 | R/W | 0h | Read: Input value, Write: Output latch for GPIO9<br>Reset type: SYSRSn |
| 8 | GPIO8 | R/W | 0h | Read: Input value, Write: Output latch for GPIO8<br>Reset type: SYSRSn |
| 7 | GPIO7 | R/W | 0h | Read: Input value, Write: Output latch for GPIO7<br>Reset type: SYSRSn |
| 6 | GPIO6 | R/W | 0h | Read: Input value, Write: Output latch for GPIO6<br>Reset type: SYSRSn |
| 5 | GPIO5 | R/W | 0h | Read: Input value, Write: Output latch for GPIO5<br>Reset type: SYSRSn |
| 4 | GPIO4 | R/W | 0h | Read: Input value, Write: Output latch for GPIO4<br>Reset type: SYSRSn |
| 3 | GPIO3 | R/W | 0h | Read: Input value, Write: Output latch for GPIO3<br>Reset type: SYSRSn |
| 2 | GPIO2 | R/W | 0h | Read: Input value, Write: Output latch for GPIO2<br>Reset type: SYSRSn |
| 1 | GPIO1 | R/W | 0h | Read: Input value, Write: Output latch for GPIO1<br>Reset type: SYSRSn |
| 0 | GPIO0 | R/W | 0h | Read: Input value, Write: Output latch for GPIO0<br>Reset type: SYSRSn |

#### 8.7.1.2.2 GPASET Register (Offset = 2h) [reset = 0h]

GPASET is shown in Figure 8-47 and described in Table 8-55.

Return to Summary Table.

GPIO Output Set (GPIO0 to GPIO31)

Writing a 1 to a field in this register sets the output data latch for the corresponding IO pin. Writes of 0 are ignored. Reads of this register always return 0.

**Figure 8-47. 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 8-55. GPASET Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO31 | R/W | 0h | Output set for GPIO31<br>Reset type: SYSRSn |
| 30 | GPIO30 | R/W | 0h | Output set for GPIO30<br>Reset type: SYSRSn |
| 29 | GPIO29 | R/W | 0h | Output set for GPIO29<br>Reset type: SYSRSn |
| 28 | GPIO28 | R/W | 0h | Output set for GPIO28<br>Reset type: SYSRSn |
| 27 | GPIO27 | R/W | 0h | Output set for GPIO27<br>Reset type: SYSRSn |
| 26 | GPIO26 | R/W | 0h | Output set for GPIO26<br>Reset type: SYSRSn |
| 25 | GPIO25 | R/W | 0h | Output set for GPIO25<br>Reset type: SYSRSn |
| 24 | GPIO24 | R/W | 0h | Output set for GPIO24<br>Reset type: SYSRSn |
| 23 | GPIO23 | R/W | 0h | Output set for GPIO23<br>Reset type: SYSRSn |
| 22 | GPIO22 | R/W | 0h | Output set for GPIO22<br>Reset type: SYSRSn |
| 21 | GPIO21 | R/W | 0h | Output set for GPIO21<br>Reset type: SYSRSn |
| 20 | GPIO20 | R/W | 0h | Output set for GPIO20<br>Reset type: SYSRSn |
| 19 | GPIO19 | R/W | 0h | Output set for GPIO19<br>Reset type: SYSRSn |

**Table 8-55. GPASET Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18 | GPIO18 | R/W | 0h | Output set for GPIO18<br>Reset type: SYSRSn |
| 17 | GPIO17 | R/W | 0h | Output set for GPIO17<br>Reset type: SYSRSn |
| 16 | GPIO16 | R/W | 0h | Output set for GPIO16<br>Reset type: SYSRSn |
| 15 | GPIO15 | R/W | 0h | Output set for GPIO15<br>Reset type: SYSRSn |
| 14 | GPIO14 | R/W | 0h | Output set for GPIO14<br>Reset type: SYSRSn |
| 13 | GPIO13 | R/W | 0h | Output set for GPIO13<br>Reset type: SYSRSn |
| 12 | GPIO12 | R/W | 0h | Output set for GPIO12<br>Reset type: SYSRSn |
| 11 | GPIO11 | R/W | 0h | Output set for GPIO11<br>Reset type: SYSRSn |
| 10 | GPIO10 | R/W | 0h | Output set for GPIO10<br>Reset type: SYSRSn |
| 9 | GPIO9 | R/W | 0h | Output set for GPIO9<br>Reset type: SYSRSn |
| 8 | GPIO8 | R/W | 0h | Output set for GPIO8<br>Reset type: SYSRSn |
| 7 | GPIO7 | R/W | 0h | Output set for GPIO7<br>Reset type: SYSRSn |
| 6 | GPIO6 | R/W | 0h | Output set for GPIO6<br>Reset type: SYSRSn |
| 5 | GPIO5 | R/W | 0h | Output set for GPIO5<br>Reset type: SYSRSn |
| 4 | GPIO4 | R/W | 0h | Output set for GPIO4<br>Reset type: SYSRSn |
| 3 | GPIO3 | R/W | 0h | Output set for GPIO3<br>Reset type: SYSRSn |
| 2 | GPIO2 | R/W | 0h | Output set for GPIO2<br>Reset type: SYSRSn |
| 1 | GPIO1 | R/W | 0h | Output set for GPIO1<br>Reset type: SYSRSn |
| 0 | GPIO0 | R/W | 0h | Output set for GPIO0<br>Reset type: SYSRSn |

#### 8.7.1.2.3 GPACLEAR Register (Offset = 4h) [reset = 0h]

GPACLEAR is shown in Figure 8-48 and described in Table 8-56.

Return to Summary Table.

GPIO Output Clear (GPIO0 to GPIO31)

Writing a 1 to a field in this register clears the output data latch for the corresponding IO pin. Writes of 0 are ignored. Reads of this register always return 0.

**Figure 8-48. 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 8-56. GPACLEAR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO31 | R/W | 0h | Output clear for GPIO31<br>Reset type: SYSRSn |
| 30 | GPIO30 | R/W | 0h | Output clear for GPIO30<br>Reset type: SYSRSn |
| 29 | GPIO29 | R/W | 0h | Output clear for GPIO29<br>Reset type: SYSRSn |
| 28 | GPIO28 | R/W | 0h | Output clear for GPIO28<br>Reset type: SYSRSn |
| 27 | GPIO27 | R/W | 0h | Output clear for GPIO27<br>Reset type: SYSRSn |
| 26 | GPIO26 | R/W | 0h | Output clear for GPIO26<br>Reset type: SYSRSn |
| 25 | GPIO25 | R/W | 0h | Output clear for GPIO25<br>Reset type: SYSRSn |
| 24 | GPIO24 | R/W | 0h | Output clear for GPIO24<br>Reset type: SYSRSn |
| 23 | GPIO23 | R/W | 0h | Output clear for GPIO23<br>Reset type: SYSRSn |
| 22 | GPIO22 | R/W | 0h | Output clear for GPIO22<br>Reset type: SYSRSn |
| 21 | GPIO21 | R/W | 0h | Output clear for GPIO21<br>Reset type: SYSRSn |
| 20 | GPIO20 | R/W | 0h | Output clear for GPIO20<br>Reset type: SYSRSn |
| 19 | GPIO19 | R/W | 0h | Output clear for GPIO19<br>Reset type: SYSRSn |

**Table 8-56. GPACLEAR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18 | GPIO18 | R/W | 0h | Output clear for GPIO18<br>Reset type: SYSRSn |
| 17 | GPIO17 | R/W | 0h | Output clear for GPIO17<br>Reset type: SYSRSn |
| 16 | GPIO16 | R/W | 0h | Output clear for GPIO16<br>Reset type: SYSRSn |
| 15 | GPIO15 | R/W | 0h | Output clear for GPIO15<br>Reset type: SYSRSn |
| 14 | GPIO14 | R/W | 0h | Output clear for GPIO14<br>Reset type: SYSRSn |
| 13 | GPIO13 | R/W | 0h | Output clear for GPIO13<br>Reset type: SYSRSn |
| 12 | GPIO12 | R/W | 0h | Output clear for GPIO12<br>Reset type: SYSRSn |
| 11 | GPIO11 | R/W | 0h | Output clear for GPIO11<br>Reset type: SYSRSn |
| 10 | GPIO10 | R/W | 0h | Output clear for GPIO10<br>Reset type: SYSRSn |
| 9 | GPIO9 | R/W | 0h | Output clear for GPIO9<br>Reset type: SYSRSn |
| 8 | GPIO8 | R/W | 0h | Output clear for GPIO8<br>Reset type: SYSRSn |
| 7 | GPIO7 | R/W | 0h | Output clear for GPIO7<br>Reset type: SYSRSn |
| 6 | GPIO6 | R/W | 0h | Output clear for GPIO6<br>Reset type: SYSRSn |
| 5 | GPIO5 | R/W | 0h | Output clear for GPIO5<br>Reset type: SYSRSn |
| 4 | GPIO4 | R/W | 0h | Output clear for GPIO4<br>Reset type: SYSRSn |
| 3 | GPIO3 | R/W | 0h | Output clear for GPIO3<br>Reset type: SYSRSn |
| 2 | GPIO2 | R/W | 0h | Output clear for GPIO2<br>Reset type: SYSRSn |
| 1 | GPIO1 | R/W | 0h | Output clear for GPIO1<br>Reset type: SYSRSn |
| 0 | GPIO0 | R/W | 0h | Output clear for GPIO0<br>Reset type: SYSRSn |

#### 8.7.1.2.4 GPATOGGLE Register (Offset = 6h) [reset = 0h]

GPATOGGLE is shown in Figure 8-49 and described in Table 8-57.

Return to Summary Table.

GPIO Output Toggle (GPIO0 to GPIO31)

Writing a 1 to a field in this register inverts the value of the output data latch for the corresponding IO pin. Writes of 0 are ignored. Reads of this register always return 0.

**Figure 8-49. 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 8-57. GPATOGGLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | GPIO31 | R/W | 0h | Output toggle for GPIO31<br>Reset type: SYSRSn |
| 30 | GPIO30 | R/W | 0h | Output toggle for GPIO30<br>Reset type: SYSRSn |
| 29 | GPIO29 | R/W | 0h | Output toggle for GPIO29<br>Reset type: SYSRSn |
| 28 | GPIO28 | R/W | 0h | Output toggle for GPIO28<br>Reset type: SYSRSn |
| 27 | GPIO27 | R/W | 0h | Output toggle for GPIO27<br>Reset type: SYSRSn |
| 26 | GPIO26 | R/W | 0h | Output toggle for GPIO26<br>Reset type: SYSRSn |
| 25 | GPIO25 | R/W | 0h | Output toggle for GPIO25<br>Reset type: SYSRSn |
| 24 | GPIO24 | R/W | 0h | Output toggle for GPIO24<br>Reset type: SYSRSn |
| 23 | GPIO23 | R/W | 0h | Output toggle for GPIO23<br>Reset type: SYSRSn |
| 22 | GPIO22 | R/W | 0h | Output toggle for GPIO22<br>Reset type: SYSRSn |
| 21 | GPIO21 | R/W | 0h | Output toggle for GPIO21<br>Reset type: SYSRSn |
| 20 | GPIO20 | R/W | 0h | Output toggle for GPIO20<br>Reset type: SYSRSn |
| 19 | GPIO19 | R/W | 0h | Output toggle for GPIO19<br>Reset type: SYSRSn |

**Table 8-57. GPATOGGLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18 | GPIO18 | R/W | 0h | Output toggle for GPIO18<br>Reset type: SYSRSn |
| 17 | GPIO17 | R/W | 0h | Output toggle for GPIO17<br>Reset type: SYSRSn |
| 16 | GPIO16 | R/W | 0h | Output toggle for GPIO16<br>Reset type: SYSRSn |
| 15 | GPIO15 | R/W | 0h | Output toggle for GPIO15<br>Reset type: SYSRSn |
| 14 | GPIO14 | R/W | 0h | Output toggle for GPIO14<br>Reset type: SYSRSn |
| 13 | GPIO13 | R/W | 0h | Output toggle for GPIO13<br>Reset type: SYSRSn |
| 12 | GPIO12 | R/W | 0h | Output toggle for GPIO12<br>Reset type: SYSRSn |
| 11 | GPIO11 | R/W | 0h | Output toggle for GPIO11<br>Reset type: SYSRSn |
| 10 | GPIO10 | R/W | 0h | Output toggle for GPIO10<br>Reset type: SYSRSn |
| 9 | GPIO9 | R/W | 0h | Output toggle for GPIO9<br>Reset type: SYSRSn |
| 8 | GPIO8 | R/W | 0h | Output toggle for GPIO8<br>Reset type: SYSRSn |
| 7 | GPIO7 | R/W | 0h | Output toggle for GPIO7<br>Reset type: SYSRSn |
| 6 | GPIO6 | R/W | 0h | Output toggle for GPIO6<br>Reset type: SYSRSn |
| 5 | GPIO5 | R/W | 0h | Output toggle for GPIO5<br>Reset type: SYSRSn |
| 4 | GPIO4 | R/W | 0h | Output toggle for GPIO4<br>Reset type: SYSRSn |
| 3 | GPIO3 | R/W | 0h | Output toggle for GPIO3<br>Reset type: SYSRSn |
| 2 | GPIO2 | R/W | 0h | Output toggle for GPIO2<br>Reset type: SYSRSn |
| 1 | GPIO1 | R/W | 0h | Output toggle for GPIO1<br>Reset type: SYSRSn |
| 0 | GPIO0 | R/W | 0h | Output toggle for GPIO0<br>Reset type: SYSRSn |

#### 8.7.1.2.5 GPBDAT Register (Offset = 8h) [reset = 0h]

GPBDAT is shown in Figure 8-50 and described in Table 8-58.

Return to Summary Table.

GPIO B Data (GPIO32 to GPIO64)

Reading a field in this register returns the input level of the corresponding IO pin after qualification and (optionally) inversion. The input value is always readable even when the pin is configured as a GPIO output or peripheral signal.

Writing to a field in this register selects the value of the output data latch for the corresponding IO pin. If the pin is configured as a GPIO output, this value will be driven onto the pin. Otherwise, the value will be latched and ignored unless the pin is reconfigured to be an output. A system reset will clear all output latches.

Due to the difference between the read and write values, sequential read-modify-write operations on this register may corrupt the state of the output latches. The GPBSET, GPBCLEAR, and GPBTOGGLE registers should be used to safely control the output latches.

**Figure 8-50. GPBDAT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | GPIO59 | GPIO58 | GPIO57 | GPIO56 |
| R-0h | R-0h | R-0h | R-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 | RESERVED | GPIO37 | RESERVED | GPIO35 | GPIO34 | GPIO33 | GPIO32 |
| R/W-0h | R-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-58. GPBDAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RESERVED | R | 0h | Reserved |
| 30 | RESERVED | R | 0h | Reserved |
| 29 | RESERVED | R | 0h | Reserved |
| 28 | RESERVED | R | 0h | Reserved |
| 27 | GPIO59 | R/W | 0h | Read: Input value, Write: Output latch for GPIO59<br>Reset type: SYSRSn |
| 26 | GPIO58 | R/W | 0h | Read: Input value, Write: Output latch for GPIO58<br>Reset type: SYSRSn |
| 25 | GPIO57 | R/W | 0h | Read: Input value, Write: Output latch for GPIO57<br>Reset type: SYSRSn |
| 24 | GPIO56 | R/W | 0h | Read: Input value, Write: Output latch for GPIO56<br>Reset type: SYSRSn |
| 23 | GPIO55 | R/W | 0h | Read: Input value, Write: Output latch for GPIO55<br>Reset type: SYSRSn |
| 22 | GPIO54 | R/W | 0h | Read: Input value, Write: Output latch for GPIO54<br>Reset type: SYSRSn |
| 21 | GPIO53 | R/W | 0h | Read: Input value, Write: Output latch for GPIO53<br>Reset type: SYSRSn |

**Table 8-58. GPBDAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 20 | GPIO52 | R/W | 0h | Read: Input value, Write: Output latch for GPIO52<br>Reset type: SYSRSn |
| 19 | GPIO51 | R/W | 0h | Read: Input value, Write: Output latch for GPIO51<br>Reset type: SYSRSn |
| 18 | GPIO50 | R/W | 0h | Read: Input value, Write: Output latch for GPIO50<br>Reset type: SYSRSn |
| 17 | GPIO49 | R/W | 0h | Read: Input value, Write: Output latch for GPIO49<br>Reset type: SYSRSn |
| 16 | GPIO48 | R/W | 0h | Read: Input value, Write: Output latch for GPIO48<br>Reset type: SYSRSn |
| 15 | GPIO47 | R/W | 0h | Read: Input value, Write: Output latch for GPIO47<br>Reset type: SYSRSn |
| 14 | GPIO46 | R/W | 0h | Read: Input value, Write: Output latch for GPIO46<br>Reset type: SYSRSn |
| 13 | GPIO45 | R/W | 0h | Read: Input value, Write: Output latch for GPIO45<br>Reset type: SYSRSn |
| 12 | GPIO44 | R/W | 0h | Read: Input value, Write: Output latch for GPIO44<br>Reset type: SYSRSn |
| 11 | GPIO43 | R/W | 0h | Read: Input value, Write: Output latch for GPIO43<br>Reset type: SYSRSn |
| 10 | GPIO42 | R/W | 0h | Read: Input value, Write: Output latch for GPIO42<br>Reset type: SYSRSn |
| 9 | GPIO41 | R/W | 0h | Read: Input value, Write: Output latch for GPIO41<br>Reset type: SYSRSn |
| 8 | GPIO40 | R/W | 0h | Read: Input value, Write: Output latch for GPIO40<br>Reset type: SYSRSn |
| 7 | GPIO39 | R/W | 0h | Read: Input value, Write: Output latch for GPIO39<br>Reset type: SYSRSn |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | GPIO37 | R/W | 0h | Read: Input value, Write: Output latch for GPIO37<br>Reset type: SYSRSn |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | GPIO35 | R/W | 0h | Read: Input value, Write: Output latch for GPIO35<br>Reset type: SYSRSn |
| 2 | GPIO34 | R/W | 0h | Read: Input value, Write: Output latch for GPIO34<br>Reset type: SYSRSn |
| 1 | GPIO33 | R/W | 0h | Read: Input value, Write: Output latch for GPIO33<br>Reset type: SYSRSn |
| 0 | GPIO32 | R/W | 0h | Read: Input value, Write: Output latch for GPIO32<br>Reset type: SYSRSn |

#### 8.7.1.2.6 GPBSET Register (Offset = Ah) [reset = 0h]

GPBSET is shown in Figure 8-51 and described in Table 8-59.

Return to Summary Table.

GPIO Output Set (GPIO32 to GPIO64)

Writing a 1 to a field in this register sets the output data latch for the corresponding IO pin. Writes of 0 are ignored. Reads of this register always return 0.

**Figure 8-51. GPBSET Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | GPIO59 | GPIO58 | GPIO57 | GPIO56 |
| R-0h | R-0h | R-0h | R-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 | RESERVED | GPIO37 | RESERVED | GPIO35 | GPIO34 | GPIO33 | GPIO32 |
| R/W-0h | R-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-59. GPBSET Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RESERVED | R | 0h | Reserved |
| 30 | RESERVED | R | 0h | Reserved |
| 29 | RESERVED | R | 0h | Reserved |
| 28 | RESERVED | R | 0h | Reserved |
| 27 | GPIO59 | R/W | 0h | Output set for GPIO59<br>Reset type: SYSRSn |
| 26 | GPIO58 | R/W | 0h | Output set for GPIO58<br>Reset type: SYSRSn |
| 25 | GPIO57 | R/W | 0h | Output set for GPIO57<br>Reset type: SYSRSn |
| 24 | GPIO56 | R/W | 0h | Output set for GPIO56<br>Reset type: SYSRSn |
| 23 | GPIO55 | R/W | 0h | Output set for GPIO55<br>Reset type: SYSRSn |
| 22 | GPIO54 | R/W | 0h | Output set for GPIO54<br>Reset type: SYSRSn |
| 21 | GPIO53 | R/W | 0h | Output set for GPIO53<br>Reset type: SYSRSn |
| 20 | GPIO52 | R/W | 0h | Output set for GPIO52<br>Reset type: SYSRSn |
| 19 | GPIO51 | R/W | 0h | Output set for GPIO51<br>Reset type: SYSRSn |
| 18 | GPIO50 | R/W | 0h | Output set for GPIO50<br>Reset type: SYSRSn |
| 17 | GPIO49 | R/W | 0h | Output set for GPIO49<br>Reset type: SYSRSn |

**Table 8-59. GPBSET Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 16 | GPIO48 | R/W | 0h | Output set for GPIO48<br>Reset type: SYSRSn |
| 15 | GPIO47 | R/W | 0h | Output set for GPIO47<br>Reset type: SYSRSn |
| 14 | GPIO46 | R/W | 0h | Output set for GPIO46<br>Reset type: SYSRSn |
| 13 | GPIO45 | R/W | 0h | Output set for GPIO45<br>Reset type: SYSRSn |
| 12 | GPIO44 | R/W | 0h | Output set for GPIO44<br>Reset type: SYSRSn |
| 11 | GPIO43 | R/W | 0h | Output set for GPIO43<br>Reset type: SYSRSn |
| 10 | GPIO42 | R/W | 0h | Output set for GPIO42<br>Reset type: SYSRSn |
| 9 | GPIO41 | R/W | 0h | Output set for GPIO41<br>Reset type: SYSRSn |
| 8 | GPIO40 | R/W | 0h | Output set for GPIO40<br>Reset type: SYSRSn |
| 7 | GPIO39 | R/W | 0h | Output set for GPIO39<br>Reset type: SYSRSn |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | GPIO37 | R/W | 0h | Output set for GPIO37<br>Reset type: SYSRSn |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | GPIO35 | R/W | 0h | Output set for GPIO35<br>Reset type: SYSRSn |
| 2 | GPIO34 | R/W | 0h | Output set for GPIO34<br>Reset type: SYSRSn |
| 1 | GPIO33 | R/W | 0h | Output set for GPIO33<br>Reset type: SYSRSn |
| 0 | GPIO32 | R/W | 0h | Output set for GPIO32<br>Reset type: SYSRSn |

#### 8.7.1.2.7 GPBCLEAR Register (Offset = Ch) [reset = 0h]

GPBCLEAR is shown in Figure 8-52 and described in Table 8-60.

Return to Summary Table.

GPIO Output Clear (GPIO32 to GPIO64)

Writing a 1 to a field in this register clears the output data latch for the corresponding IO pin. Writes of 0 are ignored. Reads of this register always return 0.

**Figure 8-52. GPBCLEAR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | GPIO59 | GPIO58 | GPIO57 | GPIO56 |
| R-0h | R-0h | R-0h | R-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 | RESERVED | GPIO37 | RESERVED | GPIO35 | GPIO34 | GPIO33 | GPIO32 |
| R/W-0h | R-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-60. GPBCLEAR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RESERVED | R | 0h | Reserved |
| 30 | RESERVED | R | 0h | Reserved |
| 29 | RESERVED | R | 0h | Reserved |
| 28 | RESERVED | R | 0h | Reserved |
| 27 | GPIO59 | R/W | 0h | Output clear for GPIO59<br>Reset type: SYSRSn |
| 26 | GPIO58 | R/W | 0h | Output clear for GPIO58<br>Reset type: SYSRSn |
| 25 | GPIO57 | R/W | 0h | Output clear for GPIO57<br>Reset type: SYSRSn |
| 24 | GPIO56 | R/W | 0h | Output clear for GPIO56<br>Reset type: SYSRSn |
| 23 | GPIO55 | R/W | 0h | Output clear for GPIO55<br>Reset type: SYSRSn |
| 22 | GPIO54 | R/W | 0h | Output clear for GPIO54<br>Reset type: SYSRSn |
| 21 | GPIO53 | R/W | 0h | Output clear for GPIO53<br>Reset type: SYSRSn |
| 20 | GPIO52 | R/W | 0h | Output clear for GPIO52<br>Reset type: SYSRSn |
| 19 | GPIO51 | R/W | 0h | Output clear for GPIO51<br>Reset type: SYSRSn |
| 18 | GPIO50 | R/W | 0h | Output clear for GPIO50<br>Reset type: SYSRSn |
| 17 | GPIO49 | R/W | 0h | Output clear for GPIO49<br>Reset type: SYSRSn |

**Table 8-60. GPBCLEAR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 16 | GPIO48 | R/W | 0h | Output clear for GPIO48<br>Reset type: SYSRSn |
| 15 | GPIO47 | R/W | 0h | Output clear for GPIO47<br>Reset type: SYSRSn |
| 14 | GPIO46 | R/W | 0h | Output clear for GPIO46<br>Reset type: SYSRSn |
| 13 | GPIO45 | R/W | 0h | Output clear for GPIO45<br>Reset type: SYSRSn |
| 12 | GPIO44 | R/W | 0h | Output clear for GPIO44<br>Reset type: SYSRSn |
| 11 | GPIO43 | R/W | 0h | Output clear for GPIO43<br>Reset type: SYSRSn |
| 10 | GPIO42 | R/W | 0h | Output clear for GPIO42<br>Reset type: SYSRSn |
| 9 | GPIO41 | R/W | 0h | Output clear for GPIO41<br>Reset type: SYSRSn |
| 8 | GPIO40 | R/W | 0h | Output clear for GPIO40<br>Reset type: SYSRSn |
| 7 | GPIO39 | R/W | 0h | Output clear for GPIO39<br>Reset type: SYSRSn |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | GPIO37 | R/W | 0h | Output clear for GPIO37<br>Reset type: SYSRSn |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | GPIO35 | R/W | 0h | Output clear for GPIO35<br>Reset type: SYSRSn |
| 2 | GPIO34 | R/W | 0h | Output clear for GPIO34<br>Reset type: SYSRSn |
| 1 | GPIO33 | R/W | 0h | Output clear for GPIO33<br>Reset type: SYSRSn |
| 0 | GPIO32 | R/W | 0h | Output clear for GPIO32<br>Reset type: SYSRSn |

#### 8.7.1.2.8 GPBTOGGLE Register (Offset = Eh) [reset = 0h]

GPBTOGGLE is shown in Figure 8-53 and described in Table 8-61.

Return to Summary Table.

GPIO Output Toggle (GPIO32 to GPIO64)

Writing a 1 to a field in this register inverts the value of the output data latch for the corresponding IO pin. Writes of 0 are ignored. Reads of this register always return 0.

**Figure 8-53. GPBTOGGLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | GPIO59 | GPIO58 | GPIO57 | GPIO56 |
| R-0h | R-0h | R-0h | R-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 | RESERVED | GPIO37 | RESERVED | GPIO35 | GPIO34 | GPIO33 | GPIO32 |
| R/W-0h | R-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-61. GPBTOGGLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RESERVED | R | 0h | Reserved |
| 30 | RESERVED | R | 0h | Reserved |
| 29 | RESERVED | R | 0h | Reserved |
| 28 | RESERVED | R | 0h | Reserved |
| 27 | GPIO59 | R/W | 0h | Output toggle for GPIO59<br>Reset type: SYSRSn |
| 26 | GPIO58 | R/W | 0h | Output toggle for GPIO58<br>Reset type: SYSRSn |
| 25 | GPIO57 | R/W | 0h | Output toggle for GPIO57<br>Reset type: SYSRSn |
| 24 | GPIO56 | R/W | 0h | Output toggle for GPIO56<br>Reset type: SYSRSn |
| 23 | GPIO55 | R/W | 0h | Output toggle for GPIO55<br>Reset type: SYSRSn |
| 22 | GPIO54 | R/W | 0h | Output toggle for GPIO54<br>Reset type: SYSRSn |
| 21 | GPIO53 | R/W | 0h | Output toggle for GPIO53<br>Reset type: SYSRSn |
| 20 | GPIO52 | R/W | 0h | Output toggle for GPIO52<br>Reset type: SYSRSn |
| 19 | GPIO51 | R/W | 0h | Output toggle for GPIO51<br>Reset type: SYSRSn |
| 18 | GPIO50 | R/W | 0h | Output toggle for GPIO50<br>Reset type: SYSRSn |
| 17 | GPIO49 | R/W | 0h | Output toggle for GPIO49<br>Reset type: SYSRSn |

**Table 8-61. GPBTOGGLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 16 | GPIO48 | R/W | 0h | Output toggle for GPIO48<br>Reset type: SYSRSn |
| 15 | GPIO47 | R/W | 0h | Output toggle for GPIO47<br>Reset type: SYSRSn |
| 14 | GPIO46 | R/W | 0h | Output toggle for GPIO46<br>Reset type: SYSRSn |
| 13 | GPIO45 | R/W | 0h | Output toggle for GPIO45<br>Reset type: SYSRSn |
| 12 | GPIO44 | R/W | 0h | Output toggle for GPIO44<br>Reset type: SYSRSn |
| 11 | GPIO43 | R/W | 0h | Output toggle for GPIO43<br>Reset type: SYSRSn |
| 10 | GPIO42 | R/W | 0h | Output toggle for GPIO42<br>Reset type: SYSRSn |
| 9 | GPIO41 | R/W | 0h | Output toggle for GPIO41<br>Reset type: SYSRSn |
| 8 | GPIO40 | R/W | 0h | Output toggle for GPIO40<br>Reset type: SYSRSn |
| 7 | GPIO39 | R/W | 0h | Output toggle for GPIO39<br>Reset type: SYSRSn |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | GPIO37 | R/W | 0h | Output toggle for GPIO37<br>Reset type: SYSRSn |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | GPIO35 | R/W | 0h | Output toggle for GPIO35<br>Reset type: SYSRSn |
| 2 | GPIO34 | R/W | 0h | Output toggle for GPIO34<br>Reset type: SYSRSn |
| 1 | GPIO33 | R/W | 0h | Output toggle for GPIO33<br>Reset type: SYSRSn |
| 0 | GPIO32 | R/W | 0h | Output toggle for GPIO32<br>Reset type: SYSRSn |

#### 8.7.1.2.9 GPHDAT Register (Offset = 38h) [reset = 0h]

GPHDAT is shown in Figure 8-54 and described in Table 8-62.

Return to Summary Table.

GPIO H Data (GPIO224 to GPIO255)

Reading a field in this register returns the input level of the corresponding IO pin after qualification and (optionally) inversion. In digital mode, these pins only support input functionality.

**Figure 8-54. GPHDAT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| GPIO247 | GPIO246 | GPIO245 | GPIO244 | GPIO243 | GPIO242 | GPIO241 | GPIO240 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| GPIO239 | GPIO238 | GPIO237 | GPIO236 | GPIO235 | GPIO234 | GPIO233 | GPIO232 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GPIO231 | GPIO230 | GPIO229 | GPIO228 | GPIO227 | GPIO226 | GPIO225 | GPIO224 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 8-62. GPHDAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RESERVED | R | 0h | Reserved |
| 30 | RESERVED | R | 0h | Reserved |
| 29 | RESERVED | R | 0h | Reserved |
| 28 | RESERVED | R | 0h | Reserved |
| 27 | RESERVED | R | 0h | Reserved |
| 26 | RESERVED | R | 0h | Reserved |
| 25 | RESERVED | R | 0h | Reserved |
| 24 | RESERVED | R | 0h | Reserved |
| 23 | GPIO247 | R/W | 0h | Read: Input value for GPIO247<br>Reset type: SYSRSn |
| 22 | GPIO246 | R/W | 0h | Read: Input value for GPIO246<br>Reset type: SYSRSn |
| 21 | GPIO245 | R/W | 0h | Read: Input value for GPIO245<br>Reset type: SYSRSn |
| 20 | GPIO244 | R/W | 0h | Read: Input value for GPIO244<br>Reset type: SYSRSn |
| 19 | GPIO243 | R/W | 0h | Read: Input value for GPIO243<br>Reset type: SYSRSn |
| 18 | GPIO242 | R/W | 0h | Read: Input value for GPIO242<br>Reset type: SYSRSn |
| 17 | GPIO241 | R/W | 0h | Read: Input value for GPIO241<br>Reset type: SYSRSn |
| 16 | GPIO240 | R/W | 0h | Read: Input value for GPIO240<br>Reset type: SYSRSn |
| 15 | GPIO239 | R/W | 0h | Read: Input value for GPIO239<br>Reset type: SYSRSn |

**Table 8-62. GPHDAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 14 | GPIO238 | R/W | 0h | Read: Input value for GPIO238<br>Reset type: SYSRSn |
| 13 | GPIO237 | R/W | 0h | Read: Input value for GPIO237<br>Reset type: SYSRSn |
| 12 | GPIO236 | R/W | 0h | Read: Input value for GPIO236<br>Reset type: SYSRSn |
| 11 | GPIO235 | R/W | 0h | Read: Input value for GPIO235<br>Reset type: SYSRSn |
| 10 | GPIO234 | R/W | 0h | Read: Input value for GPIO234<br>Reset type: SYSRSn |
| 9 | GPIO233 | R/W | 0h | Read: Input value for GPIO233<br>Reset type: SYSRSn |
| 8 | GPIO232 | R/W | 0h | Read: Input value for GPIO232<br>Reset type: SYSRSn |
| 7 | GPIO231 | R/W | 0h | Read: Input value for GPIO231<br>Reset type: SYSRSn |
| 6 | GPIO230 | R/W | 0h | Read: Input value for GPIO230<br>Reset type: SYSRSn |
| 5 | GPIO229 | R/W | 0h | Read: Input value for GPIO229<br>Reset type: SYSRSn |
| 4 | GPIO228 | R/W | 0h | Read: Input value for GPIO228<br>Reset type: SYSRSn |
| 3 | GPIO227 | R/W | 0h | Read: Input value for GPIO227<br>Reset type: SYSRSn |
| 2 | GPIO226 | R/W | 0h | Read: Input value for GPIO226<br>Reset type: SYSRSn |
| 1 | GPIO225 | R/W | 0h | Read: Input value for GPIO225<br>Reset type: SYSRSn |
| 0 | GPIO224 | R/W | 0h | Read: Input value for GPIO224<br>Reset type: SYSRSn |

# Crossbar (X-BAR)

The crossbars (referred to as X-BAR throughout this document) provide flexibility to connect device inputs, outputs, and internal resources in a variety of configurations. The device contains a total of three X-BARs: the Input X-BAR, the Output X-BAR, and the ePWM X-BAR. Each of the X-BARs is named according to where they take signals. For example, the Input X-BAR brings external signals “in” to the device. The Output X-BAR takes internal signals “out” of the device to a GPIO. The ePWM X-BAR takes signals and brings them to the ePWM modules. You can read more about each of these X-BARs in the following sections.

**Topic** **Page**

## 9.1 GPIO Input X-BAR

On this device, the Input X-BAR is used to route signals from a GPIO to many different IP blocks such as the ADC(s), eCAP(s), ePWM(s), and external interrupts. The Input X-BAR has access to every GPIO and can route each signal to any (or multiple) of the IP blocks previously mentioned. The digital input of AIOs are also available on the Input X-BAR. This flexibility relieves some of the constraints on peripheral muxing by just requiring any GPIO pin to be available. It is important to note that the function selected on the GPIO mux does not affect the Input X-BAR. The Input X-BAR simply connects the signal on the input buffer to the selected destination. Therefore, you can do things such as route the output of one peripheral to another (that is, measure the output of an ePWM with an eCAP for a frequency test).

The Input X-BAR is configured via the INPUTxSELECT registers. The available IP destination(s) for each INPUTx is shown in Figure 9-1. For more information on configuration, see the INPUT_XBAR_REGS register definitions at the end of this chapter.

**Figure 9-1. Input X-BAR**

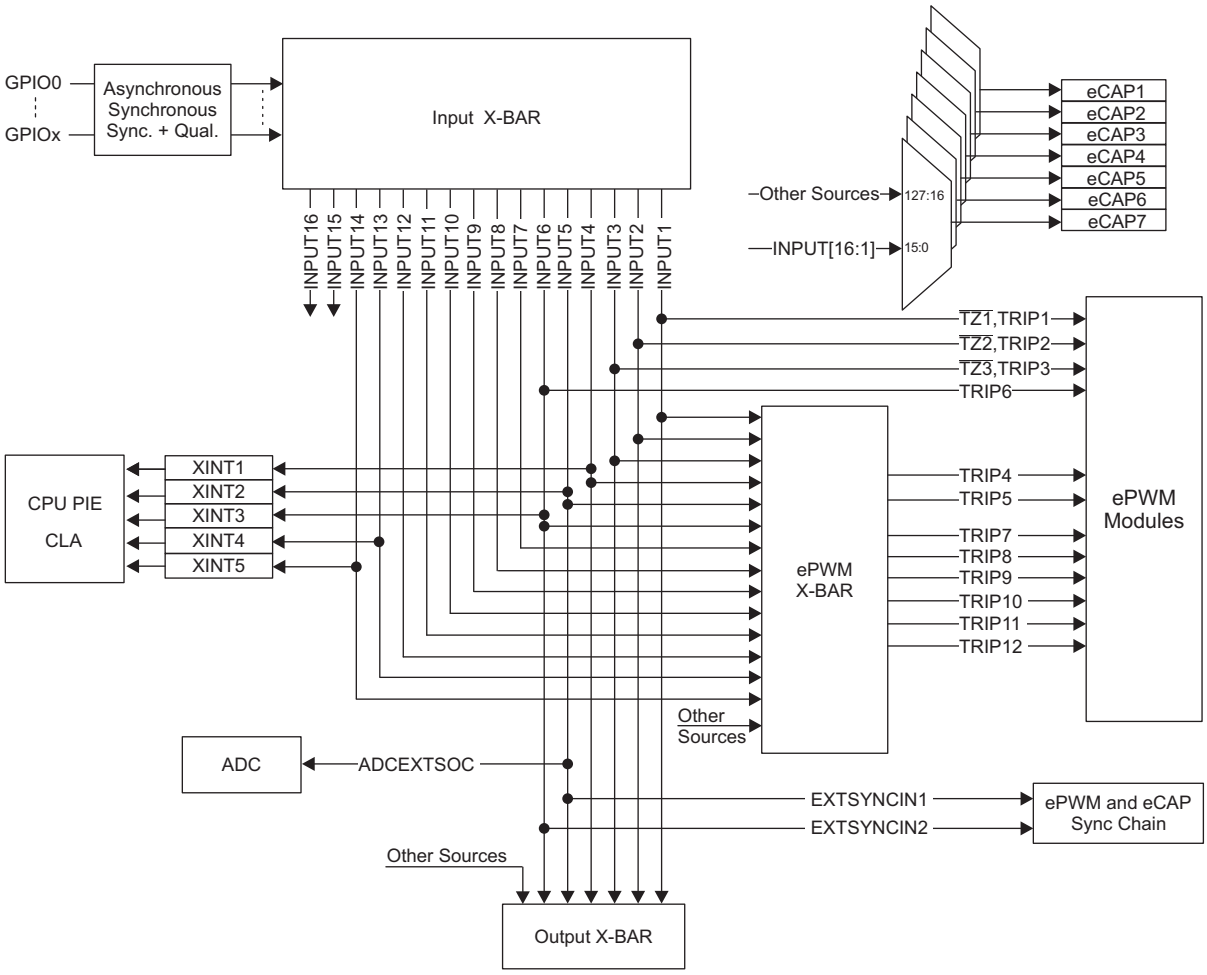


**Table 9-1. Input X-BAR Destinations**

| INPUT | DESTINATIONS |
|---|---|
| INPUT1 | eCAPx, ePWM X-BAR, ePWM[TZ1,TRIP1], Output X-BAR |
| INPUT2 | eCAPx, ePWM X-BAR, ePWM[TZ2,TRIP2], Output X-BAR |
| INPUT3 | eCAPx, ePWM X-BAR, ePWM[TZ3,TRIP3], Output X-BAR |
| INPUT4 | eCAPx, ePWM X-BAR, XINT1, Output X-BAR |
| INPUT5 | eCAPx, ePWM X-BAR, XINT2, ADCEXTSOC, EXTSYNCIN1, Output X-BAR |
| INPUT6 | eCAPx, ePWM X-BAR, XINT3, ePWM[TRIP6], EXTSYNCIN2, Output X-BAR |
| INPUT7 | eCAPx, ePWM X-BAR |
| INPUT8 | eCAPx, ePWM X-BAR |
| INPUT9 | eCAPx, ePWM X-BAR |
| INPUT10 | eCAPx, ePWM X-BAR |
| INPUT11 | eCAPx, ePWM X-BAR |
| INPUT12 | eCAPx, ePWM X-BAR |
| INPUT13 | eCAPx, ePWM X-BAR, XINT4 |
| INPUT14 | eCAPx, ePWM X-BAR, XINT5 |
| INPUT15 | eCAPx |
| INPUT16 | eCAPx |

## 9.2 ePWM and GPIO Output X-BAR

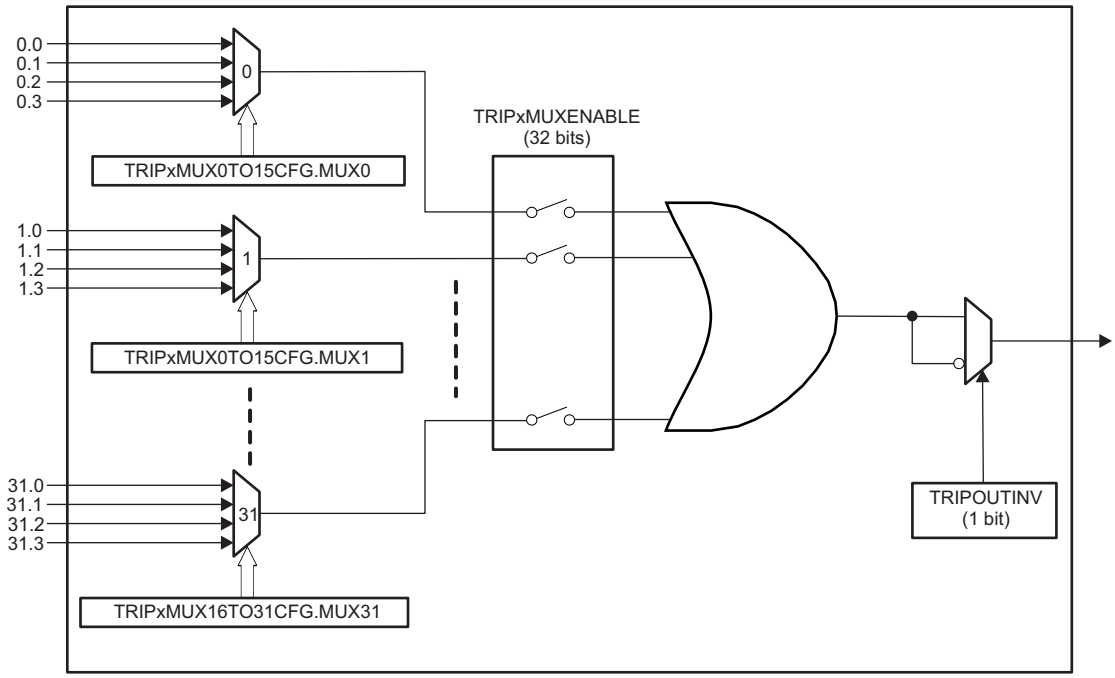
### *9.2.1 ePWM X-BAR*

The ePWM X-BAR brings signals to the ePWM modules. Specifically, the ePWM X-BAR is connected to the Digital Compare (DC) submodule of each ePWM module for actions such as tripzones and syncing. Please refer to the *ePWM* chapter for more information on additional ways the DC submodule can be used. Figure 9-4 shows the architecture of the ePWM X-BAR. It is worth noting that the architecture of the ePWM X-BAR is identical to the architecture of the Output X-BAR (with the exception of the output latch).

#### 9.2.1.1 ePWM X-BAR Architecture

The ePWM X-BAR has eight outputs which are routed to each ePWM module. Figure 18-57 represents the architecture of a single output but it is identical to the architecture of all of the other outputs.

**Figure 9-2. ePWM Architecture - Single Output**




First, determine the signal(s) which should be passed to the ePWM by referencing Table 9-2. You may select up to one signal per mux (31 total muxes) for each TRIPx output. Select the inputs to each mux via the TRIPxMUX0TO15CFG and TRIPxMUX16TO31CFG registers. In order to pass any signal through to the ePWM, you must also enable the mux in the TRIPxMUXENABLE register. All muxes which are enabled will be logically OR'd before being passed on to the respective TRIPx signal on the ePWM. You may also optionally invert the signal via the TRIPOUTINV register.

**Table 9-2. ePWM X-BAR Mux Configuration Table**

| Mux | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 0 | CMPSS1.CTRIPH | CMPSS1.CTRIPH_OR_CTRIPL | ADCAEVT1 | ECAP1OUT |
| 1 | CMPSS1.CTRIPL | INPUTXBAR1 | | ADCCEVT1 |
| 2 | CMPSS2.CTRIPH | CMPSS2.CTRIPH_OR_CTRIPL | ADCAEVT2 | ECAP2OUT |
| 3 | CMPSS2.CTRIPL | INPUTXBAR2 | | ADCCEVT2 |
| 4 | CMPSS3.CTRIPH | CMPSS3.CTRIPH_OR_CTRIPL | ADCAEVT3 | ECAP3OUT |
| 5 | CMPSS3.CTRIPL | INPUTXBAR3 | | ADCCEVT3 |
| 6 | CMPSS4.CTRIPH | CMPSS4.CTRIPH_OR_CTRIPL | ADCAEVT4 | ECAP4OUT |
| 7 | CMPSS4.CTRIPL | INPUTXBAR4 | | ADCCEVT4 |
| 8 | CMPSS5.CTRIPH | CMPSS5.CTRIPH_OR_CTRIPL | ADCBEVT1 | ECAP5OUT |
| 9 | CMPSS5.CTRIPL | INPUTXBAR5 | | |
| 10 | CMPSS6.CTRIPH | CMPSS6.CTRIPH_OR_CTRIPL | ADCBEVT2 | ECAP6OUT |
| 11 | CMPSS6.CTRIPL | INPUTXBAR6 | | |
| 12 | CMPSS7.CTRIPH | CMPSS7.CTRIPH_OR_CTRIPL | ADCBEVT3 | ECAP7OUT |
| 13 | CMPSS7.CTRIPL | ADCSOCAO | | |
| 14 | | | ADCBEVT4 | EXTSYNCOUT |
| 15 | | ADCSOCBO | | |
| 16 | SD1FLT1.COMPH | SD1FLT1.COMPH_OR_COMPL | | |
| 17 | SD1FLT1.COMPL | INPUT7 | | CLAHALT |
| 18 | SD1FLT2.COMPH | SD1FLT2.COMPH_OR_COMPL | | |
| 19 | SD1FLT2.COMPL | INPUT8 | | |
| 20 | SD1FLT3.COMPH | SD1FLT3.COMPH_OR_COMPL | | |
| 21 | SD1FLT3.COMPL | INPUT9 | | |
| 22 | SD1FLT4.COMPH | SD1FLT4.COMPH_OR_COMPL | | |
| 23 | | INPUT10 | | |
| 24 | | | | |
| 25 | | INPUT11 | | |
| 26 | | | | |
| 27 | | INPUT12 | | |
| 28 | | | | |
| 29 | | INPUT13 | | |
| 30 | | | | |
| 31 | | INPUT14 | | |

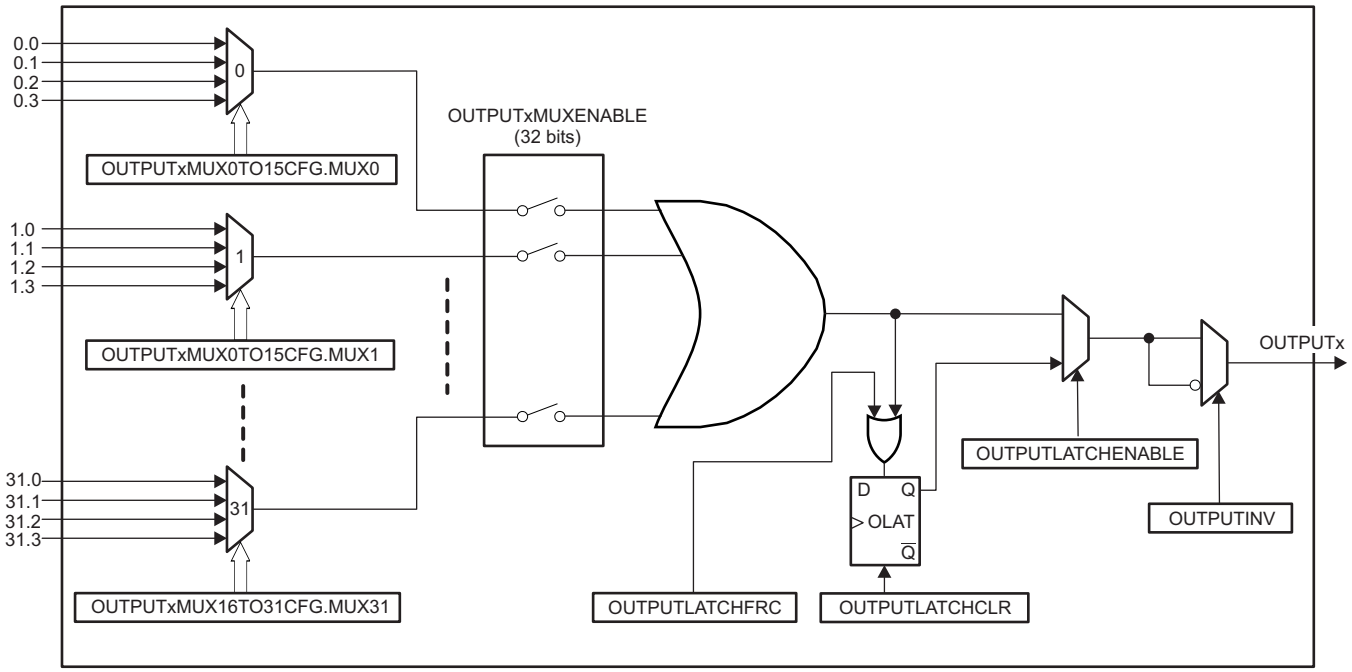
### 9.2.2 GPIO Output X-BAR

The GPIO Output X-BAR takes signals from inside the device and brings them out to a GPIO. Figure 9-4 shows the architecture of the GPIO Output X-BAR. The signals which are available to bring to the GPIO are listed in Table 9-3. The X-BAR contains eight outputs and each will contain at least one position on the GPIO mux, denoted as OUTPUTXBARx. The X-BAR allows the selection of a single signal or a logical OR of up to 32 signals.

#### 9.2.2.1 GPIO Output X-BAR Architecture

The Output X-BAR has eight outputs which are routed to the GPIO module. Figure 9-3 represents the architecture of a single output, but it is identical to the architecture of all of the other outputs. It is worth noting that the architecture of the Output X-BAR (with the exception of the output latch) is identical to the architecture of the ePWM X-BAR.

**Figure 9-3. GPIO Output X-BAR Architecture**




First, determine the signal(s) which should be passed to the GPIO by referencing Table 9-3. You may select up to one signal per mux (32 total muxes) for each OUTPUTXBARx output. Select the inputs to each mux via the OUTPUTxMUX0TO15CFG and OUTPUTxMUX16TO31CFG registers.

In order to pass any signal through to the GPIO, you must also enable the mux in the OUTPUTxMUXENABLE register. All muxes which are enabled will be logically OR'd before being passed on to the respective OUTPUTx signal on the GPIO module. You may also optionally invert the signal via the OUTPUTINV register. The signal will only be seen on the GPIO if the proper OUTPUTx muxing options are selected via the GpioCtrlRegs.GPxMUX and GpioCtrlRegs.GPxGMUX registers.

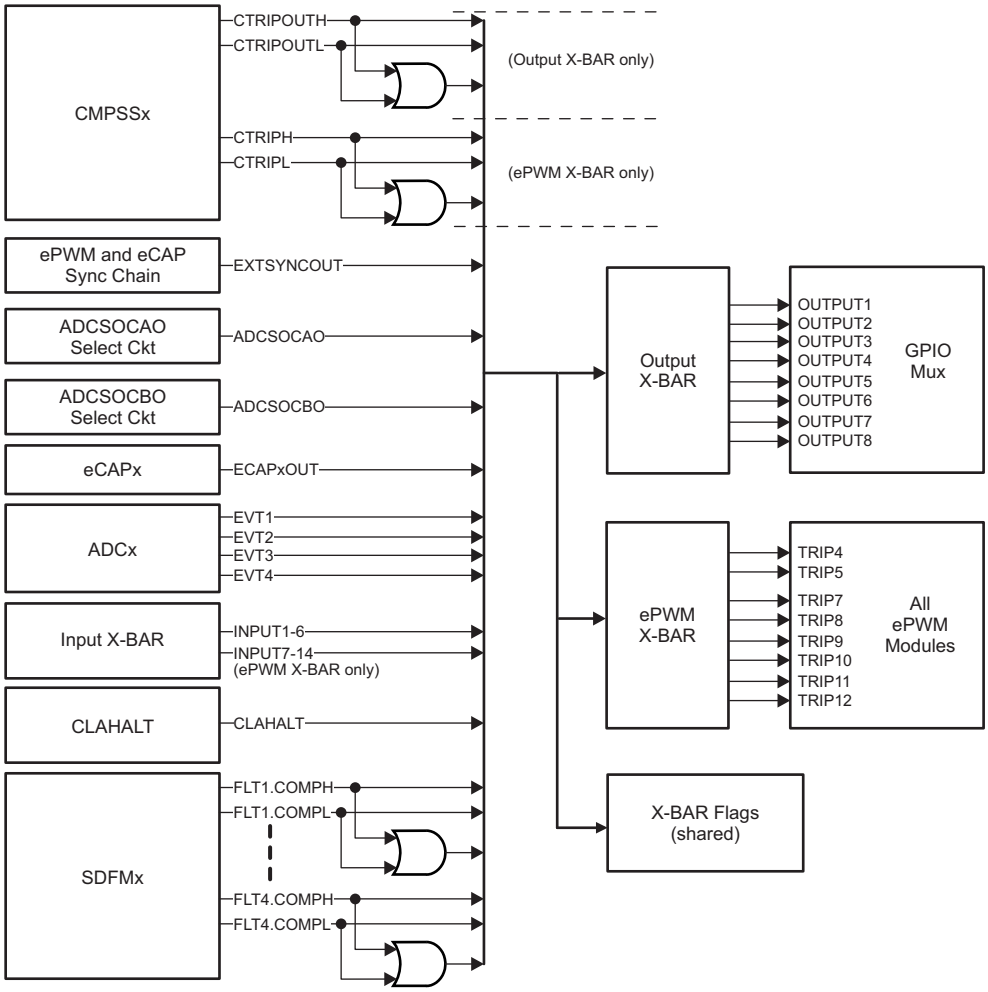
**Table 9-3. Output X-Bar Mux Configuration Table**

| Mux | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | CMPSS1.CTRIPOUTH | CMPSS1.CTRIPOUTH_OR_CTRIPOUTL | ADCAEVT1 | ECAP1OUT |
| 1 | CMPSS1.CTRIPOUTL | INPUTXBAR1 | | ADCCEVT1 |
| 2 | CMPSS2.CTRIPOUTH | CMPSS2.CTRIPOUTH_OR_CTRIPOUTL | ADCAEVT2 | ECAP2OUT |
| 3 | CMPSS2.CTRIPOUTL | INPUTXBAR2 | | ADCCEVT2 |
| 4 | CMPSS3.CTRIPOUTH | CMPSS3.CTRIPOUTH_OR_CTRIPOUTL | ADCAEVT3 | ECAP3OUT |
| 5 | CMPSS3.CTRIPOUTL | INPUTXBAR3 | | ADCCEVT3 |
| 6 | CMPSS4.CTRIPOUTH | CMPSS4.CTRIPOUTH_OR_CTRIPOUTL | ADCAEVT4 | ECAP4OUT |
| 7 | CMPSS4.CTRIPOUTL | INPUTXBAR4 | | ADCCEVT4 |
| 8 | CMPSS5.CTRIPOUTH | CMPSS5.CTRIPOUTH_OR_CTRIPOUTL | ADCBEVT1 | ECAP5OUT |
| 9 | CMPSS5.CTRIPOUTL | INPUTXBAR5 | | |
| 10 | CMPSS6.CTRIPOUTH | CMPSS6.CTRIPOUTH_OR_CTRIPOUTL | ADCBEVT2 | ECAP6OUT |
| 11 | CMPSS6.CTRIPOUTL | INPUTXBAR6 | | |
| 12 | CMPSS7.CTRIPOUTH | CMPSS7.CTRIPOUTH_OR_CTRIPOUTL | ADCBEVT3 | ECAP7OUT |
| 13 | CMPSS7.CTRIPOUTL | ADCSOCAO | | |
| 14 | | | ADCBEVT4 | EXTSYNCOUT |
| 15 | | ADCSOCBO | | |
| 16 | SD1FLT1.COMPH | SD1FLT1.COMPH_OR_COMPL | | |
| 17 | SD1FLT1.COMPL | | | CLAHALT |
| 18 | SD1FLT2.COMPH | SD1FLT2.COMPH_OR_COMPL | | |
| 19 | SD1FLT2.COMPL | | | |
| 20 | SD1FLT3.COMPH | SD1FLT3.COMPH_OR_COMPL | | |
| 21 | SD1FLT3.COMPL | | | |
| 22 | SD1FLT4.COMPH | SD1FLT4.COMPH_OR_COMPL | | |
| 23 | SD1FLT4.COMPL | | | |

### 9.2.3 X-BAR Flags

With the exception of the CMPSS signals, the ePWM X-BAR and the Output X-BAR have all of the same input signals. Due to the inputs being similar, the ePWM X-BAR and Output X-BAR leverage a single set of input flags to indicate which input signals have been triggered. This allows software to check the input flags when an event occurs. See Figure 9-4 for more information. There is a bit allocated for each input signal in one of the XBARFLGx registers. The flag will remain set until cleared through the appropriate XBARCLRx register.

**Figure 9-4. ePWM and Output X-BARs Sources**



## 9.3 X-BAR Registers

### 9.3.1 Crossbar Base Addresses

**Table 9-4. System Control Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| InputXbarRegs | INPUT_XBAR_REGS | 0x0000_7900 | 0x0000_791F |
| XbarRegs | XBAR_REGS | 0x0000_7920 | 0x0000_793F |
| EpwmXbarRegs | EPWM_XBAR_REGS | 0x0000_7A00 | 0x0000_7A3F |
| OutputXbarRegs | OUTPUT_XBAR_REGS | 0x0000_7A80 | 0x0000_7ABF |

### 9.3.1.1 INPUT_XBAR_REGS Registers

Table 9-5 lists the memory-mapped registers for the INPUT_XBAR_REGS. All register offset addresses not listed in Table 9-5 should be considered as reserved locations and the register contents should not be modified.

**Table 9-5. INPUT_XBAR_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | INPUT1SELECT | INPUT1 Input Select Register (GPIO0 to x) | EALLOW | 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 |
| Eh | INPUT15SELECT | INPUT15 Input Select Register (GPIO0 to x) | EALLOW | Go |
| Fh | INPUT16SELECT | INPUT16 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 9-6 shows the codes that are used for access types in this section.

**Table 9-6. INPUT_XBAR_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| WSOnce | SOnce<br>W | Set once<br>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. |

#### 9.3.1.1.1 INPUT1SELECT Register (Offset = 0h) [reset = FFFEh]

INPUT1SELECT is shown in Figure 9-5 and described in Table 9-7.

Return to Summary Table.

INPUT1 Input Select Register (GPIO0 to x)

**Figure 9-5. INPUT1SELECT Register**

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

**Table 9-7. INPUT1SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | FFFEh | Select GPIO for INPUTx signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xFFFD: '1' will be driven to the destination<br>0xFFFE: '1' will be driven to the destination<br>0xFFFF: '0' will be driven to the destination<br>NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.1.2 INPUT2SELECT Register (Offset = 1h) [reset = FFFEh]

INPUT2SELECT is shown in Figure 9-6 and described in Table 9-8.

Return to Summary Table.

INPUT2 Input Select Register (GPIO0 to x)

**Figure 9-6. INPUT2SELECT Register**

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

**Table 9-8. INPUT2SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | FFFEh | Select GPIO for INPUTx signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xFFFD: '1' will be driven to the destination<br>0xFFFE: '1' will be driven to the destination<br>0xFFFF: '0' will be driven to the destination<br>NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.1.3 INPUT3SELECT Register (Offset = 2h) [reset = FFFEh]

INPUT3SELECT is shown in Figure 9-7 and described in Table 9-9.

Return to Summary Table.

INPUT3 Input Select Register (GPIO0 to x)

**Figure 9-7. INPUT3SELECT Register**

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

**Table 9-9. INPUT3SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | FFFEh | Select GPIO for INPUTx signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xFFFD: '1' will be driven to the destination<br>0xFFFE: '1' will be driven to the destination<br>0xFFFF: '0' will be driven to the destination<br>NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.1.4 INPUT4SELECT Register (Offset = 3h) [reset = FFFEh]

INPUT4SELECT is shown in Figure 9-8 and described in Table 9-10.

Return to Summary Table.

INPUT4 Input Select Register (GPIO0 to x)

**Figure 9-8. INPUT4SELECT Register**

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

**Table 9-10. INPUT4SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | FFFEh | Select GPIO for INPUTx signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xFFFD: '1' will be driven to the destination<br>0xFFFE: '1' will be driven to the destination<br>0xFFFF: '0' will be driven to the destination<br>NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.1.5 INPUT5SELECT Register (Offset = 4h) [reset = FFFEh]

INPUT5SELECT is shown in Figure 9-9 and described in Table 9-11.

Return to Summary Table.

INPUT5 Input Select Register (GPIO0 to x)

**Figure 9-9. INPUT5SELECT Register**

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

**Table 9-11. INPUT5SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | FFFEh | Select GPIO for INPUTx signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xFFFD: '1' will be driven to the destination<br>0xFFFE: '1' will be driven to the destination<br>0xFFFF: '0' will be driven to the destination<br>NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.1.6 INPUT6SELECT Register (Offset = 5h) [reset = FFFEh]

INPUT6SELECT is shown in Figure 9-10 and described in Table 9-12.

Return to Summary Table.

INPUT6 Input Select Register (GPIO0 to x)

**Figure 9-10. INPUT6SELECT Register**

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

**Table 9-12. INPUT6SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | FFFEh | Select GPIO for INPUTx signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xFFFD: '1' will be driven to the destination<br>0xFFFE: '1' will be driven to the destination<br>0xFFFF: '0' will be driven to the destination<br>NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.1.7 INPUT7SELECT Register (Offset = 6h) [reset = FFFEh]

INPUT7SELECT is shown in Figure 9-11 and described in Table 9-13.

Return to Summary Table.

INPUT7 Input Select Register (GPIO0 to x)

**Figure 9-11. INPUT7SELECT Register**

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

**Table 9-13. INPUT7SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | FFFEh | Select GPIO for INPUTx signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xFFFD: '1' will be driven to the destination<br>0xFFFE: '1' will be driven to the destination<br>0xFFFF: '0' will be driven to the destination<br>NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.1.8 INPUT8SELECT Register (Offset = 7h) [reset = FFFEh]

INPUT8SELECT is shown in Figure 9-12 and described in Table 9-14.

Return to Summary Table.

INPUT8 Input Select Register (GPIO0 to x)

**Figure 9-12. INPUT8SELECT Register**

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

**Table 9-14. INPUT8SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | FFFEh | Select GPIO for INPUTx signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xFFFD: '1' will be driven to the destination<br>0xFFFE: '1' will be driven to the destination<br>0xFFFF: '0' will be driven to the destination<br>NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.1.9 INPUT9SELECT Register (Offset = 8h) [reset = FFFEh]

INPUT9SELECT is shown in Figure 9-13 and described in Table 9-15.

Return to Summary Table.

INPUT9 Input Select Register (GPIO0 to x)

**Figure 9-13. INPUT9SELECT Register**

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

**Table 9-15. INPUT9SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | FFFEh | Select GPIO for INPUTx signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xFFFD: '1' will be driven to the destination<br>0xFFFE: '1' will be driven to the destination<br>0xFFFF: '0' will be driven to the destination<br>NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.1.10 INPUT10SELECT Register (Offset = 9h) [reset = FFFEh]

INPUT10SELECT is shown in Figure 9-14 and described in Table 9-16.

Return to Summary Table.

INPUT10 Input Select Register (GPIO0 to x)

**Figure 9-14. INPUT10SELECT Register**

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

**Table 9-16. INPUT10SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | FFFEh | Select GPIO for INPUTx signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xFFFD: '1' will be driven to the destination<br>0xFFFE: '1' will be driven to the destination<br>0xFFFF: '0' will be driven to the destination<br>NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.1.11 INPUT11SELECT Register (Offset = Ah) [reset = FFFEh]

INPUT11SELECT is shown in Figure 9-15 and described in Table 9-17.

Return to Summary Table.

INPUT11 Input Select Register (GPIO0 to x)

**Figure 9-15. INPUT11SELECT Register**

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

**Table 9-17. INPUT11SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | FFFEh | Select GPIO for INPUTx signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xFFFD: '1' will be driven to the destination<br>0xFFFE: '1' will be driven to the destination<br>0xFFFF: '0' will be driven to the destination<br>NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.1.12 INPUT12SELECT Register (Offset = Bh) [reset = FFFEh]

INPUT12SELECT is shown in Figure 9-16 and described in Table 9-18.

Return to Summary Table.

INPUT12 Input Select Register (GPIO0 to x)

**Figure 9-16. INPUT12SELECT Register**

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

**Table 9-18. INPUT12SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | FFFEh | Select GPIO for INPUTx signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xFFFD: '1' will be driven to the destination<br>0xFFFE: '1' will be driven to the destination<br>0xFFFF: '0' will be driven to the destination<br>NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.1.13 INPUT13SELECT Register (Offset = Ch) [reset = FFFEh]

INPUT13SELECT is shown in Figure 9-17 and described in Table 9-19.

Return to Summary Table.

INPUT13 Input Select Register (GPIO0 to x)

**Figure 9-17. INPUT13SELECT Register**

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

**Table 9-19. INPUT13SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | FFFEh | Select GPIO for INPUTx signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xFFFD: '1' will be driven to the destination<br>0xFFFE: '1' will be driven to the destination<br>0xFFFF: '0' will be driven to the destination<br>NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.1.14 INPUT14SELECT Register (Offset = Dh) [reset = FFFEh]

INPUT14SELECT is shown in Figure 9-18 and described in Table 9-20.

Return to Summary Table.

INPUT14 Input Select Register (GPIO0 to x)

**Figure 9-18. INPUT14SELECT Register**

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

**Table 9-20. INPUT14SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | FFFEh | Select GPIO for INPUTx signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xFFFD: '1' will be driven to the destination<br>0xFFFE: '1' will be driven to the destination<br>0xFFFF: '0' will be driven to the destination<br>NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.1.15 INPUT15SELECT Register (Offset = Eh) [reset = FFFEh]

INPUT15SELECT is shown in Figure 9-19 and described in Table 9-21.

Return to Summary Table.

INPUT15 Input Select Register (GPIO0 to x)

**Figure 9-19. INPUT15SELECT Register**

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

**Table 9-21. INPUT15SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | FFFEh | Select GPIO for INPUTx signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xFFFD: '1' will be driven to the destination<br>0xFFFE: '1' will be driven to the destination<br>0xFFFF: '0' will be driven to the destination<br>NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.1.16 INPUT16SELECT Register (Offset = Fh) [reset = FFFEh]

INPUT16SELECT is shown in Figure 9-20 and described in Table 9-22.

Return to Summary Table.

INPUT16 Input Select Register (GPIO0 to x)

**Figure 9-20. INPUT16SELECT Register**

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

**Table 9-22. INPUT16SELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SELECT | R/W | FFFEh | Select GPIO for INPUTx signal:<br>0x0 : Select GPIO0<br>0x1 : Select GPIO1<br>0x2 : Select GPIO2<br>...<br>0xFFFD: '1' will be driven to the destination<br>0xFFFE: '1' will be driven to the destination<br>0xFFFF: '0' will be driven to the destination<br>NOTE: SELECT value greater than the available number of GPIO pins on a device (except 0xFFFF) will cause the destination to be driven '1'.<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.1.17 INPUTSELECTLOCK Register (Offset = 1Eh) [reset = 0h]

INPUTSELECTLOCK is shown in Figure 9-21 and described in Table 9-23.

Return to Summary Table.

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. 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 9-23. 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: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 14 | INPUT15SELECT | R/WSOnce | 0h | Lock bit for INPUT15SELECT Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 13 | INPUT14SELECT | R/WSOnce | 0h | Lock bit for INPUT14SELECT Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 12 | INPUT13SELECT | R/WSOnce | 0h | Lock bit for INPUT13SELECT Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 11 | INPUT12SELECT | R/WSOnce | 0h | Lock bit for INPUT12SELECT Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 10 | INPUT11SELECT | R/WSOnce | 0h | Lock bit for INPUT11SELECT Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |

**Table 9-23. INPUTSELECTLOCK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9 | INPUT10SELECT | R/WSOnce | 0h | Lock bit for INPUT10SELECT Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 8 | INPUT9SELECT | R/WSOnce | 0h | Lock bit for INPUT9SELECT Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 7 | INPUT8SELECT | R/WSOnce | 0h | Lock bit for INPUT8SELECT Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 6 | INPUT7SELECT | R/WSOnce | 0h | Lock bit for INPUT7SELECT Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 5 | INPUT6SELECT | R/WSOnce | 0h | Lock bit for INPUT6SELECT Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 4 | INPUT5SELECT | R/WSOnce | 0h | Lock bit for INPUT5SELECT Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 3 | INPUT4SELECT | R/WSOnce | 0h | Lock bit for INPUT4SELECT Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 2 | INPUT3SELECT | R/WSOnce | 0h | Lock bit for INPUT3SELECT Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 1 | INPUT2SELECT | R/WSOnce | 0h | Lock bit for INPUT2SELECT Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |
| 0 | INPUT1SELECT | R/WSOnce | 0h | Lock bit for INPUT1SELECT Register:<br>0: Respective register is not locked<br>1: Respective register is locked.<br>Reset type: CPU1.SYSRSn |

### 9.3.1.2 XBAR_REGS Registers

Table 9-24 lists the memory-mapped registers for the XBAR_REGS. All register offset addresses not listed in Table 9-24 should be considered as reserved locations and the register contents should not be modified.

**Table 9-24. XBAR_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | XBARFLG1 | X-Bar Input Flag Register 1 | | Go |
| 2h | XBARFLG2 | X-Bar Input Flag Register 2 | | Go |
| 4h | XBARFLG3 | X-Bar Input Flag Register 3 | | Go |
| 6h | XBARFLG4 | X-Bar Input Flag Register 4 | | 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 |
| Eh | XBARCLR4 | X-Bar Input Flag Clear Register 4 | | Go |

Complex bit access types are encoded to fit into small table cells. Table 9-25 shows the codes that are used for access types in this section.

**Table 9-25. XBAR_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W=1 | 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. |

#### 9.3.1.2.1 XBARFLG1 Register (Offset = 0h) [reset = 0h]

XBARFLG1 is shown in Figure 9-22 and described in Table 9-26.

Return to 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 9-26. XBARFLG1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | CMPSS8_CTRIPOUTH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS8_CTRIPOUTH input was triggered<br>0: CMPSS8_CTRIPOUTH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 30 | CMPSS8_CTRIPOUTL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS8_CTRIPOUTL input was triggered<br>0: CMPSS8_CTRIPOUTL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 29 | CMPSS7_CTRIPOUTH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS7_CTRIPOUTH input was triggered<br>0: CMPSS7_CTRIPOUTH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |

**Table 9-26. XBARFLG1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 28 | CMPSS7_CTRIPOUTL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS7_CTRIPOUTL input was triggered<br>0: CMPSS7_CTRIPOUTL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 27 | CMPSS6_CTRIPOUTH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS6_CTRIPOUTH input was triggered<br>0: CMPSS6_CTRIPOUTH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 26 | CMPSS6_CTRIPOUTL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS6_CTRIPOUTL input was triggered<br>0: CMPSS6_CTRIPOUTL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 25 | CMPSS5_CTRIPOUTH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS5_CTRIPOUTH input was triggered<br>0: CMPSS5_CTRIPOUTH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 24 | CMPSS5_CTRIPOUTL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS5_CTRIPOUTL input was triggered<br>0: CMPSS5_CTRIPOUTL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 23 | CMPSS4_CTRIPOUTH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS4_CTRIPOUTH input was triggered<br>0: CMPSS4_CTRIPOUTH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |

**Table 9-26. XBARFLG1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 22 | CMPSS4_CTRIPOUTL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS4_CTRIPOUTL input was triggered<br>0: CMPSS4_CTRIPOUTL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 21 | CMPSS3_CTRIPOUTH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS3_CTRIPOUTH input was triggered<br>0: CMPSS3_CTRIPOUTH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 20 | CMPSS3_CTRIPOUTL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS3_CTRIPOUTL input was triggered<br>0: CMPSS3_CTRIPOUTL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 19 | CMPSS2_CTRIPOUTH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS2_CTRIPOUTH input was triggered<br>0: CMPSS2_CTRIPOUTH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 18 | CMPSS2_CTRIPOUTL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS2_CTRIPOUTL input was triggered<br>0: CMPSS2_CTRIPOUTL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 17 | CMPSS1_CTRIPOUTH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS1_CTRIPOUTH input was triggered<br>0: CMPSS1_CTRIPOUTH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |

**Table 9-26. XBARFLG1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 16 | CMPSS1_CTRIPOUTL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS1_CTRIPOUTL input was triggered<br>0: CMPSS1_CTRIPOUTL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 15 | CMPSS8_CTRIPH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS8_CTRIPH input was triggered<br>0: CMPSS8_CTRIPH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 14 | CMPSS8_CTRIPL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS8_CTRIPL input was triggered<br>0: CMPSS8_CTRIPL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 13 | CMPSS7_CTRIPH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS7_CTRIPH input was triggered<br>0: CMPSS7_CTRIPH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 12 | CMPSS7_CTRIPL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS7_CTRIPL input was triggered<br>0: CMPSS7_CTRIPL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 11 | CMPSS6_CTRIPH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS6_CTRIPH input was triggered<br>0: CMPSS6_CTRIPH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |

**Table 9-26. XBARFLG1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 10 | CMPSS6_CTRIPL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS6_CTRIPL input was triggered<br>0: CMPSS6_CTRIPL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 9 | CMPSS5_CTRIPH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS5_CTRIPH input was triggered<br>0: CMPSS5_CTRIPH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 8 | CMPSS5_CTRIPL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS5_CTRIPL input was triggered<br>0: CMPSS5_CTRIPL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 7 | CMPSS4_CTRIPH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS4_CTRIPH input was triggered<br>0: CMPSS4_CTRIPH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 6 | CMPSS4_CTRIPL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS4_CTRIPL input was triggered<br>0: CMPSS4_CTRIPL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 5 | CMPSS3_CTRIPH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS3_CTRIPH input was triggered<br>0: CMPSS3_CTRIPH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |

**Table 9-26. XBARFLG1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | CMPSS3_CTRIPL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS3_CTRIPL input was triggered<br>0: CMPSS3_CTRIPL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 3 | CMPSS2_CTRIPH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS2_CTRIPH input was triggered<br>0: CMPSS2_CTRIPH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 2 | CMPSS2_CTRIPL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS2_CTRIPL input was triggered<br>0: CMPSS2_CTRIPL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 1 | CMPSS1_CTRIPH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS1_CTRIPH input was triggered<br>0: CMPSS1_CTRIPH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 0 | CMPSS1_CTRIPL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CMPSS1_CTRIPL input was triggered<br>0: CMPSS1_CTRIPL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.2.2 XBARFLG2 Register (Offset = 2h) [reset = 0h]

XBARFLG2 is shown in Figure 9-23 and described in Table 9-27.

Return to 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 |
| INPUT14 | INPUT13 | INPUT12 | INPUT11 | INPUT10 | INPUT9 | INPUT8 | INPUT7 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ADCSOCB | ADCSOCA | INPUT6 | INPUT5 | INPUT4 | INPUT3 | INPUT2 | INPUT1 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 9-27. XBARFLG2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | ADCCEVT1 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: ADCCEVT1 input was triggered<br>0: ADCCEVT1 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 30 | ADCBEVT4 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: ADCBEVT4 input was triggered<br>0: ADCBEVT4 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 29 | ADCBEVT3 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: ADCBEVT3 input was triggered<br>0: ADCBEVT3 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |

**Table 9-27. XBARFLG2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 28 | ADCBEVT2 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: ADCBEVT2 input was triggered<br>0: ADCBEVT2 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 27 | ADCBEVT1 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: ADCBEVT1 input was triggered<br>0: ADCBEVT1 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 26 | ADCAEVT4 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: ADCAEVT4 input was triggered<br>0: ADCAEVT4 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 25 | ADCAEVT3 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: ADCAEVT3 input was triggered<br>0: ADCAEVT3 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 24 | ADCAEVT2 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: ADCAEVT2 input was triggered<br>0: ADCAEVT2 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 23 | ADCAEVT1 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: ADCAEVT1 input was triggered<br>0: ADCAEVT1 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |

**Table 9-27. XBARFLG2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 22 | EXTSYNCOUT | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: EXTSYNCOUT input was triggered<br>0: EXTSYNCOUT Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 21 | ECAP6_OUT | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: ECAP6_OUT input was triggered<br>0: ECAP6_OUT Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 20 | ECAP5_OUT | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: ECAP5_OUT input was triggered<br>0: ECAP5_OUT Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 19 | ECAP4_OUT | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: ECAP4_OUT input was triggered<br>0: ECAP4_OUT Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 18 | ECAP3_OUT | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: ECAP3_OUT input was triggered<br>0: ECAP3_OUT Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 17 | ECAP2_OUT | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: ECAP2_OUT input was triggered<br>0: ECAP2_OUT Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |

**Table 9-27. XBARFLG2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 16 | ECAP1_OUT | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: ECAP1_OUT input was triggered<br>0: ECAP1_OUT Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 15 | INPUT14 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: INPUT14 input was triggered<br>0: INPUT14 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 14 | INPUT13 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: INPUT13 input was triggered<br>0: INPUT13 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 13 | INPUT12 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: INPUT12 input was triggered<br>0: INPUT12 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 12 | INPUT11 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: INPUT11 input was triggered<br>0: INPUT11 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 11 | INPUT10 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: INPUT10 input was triggered<br>0: INPUT10 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |

**Table 9-27. XBARFLG2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 10 | INPUT9 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: INPUT9 input was triggered<br>0: INPUT9 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 9 | INPUT8 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: INPUT8 input was triggered<br>0: INPUT8 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 8 | INPUT7 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: INPUT7 input was triggered<br>0: INPUT7 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 7 | ADCSOCB | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: ADCSOCB input was triggered<br>0: ADCSOCB Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 6 | ADCSOCA | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: ADCSOCA input was triggered<br>0: ADCSOCA Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 5 | INPUT6 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: INPUT6 input was triggered<br>0: INPUT6 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |

**Table 9-27. XBARFLG2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | INPUT5 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: INPUT5 input was triggered<br>0: INPUT5 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 3 | INPUT4 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: INPUT4 input was triggered<br>0: INPUT4 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 2 | INPUT3 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: INPUT3 input was triggered<br>0: INPUT3 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 1 | INPUT2 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: INPUT2 input was triggered<br>0: INPUT2 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 0 | INPUT1 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: INPUT1 input was triggered<br>0: INPUT1 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.2.3 XBARFLG3 Register (Offset = 4h) [reset = 0h]

XBARFLG3 is shown in Figure 9-24 and described in Table 9-28.

Return to 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 |
|---|---|---|---|---|---|---|---|
| SD1FLT4_DRINT | SD1FLT4_COMPZ | SD1FLT3_DRINT | SD1FLT3_COMPZ | SD1FLT2_DRINT | SD1FLT2_COMPZ | SD1FLT1_DRINT | SD1FLT1_COMPZ |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ECAP7_OUT | SD2FLT4_COMPH | SD2FLT4_COMPL | SD2FLT3_COMPH | SD2FLT3_COMPL | SD2FLT2_COMPH | SD2FLT2_COMPL | SD2FLT1_COMPH |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SD2FLT1_COMPL | SD1FLT4_COMPH | SD1FLT4_COMPL | SD1FLT3_COMPH | SD1FLT3_COMPL | SD1FLT2_COMPH | SD1FLT2_COMPL | SD1FLT1_COMPH |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SD1FLT1_COMPL | | | | | ADCCEVT4 | ADCCEVT3 | ADCCEVT2 |
| R-0h | | | | | R-0h | R-0h | R-0h |

**Table 9-28. XBARFLG3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | SD1FLT4_DRINT | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD1FLT4_DRINT input was triggered<br>0: SD1FLT4_DRINT Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 30 | SD1FLT4_COMPZ | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD1FLT4_COMPZ input was triggered<br>0: SD1FLT4_COMPZ Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 29 | SD1FLT3_DRINT | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD1FLT3_DRINT input was triggered<br>0: SD1FLT3_DRINT Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |

**Table 9-28. XBARFLG3 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 28 | SD1FLT3_COMPZ | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD1FLT3_COMPZ input was triggered<br>0: SD1FLT3_COMPZ Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 27 | SD1FLT2_DRINT | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD1FLT2_DRINT input was triggered<br>0: SD1FLT2_DRINT Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 26 | SD1FLT2_COMPZ | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD1FLT2_COMPZ input was triggered<br>0: SD1FLT2_COMPZ Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 25 | SD1FLT1_DRINT | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD1FLT1_DRINT input was triggered<br>0: SD1FLT1_DRINT Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 24 | SD1FLT1_COMPZ | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD1FLT1_COMPZ input was triggered<br>0: SD1FLT1_COMPZ Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 23 | ECAP7_OUT | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: ECAP7_OUT input was triggered<br>0: ECAP7_OUT Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |

**Table 9-28. XBARFLG3 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 22 | SD2FLT4_COMPH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD2FLT4_COMPH input was triggered<br>0: SD2FLT4_COMPH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 21 | SD2FLT4_COMPL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD2FLT4_COMPL input was triggered<br>0: SD2FLT4_COMPL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 20 | SD2FLT3_COMPH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD2FLT3_COMPH input was triggered<br>0: SD2FLT3_COMPH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 19 | SD2FLT3_COMPL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD2FLT3_COMPL input was triggered<br>0: SD2FLT3_COMPL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 18 | SD2FLT2_COMPH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD2FLT2_COMPH input was triggered<br>0: SD2FLT2_COMPH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 17 | SD2FLT2_COMPL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD2FLT2_COMPL input was triggered<br>0: SD2FLT2_COMPL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |

**Table 9-28. XBARFLG3 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 16 | SD2FLT1_COMPH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD2FLT1_COMPH input was triggered<br>0: SD2FLT1_COMPH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 15 | SD2FLT1_COMPL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD2FLT1_COMPL input was triggered<br>0: SD2FLT1_COMPL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 14 | SD1FLT4_COMPH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD1FLT4_COMPH input was triggered<br>0: SD1FLT4_COMPH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 13 | SD1FLT4_COMPL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD1FLT4_COMPL input was triggered<br>0: SD1FLT4_COMPL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 12 | SD1FLT3_COMPH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD1FLT3_COMPH input was triggered<br>0: SD1FLT3_COMPH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 11 | SD1FLT3_COMPL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD1FLT3_COMPL input was triggered<br>0: SD1FLT3_COMPL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |

**Table 9-28. XBARFLG3 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 10 | SD1FLT2_COMPH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD1FLT2_COMPH input was triggered<br>0: SD1FLT2_COMPH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 9 | SD1FLT2_COMPL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD1FLT2_COMPL input was triggered<br>0: SD1FLT2_COMPL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 8 | SD1FLT1_COMPH | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD1FLT1_COMPH input was triggered<br>0: SD1FLT1_COMPH Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 7 | SD1FLT1_COMPL | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD1FLT1_COMPL input was triggered<br>0: SD1FLT1_COMPL Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | ADCCEVT4 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: ADCCEVT4 input was triggered<br>0: ADCCEVT4 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 1 | ADCCEVT3 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: ADCCEVT3 input was triggered<br>0: ADCCEVT3 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |

**Table 9-28. XBARFLG3 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | ADCCEVT2 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: ADCCEVT2 input was triggered<br>0: ADCCEVT2 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.2.4 XBARFLG4 Register (Offset = 6h) [reset = 0h]

XBARFLG4 is shown in Figure 9-25 and described in Table 9-29.

Return to 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 9-25. XBARFLG4 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CLAHALT | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | RESERVED | RESERVED | RESERVED | MCANA_FEVT2 | MCANA_FEVT1 | MCANA_FEVT0 | EMAC_PPS0 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SD2FLT4_DRINT | SD2FLT4_COMPZ | SD2FLT3_DRINT | SD2FLT3_COMPZ | SD2FLT2_DRINT | SD2FLT2_COMPZ | SD2FLT1_DRINT | SD2FLT1_COMPZ |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 9-29. XBARFLG4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | CLAHALT | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: CLAHALT input was triggered<br>0: CLAHALT Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 30 | RESERVED | R | 0h | Reserved |
| 29 | RESERVED | R | 0h | Reserved |
| 28 | RESERVED | R | 0h | Reserved |
| 27 | RESERVED | R | 0h | Reserved |
| 26 | RESERVED | R | 0h | Reserved |
| 25 | RESERVED | R | 0h | Reserved |
| 24 | RESERVED | R | 0h | Reserved |
| 23 | RESERVED | R | 0h | Reserved |
| 22 | RESERVED | R | 0h | Reserved |
| 21 | RESERVED | R | 0h | Reserved |
| 20 | RESERVED | R | 0h | Reserved |
| 19 | RESERVED | R | 0h | Reserved |
| 18 | RESERVED | R | 0h | Reserved |
| 17 | RESERVED | R | 0h | Reserved |
| 16 | RESERVED | R | 0h | Reserved |
| 15 | RESERVED | R | 0h | Reserved |
| 14 | RESERVED | R | 0h | Reserved |

**Table 9-29. XBARFLG4 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 13 | RESERVED | R | 0h | Reserved |
| 12 | RESERVED | R | 0h | Reserved |
| 11 | MCANA_FEVT2 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: MCANA_FEVT2 input was triggered<br>0: MCANA_FEVT2 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 10 | MCANA_FEVT1 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: MCANA_FEVT1 input was triggered<br>0: MCANA_FEVT1 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 9 | MCANA_FEVT0 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: MCANA_FEVT0 input was triggered<br>0: MCANA_FEVT0 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 8 | EMAC_PPS0 | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: EMAC_PPS0 input was triggered<br>0: EMAC_PPS0 Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 7 | SD2FLT4_DRINT | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD2FLT4_DRINT input was triggered<br>0: SD2FLT4_DRINT Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 6 | SD2FLT4_COMPZ | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD2FLT4_COMPZ input was triggered<br>0: SD2FLT4_COMPZ Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |

**Table 9-29. XBARFLG4 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | SD2FLT3_DRINT | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD2FLT3_DRINT input was triggered<br>0: SD2FLT3_DRINT Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 4 | SD2FLT3_COMPZ | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD2FLT3_COMPZ input was triggered<br>0: SD2FLT3_COMPZ Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 3 | SD2FLT2_DRINT | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD2FLT2_DRINT input was triggered<br>0: SD2FLT2_DRINT Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 2 | SD2FLT2_COMPZ | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD2FLT2_COMPZ input was triggered<br>0: SD2FLT2_COMPZ Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 1 | SD2FLT1_DRINT | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD2FLT1_DRINT input was triggered<br>0: SD2FLT1_DRINT Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |
| 0 | SD2FLT1_COMPZ | R | 0h | This register is used to Flag the inputs of the X-Bars to provide software knowledge of the input sources which got triggered.<br>1: SD2FLT1_COMPZ input was triggered<br>0: SD2FLT1_COMPZ Input was not triggered<br>Note:<br>[1] setting of this bit has priority over clear by software<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.2.5 XBARCLR1 Register (Offset = 8h) [reset = 0h]

XBARCLR1 is shown in Figure 9-26 and described in Table 9-30.

Return to 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 9-26. 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/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CMPSS4_CTRIPOUTH | CMPSS4_CTRIPOUTL | CMPSS3_CTRIPOUTH | CMPSS3_CTRIPOUTL | CMPSS2_CTRIPOUTH | CMPSS2_CTRIPOUTL | CMPSS1_CTRIPOUTH | CMPSS1_CTRIPOUTL |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CMPSS8_CTRIPH | CMPSS8_CTRIPL | CMPSS7_CTRIPH | CMPSS7_CTRIPL | CMPSS6_CTRIPH | CMPSS6_CTRIPL | CMPSS5_CTRIPH | CMPSS5_CTRIPL |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CMPSS4_CTRIPH | CMPSS4_CTRIPL | CMPSS3_CTRIPH | CMPSS3_CTRIPL | CMPSS2_CTRIPH | CMPSS2_CTRIPL | CMPSS1_CTRIPH | CMPSS1_CTRIPL |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 9-30. XBARCLR1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | CMPSS8_CTRIPOUTH | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS8_CTRIPOUTH bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 30 | CMPSS8_CTRIPOUTL | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS8_CTRIPOUTL bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 29 | CMPSS7_CTRIPOUTH | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS7_CTRIPOUTH bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 28 | CMPSS7_CTRIPOUTL | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS7_CTRIPOUTL bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 27 | CMPSS6_CTRIPOUTH | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS6_CTRIPOUTH bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 26 | CMPSS6_CTRIPOUTL | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS6_CTRIPOUTL bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |

**Table 9-30. XBARCLR1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 25 | CMPSS5_CTRIPOUTH | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS5_CTRIPOUTH bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 24 | CMPSS5_CTRIPOUTL | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS5_CTRIPOUTL bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 23 | CMPSS4_CTRIPOUTH | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS4_CTRIPOUTH bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 22 | CMPSS4_CTRIPOUTL | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS4_CTRIPOUTL bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 21 | CMPSS3_CTRIPOUTH | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS3_CTRIPOUTH bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 20 | CMPSS3_CTRIPOUTL | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS3_CTRIPOUTL bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 19 | CMPSS2_CTRIPOUTH | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS2_CTRIPOUTH bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 18 | CMPSS2_CTRIPOUTL | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS2_CTRIPOUTL bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 17 | CMPSS1_CTRIPOUTH | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS1_CTRIPOUTH bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 16 | CMPSS1_CTRIPOUTL | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS1_CTRIPOUTL bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 15 | CMPSS8_CTRIPH | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS8_CTRIPH bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 14 | CMPSS8_CTRIPL | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS8_CTRIPL bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 13 | CMPSS7_CTRIPH | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS7_CTRIPH bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |

**Table 9-30. XBARCLR1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 12 | CMPSS7_CTRIPL | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS7_CTRIPL bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 11 | CMPSS6_CTRIPH | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS6_CTRIPH bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 10 | CMPSS6_CTRIPL | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS6_CTRIPL bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 9 | CMPSS5_CTRIPH | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS5_CTRIPH bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 8 | CMPSS5_CTRIPL | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS5_CTRIPL bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 7 | CMPSS4_CTRIPH | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS4_CTRIPH bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 6 | CMPSS4_CTRIPL | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS4_CTRIPL bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 5 | CMPSS3_CTRIPH | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS3_CTRIPH bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 4 | CMPSS3_CTRIPL | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS3_CTRIPL bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 3 | CMPSS2_CTRIPH | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS2_CTRIPH bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 2 | CMPSS2_CTRIPL | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS2_CTRIPL bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 1 | CMPSS1_CTRIPH | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS1_CTRIPH bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 0 | CMPSS1_CTRIPL | R=0/W=1 | 0h | Writing 1 to this bit clears the CMPSS1_CTRIPL bit in the XBARFLG1 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.2.6 XBARCLR2 Register (Offset = Ah) [reset = 0h]

XBARCLR2 is shown in Figure 9-27 and described in Table 9-31.

Return to 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 9-27. XBARCLR2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| ADCCEVT1 | ADCBEVT4 | ADCBEVT3 | ADCBEVT2 | ADCBEVT1 | ADCAEVT4 | ADCAEVT3 | ADCAEVT2 |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ADCAEVT1 | EXTSYNCOUT | ECAP6_OUT | ECAP5_OUT | ECAP4_OUT | ECAP3_OUT | ECAP2_OUT | ECAP1_OUT |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| INPUT14 | INPUT13 | INPUT12 | INPUT11 | INPUT10 | INPUT9 | INPUT8 | INPUT7 |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ADCSOCB | ADCSOCA | INPUT6 | INPUT5 | INPUT4 | INPUT3 | INPUT2 | INPUT1 |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 9-31. XBARCLR2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | ADCCEVT1 | R=0/W=1 | 0h | Writing 1 to this bit clears the ADCCEVT1 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 30 | ADCBEVT4 | R=0/W=1 | 0h | Writing 1 to this bit clears the ADCBEVT4 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 29 | ADCBEVT3 | R=0/W=1 | 0h | Writing 1 to this bit clears the ADCBEVT3 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 28 | ADCBEVT2 | R=0/W=1 | 0h | Writing 1 to this bit clears the ADCBEVT2 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 27 | ADCBEVT1 | R=0/W=1 | 0h | Writing 1 to this bit clears the ADCBEVT1 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 26 | ADCAEVT4 | R=0/W=1 | 0h | Writing 1 to this bit clears the ADCAEVT4 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 25 | ADCAEVT3 | R=0/W=1 | 0h | Writing 1 to this bit clears the ADCAEVT3 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |

**Table 9-31. XBARCLR2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24 | ADCAEVT2 | R=0/W=1 | 0h | Writing 1 to this bit clears the ADCAEVT2 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 23 | ADCAEVT1 | R=0/W=1 | 0h | Writing 1 to this bit clears the ADCAEVT1 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 22 | EXTSYNCOUT | R=0/W=1 | 0h | Writing 1 to this bit clears the EXTSYNCOUT bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 21 | ECAP6_OUT | R=0/W=1 | 0h | Writing 1 to this bit clears the ECAP6_OUT bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 20 | ECAP5_OUT | R=0/W=1 | 0h | Writing 1 to this bit clears the ECAP5_OUT bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 19 | ECAP4_OUT | R=0/W=1 | 0h | Writing 1 to this bit clears the ECAP4_OUT bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 18 | ECAP3_OUT | R=0/W=1 | 0h | Writing 1 to this bit clears the ECAP3_OUT bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 17 | ECAP2_OUT | R=0/W=1 | 0h | Writing 1 to this bit clears the ECAP2_OUT bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 16 | ECAP1_OUT | R=0/W=1 | 0h | Writing 1 to this bit clears the ECAP1_OUT bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 15 | INPUT14 | R=0/W=1 | 0h | Writing 1 to this bit clears the INPUT14 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 14 | INPUT13 | R=0/W=1 | 0h | Writing 1 to this bit clears the INPUT13 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 13 | INPUT12 | R=0/W=1 | 0h | Writing 1 to this bit clears the INPUT12 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 12 | INPUT11 | R=0/W=1 | 0h | Writing 1 to this bit clears the INPUT11 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |

**Table 9-31. XBARCLR2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | INPUT10 | R=0/W=1 | 0h | Writing 1 to this bit clears the INPUT10 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 10 | INPUT9 | R=0/W=1 | 0h | Writing 1 to this bit clears the INPUT9 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 9 | INPUT8 | R=0/W=1 | 0h | Writing 1 to this bit clears the INPUT8 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 8 | INPUT7 | R=0/W=1 | 0h | Writing 1 to this bit clears the INPUT7 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 7 | ADCSOCB | R=0/W=1 | 0h | Writing 1 to this bit clears the ADCSOCB bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 6 | ADCSOCA | R=0/W=1 | 0h | Writing 1 to this bit clears the ADCSOCA bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 5 | INPUT6 | R=0/W=1 | 0h | Writing 1 to this bit clears the INPUT6 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 4 | INPUT5 | R=0/W=1 | 0h | Writing 1 to this bit clears the INPUT5 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 3 | INPUT4 | R=0/W=1 | 0h | Writing 1 to this bit clears the INPUT4 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 2 | INPUT3 | R=0/W=1 | 0h | Writing 1 to this bit clears the INPUT3 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 1 | INPUT2 | R=0/W=1 | 0h | Writing 1 to this bit clears the INPUT2 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 0 | INPUT1 | R=0/W=1 | 0h | Writing 1 to this bit clears the INPUT1 bit in the XBARFLG2 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.2.7 XBARCLR3 Register (Offset = Ch) [reset = 0h]

XBARCLR3 is shown in Figure 9-28 and described in Table 9-32.

Return to 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 9-28. XBARCLR3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| SD1FLT4_DRINT | SD1FLT4_COMPZ | SD1FLT3_DRINT | SD1FLT3_COMPZ | SD1FLT2_DRINT | SD1FLT2_COMPZ | SD1FLT1_DRINT | SD1FLT1_COMPZ |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ECAP7_OUT | SD2FLT4_COMPH | SD2FLT4_COMPL | SD2FLT3_COMPH | SD2FLT3_COMPL | SD2FLT2_COMPH | SD2FLT2_COMPL | SD2FLT1_COMPH |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SD2FLT1_COMPL | SD1FLT4_COMPH | SD1FLT4_COMPL | SD1FLT3_COMPH | SD1FLT3_COMPL | SD1FLT2_COMPH | SD1FLT2_COMPL | SD1FLT1_COMPH |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SD1FLT1_COMPL | | | | | ADCCEVT4 | ADCCEVT3 | ADCCEVT2 |
| R=0/W=1-0h | | | | | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 9-32. XBARCLR3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | SD1FLT4_DRINT | R=0/W=1 | 0h | Writing 1 to this bit clears the SD1FLT4_DRINT bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 30 | SD1FLT4_COMPZ | R=0/W=1 | 0h | Writing 1 to this bit clears the SD1FLT4_COMPZ bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 29 | SD1FLT3_DRINT | R=0/W=1 | 0h | Writing 1 to this bit clears the SD1FLT3_DRINT bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 28 | SD1FLT3_COMPZ | R=0/W=1 | 0h | Writing 1 to this bit clears the SD1FLT3_COMPZ bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 27 | SD1FLT2_DRINT | R=0/W=1 | 0h | Writing 1 to this bit clears the SD1FLT2_DRINT bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 26 | SD1FLT2_COMPZ | R=0/W=1 | 0h | Writing 1 to this bit clears the SD1FLT2_COMPZ bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |

**Table 9-32. XBARCLR3 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 25 | SD1FLT1_DRINT | R=0/W=1 | 0h | Writing 1 to this bit clears the SD1FLT1_DRINT bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 24 | SD1FLT1_COMPZ | R=0/W=1 | 0h | Writing 1 to this bit clears the SD1FLT1_COMPZ bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 23 | ECAP7_OUT | R=0/W=1 | 0h | Writing 1 to this bit clears the ECAP7_OUT bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 22 | SD2FLT4_COMPH | R=0/W=1 | 0h | Writing 1 to this bit clears the SD2FLT4_COMPH bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 21 | SD2FLT4_COMPL | R=0/W=1 | 0h | Writing 1 to this bit clears the SD2FLT4_COMPL bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 20 | SD2FLT3_COMPH | R=0/W=1 | 0h | Writing 1 to this bit clears the SD2FLT3_COMPH bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 19 | SD2FLT3_COMPL | R=0/W=1 | 0h | Writing 1 to this bit clears the SD2FLT3_COMPL bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 18 | SD2FLT2_COMPH | R=0/W=1 | 0h | Writing 1 to this bit clears the SD2FLT2_COMPH bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 17 | SD2FLT2_COMPL | R=0/W=1 | 0h | Writing 1 to this bit clears the SD2FLT2_COMPL bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 16 | SD2FLT1_COMPH | R=0/W=1 | 0h | Writing 1 to this bit clears the SD2FLT1_COMPH bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 15 | SD2FLT1_COMPL | R=0/W=1 | 0h | Writing 1 to this bit clears the SD2FLT1_COMPL bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 14 | SD1FLT4_COMPH | R=0/W=1 | 0h | Writing 1 to this bit clears the SD1FLT4_COMPH bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 13 | SD1FLT4_COMPL | R=0/W=1 | 0h | Writing 1 to this bit clears the SD1FLT4_COMPL bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |

**Table 9-32. XBARCLR3 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 12 | SD1FLT3_COMPH | R=0/W=1 | 0h | Writing 1 to this bit clears the SD1FLT3_COMPH bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 11 | SD1FLT3_COMPL | R=0/W=1 | 0h | Writing 1 to this bit clears the SD1FLT3_COMPL bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 10 | SD1FLT2_COMPH | R=0/W=1 | 0h | Writing 1 to this bit clears the SD1FLT2_COMPH bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 9 | SD1FLT2_COMPL | R=0/W=1 | 0h | Writing 1 to this bit clears the SD1FLT2_COMPL bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 8 | SD1FLT1_COMPH | R=0/W=1 | 0h | Writing 1 to this bit clears the SD1FLT1_COMPH bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 7 | SD1FLT1_COMPL | R=0/W=1 | 0h | Writing 1 to this bit clears the SD1FLT1_COMPL bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | ADCCEVT4 | R=0/W=1 | 0h | Writing 1 to this bit clears the ADCCEVT4 bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 1 | ADCCEVT3 | R=0/W=1 | 0h | Writing 1 to this bit clears the ADCCEVT3 bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 0 | ADCCEVT2 | R=0/W=1 | 0h | Writing 1 to this bit clears the ADCCEVT2 bit in the XBARFLG3 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |

#### 9.3.1.2.8 XBARCLR4 Register (Offset = Eh) [reset = 0h]

XBARCLR4 is shown in Figure 9-29 and described in Table 9-33.

Return to Summary Table.

This register is used to clear the flag(s) in the XBARFLG4 register.
1: Clears the corresponding bit in the XBARFLG4 register.
0: Writing 0 has no effect

**Figure 9-29. XBARCLR4 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CLAHALT | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SD2FLT4_DRINT | SD2FLT4_COMPZ | SD2FLT3_DRINT | SD2FLT3_COMPZ | SD2FLT2_DRINT | SD2FLT2_COMPZ | SD2FLT1_DRINT | SD2FLT1_COMPZ |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 9-33. XBARCLR4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | CLAHALT | R | 0h | Writing 1 to this bit clears the CLAHALT bit in the XBARFLG4 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 30 | RESERVED | R | 0h | Reserved |
| 29 | RESERVED | R | 0h | Reserved |
| 28 | RESERVED | R | 0h | Reserved |
| 27 | RESERVED | R | 0h | Reserved |
| 26 | RESERVED | R | 0h | Reserved |
| 25 | RESERVED | R | 0h | Reserved |
| 24 | RESERVED | R | 0h | Reserved |
| 23 | RESERVED | R | 0h | Reserved |
| 22 | RESERVED | R | 0h | Reserved |
| 21 | RESERVED | R | 0h | Reserved |
| 20 | RESERVED | R | 0h | Reserved |
| 19 | RESERVED | R | 0h | Reserved |
| 18 | RESERVED | R | 0h | Reserved |
| 17 | RESERVED | R | 0h | Reserved |
| 16 | RESERVED | R | 0h | Reserved |
| 15 | RESERVED | R | 0h | Reserved |
| 14 | RESERVED | R | 0h | Reserved |
| 13 | RESERVED | R | 0h | Reserved |
| 12 | RESERVED | R | 0h | Reserved |
| 11 | RESERVED | R | 0h | Reserved |
| 10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |

**Table 9-33. XBARCLR4 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7 | SD2FLT4_DRINT | R=0/W=1 | 0h | Writing 1 to this bit clears the SD2FLT4_DRINT bit in the XBARFLG4 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 6 | SD2FLT4_COMPZ | R=0/W=1 | 0h | Writing 1 to this bit clears the SD2FLT4_COMPZ bit in the XBARFLG4 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 5 | SD2FLT3_DRINT | R=0/W=1 | 0h | Writing 1 to this bit clears the SD2FLT3_DRINT bit in the XBARFLG4 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 4 | SD2FLT3_COMPZ | R=0/W=1 | 0h | Writing 1 to this bit clears the SD2FLT3_COMPZ bit in the XBARFLG4 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 3 | SD2FLT2_DRINT | R=0/W=1 | 0h | Writing 1 to this bit clears the SD2FLT2_DRINT bit in the XBARFLG4 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 2 | SD2FLT2_COMPZ | R=0/W=1 | 0h | Writing 1 to this bit clears the SD2FLT2_COMPZ bit in the XBARFLG4 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 1 | SD2FLT1_DRINT | R=0/W=1 | 0h | Writing 1 to this bit clears the SD2FLT1_DRINT bit in the XBARFLG4 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |
| 0 | SD2FLT1_COMPZ | R=0/W=1 | 0h | Writing 1 to this bit clears the SD2FLT1_COMPZ bit in the XBARFLG4 register.<br>Writing 0 has no effect<br>Reset type: CPU1.SYSRSn |

### 9.3.1.3 EPWM_XBAR_REGS Registers

Table 9-34 lists the memory-mapped registers for the EPWM_XBAR_REGS. All register offset addresses not listed in Table 9-34 should be considered as reserved locations and the register contents should not be modified.

**Table 9-34. EPWM_XBAR_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | TRIP4MUX0TO15CFG | ePWM XBAR Mux Configuration for TRIP4 | EALLOW | 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 9-35 shows the codes that are used for access types in this section.

**Table 9-35. EPWM_XBAR_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| WSOnce | SOnce<br>W | Set once<br>Write |
| **Reset or Default Value** | | |
| *-n* | | Value after reset or the default value |
| **Register Array Variables** | | |

**Table 9-35. EPWM_XBAR_REGS Access Type Codes (continued)**

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

#### 9.3.1.3.1 TRIP4MUX0TO15CFG Register (Offset = 0h) [reset = 0h]

TRIP4MUX0TO15CFG is shown in Figure 9-30 and described in Table 9-36.

Return to Summary Table.

ePWM XBAR Mux Configuration for TRIP4

**Figure 9-30. TRIP4MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-36. TRIP4MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux15:<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 |

**Table 9-36. TRIP4MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-36. TRIP4MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.2 TRIP4MUX16TO31CFG Register (Offset = 2h) [reset = 0h]

TRIP4MUX16TO31CFG is shown in Figure 9-31 and described in Table 9-37.

Return to Summary Table.

ePWM XBAR Mux Configuration for TRIP4

**Figure 9-31. TRIP4MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-37. TRIP4MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for EPWM-XBAR TRIP4 Mux31:<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 |

**Table 9-37. TRIP4MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-37. TRIP4MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.3 TRIP5MUX0TO15CFG Register (Offset = 4h) [reset = 0h]

TRIP5MUX0TO15CFG is shown in Figure 9-32 and described in Table 9-38.

Return to Summary Table.

ePWM XBAR Mux Configuration for TRIP5

**Figure 9-32. TRIP5MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-38. TRIP5MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux15:<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 |

**Table 9-38. TRIP5MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-38. TRIP5MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.4 TRIP5MUX16TO31CFG Register (Offset = 6h) [reset = 0h]

TRIP5MUX16TO31CFG is shown in Figure 9-33 and described in Table 9-39.

Return to Summary Table.

ePWM XBAR Mux Configuration for TRIP5

**Figure 9-33. TRIP5MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-39. TRIP5MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for EPWM-XBAR TRIP5 Mux31:<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 |

**Table 9-39. TRIP5MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-39. TRIP5MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.5 TRIP7MUX0TO15CFG Register (Offset = 8h) [reset = 0h]

TRIP7MUX0TO15CFG is shown in Figure 9-34 and described in Table 9-40.

Return to Summary Table.

ePWM XBAR Mux Configuration for TRIP7

**Figure 9-34. TRIP7MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-40. TRIP7MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux15:<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 |

**Table 9-40. TRIP7MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-40. TRIP7MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.6 TRIP7MUX16TO31CFG Register (Offset = Ah) [reset = 0h]

TRIP7MUX16TO31CFG is shown in Figure 9-35 and described in Table 9-41.

Return to Summary Table.

ePWM XBAR Mux Configuration for TRIP7

**Figure 9-35. TRIP7MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-41. TRIP7MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for EPWM-XBAR TRIP7 Mux31:<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 |

**Table 9-41. TRIP7MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-41. TRIP7MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.7 TRIP8MUX0TO15CFG Register (Offset = Ch) [reset = 0h]

TRIP8MUX0TO15CFG is shown in Figure 9-36 and described in Table 9-42.

Return to Summary Table.

ePWM XBAR Mux Configuration for TRIP8

**Figure 9-36. TRIP8MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-42. TRIP8MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux15:<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 |

**Table 9-42. TRIP8MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-42. TRIP8MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.8 TRIP8MUX16TO31CFG Register (Offset = Eh) [reset = 0h]

TRIP8MUX16TO31CFG is shown in Figure 9-37 and described in Table 9-43.

Return to Summary Table.

ePWM XBAR Mux Configuration for TRIP8

**Figure 9-37. TRIP8MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-43. TRIP8MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for EPWM-XBAR TRIP8 Mux31:<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 |

**Table 9-43. TRIP8MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-43. TRIP8MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.9 TRIP9MUX0TO15CFG Register (Offset = 10h) [reset = 0h]

TRIP9MUX0TO15CFG is shown in Figure 9-38 and described in Table 9-44.

Return to Summary Table.

ePWM XBAR Mux Configuration for TRIP9

**Figure 9-38. TRIP9MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-44. TRIP9MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux15:<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 |

**Table 9-44. TRIP9MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-44. TRIP9MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.10 TRIP9MUX16TO31CFG Register (Offset = 12h) [reset = 0h]

TRIP9MUX16TO31CFG is shown in Figure 9-39 and described in Table 9-45.

Return to Summary Table.

ePWM XBAR Mux Configuration for TRIP9

**Figure 9-39. TRIP9MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-45. TRIP9MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for EPWM-XBAR TRIP9 Mux31:<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 |

**Table 9-45. TRIP9MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-45. TRIP9MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.11 TRIP10MUX0TO15CFG Register (Offset = 14h) [reset = 0h]

TRIP10MUX0TO15CFG is shown in Figure 9-40 and described in Table 9-46.

Return to Summary Table.

ePWM XBAR Mux Configuration for TRIP10

**Figure 9-40. TRIP10MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-46. TRIP10MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux15:<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 |

**Table 9-46. TRIP10MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-46. TRIP10MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.12 TRIP10MUX16TO31CFG Register (Offset = 16h) [reset = 0h]

TRIP10MUX16TO31CFG is shown in Figure 9-41 and described in Table 9-47.

Return to Summary Table.

ePWM XBAR Mux Configuration for TRIP10

**Figure 9-41. TRIP10MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-47. TRIP10MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for EPWM-XBAR TRIP10 Mux31:<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 |

**Table 9-47. TRIP10MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-47. TRIP10MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.13 TRIP11MUX0TO15CFG Register (Offset = 18h) [reset = 0h]

TRIP11MUX0TO15CFG is shown in Figure 9-42 and described in Table 9-48.

Return to Summary Table.

ePWM XBAR Mux Configuration for TRIP11

**Figure 9-42. TRIP11MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-48. TRIP11MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux15:<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 |

**Table 9-48. TRIP11MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-48. TRIP11MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.14 TRIP11MUX16TO31CFG Register (Offset = 1Ah) [reset = 0h]

TRIP11MUX16TO31CFG is shown in Figure 9-43 and described in Table 9-49.

Return to Summary Table.

ePWM XBAR Mux Configuration for TRIP11

**Figure 9-43. TRIP11MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-49. TRIP11MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for EPWM-XBAR TRIP11 Mux31:<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 |

**Table 9-49. TRIP11MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-49. TRIP11MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.15 TRIP12MUX0TO15CFG Register (Offset = 1Ch) [reset = 0h]

TRIP12MUX0TO15CFG is shown in Figure 9-44 and described in Table 9-50.

Return to Summary Table.

ePWM XBAR Mux Configuration for TRIP12

**Figure 9-44. TRIP12MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-50. TRIP12MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux15:<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 |

**Table 9-50. TRIP12MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-50. TRIP12MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.16 TRIP12MUX16TO31CFG Register (Offset = 1Eh) [reset = 0h]

TRIP12MUX16TO31CFG is shown in Figure 9-45 and described in Table 9-51.

Return to Summary Table.

ePWM XBAR Mux Configuration for TRIP12

**Figure 9-45. TRIP12MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-51. TRIP12MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for EPWM-XBAR TRIP12 Mux31:<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 |

**Table 9-51. TRIP12MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-51. TRIP12MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.17 TRIP4MUXENABLE Register (Offset = 20h) [reset = 0h]

TRIP4MUXENABLE is shown in Figure 9-46 and described in Table 9-52.

Return to Summary Table.

ePWM XBAR Mux Enable for TRIP4

**Figure 9-46. TRIP4MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | 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. 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 enabled 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 enabled 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 enabled 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 enabled 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 |

**Table 9-52. TRIP4MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of Mux27 to drive TRIP4 of EPWM-XBAR<br>0: Respective output of Mux27 is enabled 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 enabled 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 enabled 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 enabled 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 |

**Table 9-52. TRIP4MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-52. TRIP4MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-52. TRIP4MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.18 TRIP5MUXENABLE Register (Offset = 22h) [reset = 0h]

TRIP5MUXENABLE is shown in Figure 9-47 and described in Table 9-53.

Return to Summary Table.

ePWM XBAR Mux Enable for TRIP5

**Figure 9-47. TRIP5MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | 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. 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 enabled 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 enabled 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 enabled 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 enabled 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 |

**Table 9-53. TRIP5MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of Mux27 to drive TRIP5 of EPWM-XBAR<br>0: Respective output of Mux27 is enabled 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 enabled 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 enabled 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 enabled 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 |

**Table 9-53. TRIP5MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-53. TRIP5MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-53. TRIP5MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.19 TRIP7MUXENABLE Register (Offset = 24h) [reset = 0h]

TRIP7MUXENABLE is shown in Figure 9-48 and described in Table 9-54.

Return to Summary Table.

ePWM XBAR Mux Enable for TRIP7

**Figure 9-48. TRIP7MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | 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. 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 enabled 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 enabled 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 enabled 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 enabled 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 |

**Table 9-54. TRIP7MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of Mux27 to drive TRIP7 of EPWM-XBAR<br>0: Respective output of Mux27 is enabled 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 enabled 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 enabled 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 enabled 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 |

**Table 9-54. TRIP7MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-54. TRIP7MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-54. TRIP7MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.20 TRIP8MUXENABLE Register (Offset = 26h) [reset = 0h]

TRIP8MUXENABLE is shown in Figure 9-49 and described in Table 9-55.

Return to Summary Table.

ePWM XBAR Mux Enable for TRIP8

**Figure 9-49. TRIP8MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | 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. 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 enabled 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 enabled 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 enabled 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 enabled 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 |

**Table 9-55. TRIP8MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of Mux27 to drive TRIP8 of EPWM-XBAR<br>0: Respective output of Mux27 is enabled 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 enabled 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 enabled 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 enabled 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 |

**Table 9-55. TRIP8MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-55. TRIP8MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-55. TRIP8MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.21 TRIP9MUXENABLE Register (Offset = 28h) [reset = 0h]

TRIP9MUXENABLE is shown in Figure 9-50 and described in Table 9-56.

Return to Summary Table.

ePWM XBAR Mux Enable for TRIP9

**Figure 9-50. TRIP9MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | 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. 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 enabled 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 enabled 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 enabled 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 enabled 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 |

**Table 9-56. TRIP9MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of Mux27 to drive TRIP9 of EPWM-XBAR<br>0: Respective output of Mux27 is enabled 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 enabled 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 enabled 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 enabled 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 |

**Table 9-56. TRIP9MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-56. TRIP9MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-56. TRIP9MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.22 TRIP10MUXENABLE Register (Offset = 2Ah) [reset = 0h]

TRIP10MUXENABLE is shown in Figure 9-51 and described in Table 9-57.

Return to Summary Table.

ePWM XBAR Mux Enable for TRIP10

**Figure 9-51. TRIP10MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-57. 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 enabled 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 enabled 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 enabled 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 enabled 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 |

**Table 9-57. TRIP10MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of Mux27 to drive TRIP10 of EPWM-XBAR<br>0: Respective output of Mux27 is enabled 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 enabled 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 enabled 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 enabled 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 |

**Table 9-57. TRIP10MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-57. TRIP10MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-57. TRIP10MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.23 TRIP11MUXENABLE Register (Offset = 2Ch) [reset = 0h]

TRIP11MUXENABLE is shown in Figure 9-52 and described in Table 9-58.

Return to Summary Table.

ePWM XBAR Mux Enable for TRIP11

**Figure 9-52. TRIP11MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | 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 9-58. TRIP11MUXENABLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MUX31 | R/W | 0h | Selects the output of Mux31 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux31 is enabled 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 enabled 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 enabled 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 enabled 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 |

**Table 9-58. TRIP11MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of Mux27 to drive TRIP11 of EPWM-XBAR<br>0: Respective output of Mux27 is enabled 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 enabled 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 enabled 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 enabled 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 |

**Table 9-58. TRIP11MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-58. TRIP11MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-58. TRIP11MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.24 TRIP12MUXENABLE Register (Offset = 2Eh) [reset = 0h]

TRIP12MUXENABLE is shown in Figure 9-53 and described in Table 9-59.

Return to Summary Table.

ePWM XBAR Mux Enable for TRIP12

**Figure 9-53. TRIP12MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-59. 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 enabled 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 enabled 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 enabled 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 enabled 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 |

**Table 9-59. TRIP12MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of Mux27 to drive TRIP12 of EPWM-XBAR<br>0: Respective output of Mux27 is enabled 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 enabled 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 enabled 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 enabled 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 |

**Table 9-59. TRIP12MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-59. TRIP12MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-59. TRIP12MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.25 TRIPOUTINV Register (Offset = 38h) [reset = 0h]

TRIPOUTINV is shown in Figure 9-54 and described in Table 9-60.

Return to Summary Table.

ePWM XBAR Output Inversion Register

**Figure 9-54. TRIPOUTINV Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TRIP12 | TRIP11 | TRIP10 | TRIP9 | TRIP8 | TRIP7 | TRIP5 | TRIP4 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-60. TRIPOUTINV Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R=0 | 0h | Reserved |
| 15-8 | RESERVED | R=0 | 0h | Reserved |
| 7 | TRIP12 | R/W | 0h | Selects polarity for TRIP12 of EPWM-XBAR<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 |

**Table 9-60. TRIPOUTINV Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.3.26 TRIPLOCK Register (Offset = 3Eh) [reset = 0h]

TRIPLOCK is shown in Figure 9-55 and described in Table 9-61.

Return to Summary Table.

ePWM XBAR Configuration Lock register

**Figure 9-55. TRIPLOCK Register**

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

**Table 9-61. TRIPLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | KEY | R=0/W | 0h | Bit-0 of this register can be set only if KEY= 0x5a5a<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 |

### 9.3.1.4 OUTPUT_XBAR_REGS Registers

Table 9-62 lists the memory-mapped registers for the OUTPUT_XBAR_REGS. All register offset addresses not listed in Table 9-62 should be considered as reserved locations and the register contents should not be modified.

**Table 9-62. OUTPUT_XBAR_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | OUTPUT1MUX0TO15CFG | Output X-BAR Mux Configuration for Output 1 | EALLOW | 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 9-63 shows the codes that are used for access types in this section.

**Table 9-63. OUTPUT_XBAR_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| W=1 | W | Write |

**Table 9-63. OUTPUT_XBAR_REGS Access Type Codes (continued)**

| Access Type | Code | Description |
|---|---|---|
| WSOnce | SOnce<br>W | Set once<br>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. |

#### 9.3.1.4.1 OUTPUT1MUX0TO15CFG Register (Offset = 0h) [reset = 0h]

OUTPUT1MUX0TO15CFG is shown in Figure 9-56 and described in Table 9-64.

Return to Summary Table.

Output X-BAR Mux Configuration for Output 1

**Figure 9-56. OUTPUT1MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-64. OUTPUT1MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for OUTPUT1 Mux15:<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 |

**Table 9-64. OUTPUT1MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-64. OUTPUT1MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.2 OUTPUT1MUX16TO31CFG Register (Offset = 2h) [reset = 0h]

OUTPUT1MUX16TO31CFG is shown in Figure 9-57 and described in Table 9-65.

Return to Summary Table.

Output X-BAR Mux Configuration for Output 1

**Figure 9-57. OUTPUT1MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-65. OUTPUT1MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for OUTPUT1 Mux31:<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 |

**Table 9-65. OUTPUT1MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-65. OUTPUT1MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.3 OUTPUT2MUX0TO15CFG Register (Offset = 4h) [reset = 0h]

OUTPUT2MUX0TO15CFG is shown in Figure 9-58 and described in Table 9-66.

Return to Summary Table.

Output X-BAR Mux Configuration for Output 2

**Figure 9-58. OUTPUT2MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-66. OUTPUT2MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for OUTPUT2 Mux15:<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 |

**Table 9-66. OUTPUT2MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-66. OUTPUT2MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.4 OUTPUT2MUX16TO31CFG Register (Offset = 6h) [reset = 0h]

OUTPUT2MUX16TO31CFG is shown in Figure 9-59 and described in Table 9-67.

Return to Summary Table.

Output X-BAR Mux Configuration for Output 2

**Figure 9-59. OUTPUT2MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-67. OUTPUT2MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for OUTPUT2 Mux31:<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 |

**Table 9-67. OUTPUT2MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-67. OUTPUT2MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.5 OUTPUT3MUX0TO15CFG Register (Offset = 8h) [reset = 0h]

OUTPUT3MUX0TO15CFG is shown in Figure 9-60 and described in Table 9-68.

Return to Summary Table.

Output X-BAR Mux Configuration for Output 3

**Figure 9-60. OUTPUT3MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-68. OUTPUT3MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for OUTPUT3 Mux15:<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 |

**Table 9-68. OUTPUT3MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-68. OUTPUT3MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.6 OUTPUT3MUX16TO31CFG Register (Offset = Ah) [reset = 0h]

OUTPUT3MUX16TO31CFG is shown in Figure 9-61 and described in Table 9-69.

Return to Summary Table.

Output X-BAR Mux Configuration for Output 3

**Figure 9-61. OUTPUT3MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-69. OUTPUT3MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for OUTPUT3 Mux31:<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 |

**Table 9-69. OUTPUT3MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-69. OUTPUT3MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.7 OUTPUT4MUX0TO15CFG Register (Offset = Ch) [reset = 0h]

OUTPUT4MUX0TO15CFG is shown in Figure 9-62 and described in Table 9-70.

Return to Summary Table.

Output X-BAR Mux Configuration for Output 4

**Figure 9-62. OUTPUT4MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-70. OUTPUT4MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for OUTPUT4 Mux15:<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 |

**Table 9-70. OUTPUT4MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-70. OUTPUT4MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.8 OUTPUT4MUX16TO31CFG Register (Offset = Eh) [reset = 0h]

OUTPUT4MUX16TO31CFG is shown in Figure 9-63 and described in Table 9-71.

Return to Summary Table.

Output X-BAR Mux Configuration for Output 4

**Figure 9-63. OUTPUT4MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-71. OUTPUT4MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for OUTPUT4 Mux31:<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 |

**Table 9-71. OUTPUT4MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-71. OUTPUT4MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.9 OUTPUT5MUX0TO15CFG Register (Offset = 10h) [reset = 0h]

OUTPUT5MUX0TO15CFG is shown in Figure 9-64 and described in Table 9-72.

Return to Summary Table.

Output X-BAR Mux Configuration for Output 5

**Figure 9-64. OUTPUT5MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-72. OUTPUT5MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for OUTPUT5 Mux15:<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 |

**Table 9-72. OUTPUT5MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-72. OUTPUT5MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.10 OUTPUT5MUX16TO31CFG Register (Offset = 12h) [reset = 0h]

OUTPUT5MUX16TO31CFG is shown in Figure 9-65 and described in Table 9-73.

Return to Summary Table.

Output X-BAR Mux Configuration for Output 5

**Figure 9-65. OUTPUT5MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-73. OUTPUT5MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for OUTPUT5 Mux31:<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 |

**Table 9-73. OUTPUT5MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-73. OUTPUT5MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.11 OUTPUT6MUX0TO15CFG Register (Offset = 14h) [reset = 0h]

OUTPUT6MUX0TO15CFG is shown in Figure 9-66 and described in Table 9-74.

Return to Summary Table.

Output X-BAR Mux Configuration for Output 6

**Figure 9-66. OUTPUT6MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-74. OUTPUT6MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for OUTPUT6 Mux15:<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 |

**Table 9-74. OUTPUT6MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-74. OUTPUT6MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.12 OUTPUT6MUX16TO31CFG Register (Offset = 16h) [reset = 0h]

OUTPUT6MUX16TO31CFG is shown in Figure 9-67 and described in Table 9-75.

Return to Summary Table.

Output X-BAR Mux Configuration for Output 6

**Figure 9-67. OUTPUT6MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-75. OUTPUT6MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for OUTPUT6 Mux31:<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 |

**Table 9-75. OUTPUT6MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-75. OUTPUT6MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.13 OUTPUT7MUX0TO15CFG Register (Offset = 18h) [reset = 0h]

OUTPUT7MUX0TO15CFG is shown in Figure 9-68 and described in Table 9-76.

Return to Summary Table.

Output X-BAR Mux Configuration for Output 7

**Figure 9-68. OUTPUT7MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-76. OUTPUT7MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for OUTPUT7 Mux15:<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 |

**Table 9-76. OUTPUT7MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-76. OUTPUT7MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.14 OUTPUT7MUX16TO31CFG Register (Offset = 1Ah) [reset = 0h]

OUTPUT7MUX16TO31CFG is shown in Figure 9-69 and described in Table 9-77.

Return to Summary Table.

Output X-BAR Mux Configuration for Output 7

**Figure 9-69. OUTPUT7MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-77. OUTPUT7MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for OUTPUT7 Mux31:<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 |

**Table 9-77. OUTPUT7MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-77. OUTPUT7MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.15 OUTPUT8MUX0TO15CFG Register (Offset = 1Ch) [reset = 0h]

OUTPUT8MUX0TO15CFG is shown in Figure 9-70 and described in Table 9-78.

Return to Summary Table.

Output X-BAR Mux Configuration for Output 8

**Figure 9-70. OUTPUT8MUX0TO15CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX15 | | MUX14 | | MUX13 | | MUX12 | | MUX11 | | MUX10 | | MUX9 | | MUX8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | | MUX6 | | MUX5 | | MUX4 | | MUX3 | | MUX2 | | MUX1 | | MUX0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-78. OUTPUT8MUX0TO15CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX15 | R/W | 0h | Select Bits for OUTPUT8 Mux15:<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 |

**Table 9-78. OUTPUT8MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-78. OUTPUT8MUX0TO15CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.16 OUTPUT8MUX16TO31CFG Register (Offset = 1Eh) [reset = 0h]

OUTPUT8MUX16TO31CFG is shown in Figure 9-71 and described in Table 9-79.

Return to Summary Table.

Output X-BAR Mux Configuration for Output 8

**Figure 9-71. OUTPUT8MUX16TO31CFG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MUX31 | | MUX30 | | MUX29 | | MUX28 | | MUX27 | | MUX26 | | MUX25 | | MUX24 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX23 | | MUX22 | | MUX21 | | MUX20 | | MUX19 | | MUX18 | | MUX17 | | MUX16 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 9-79. OUTPUT8MUX16TO31CFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-30 | MUX31 | R/W | 0h | Select Bits for OUTPUT8 Mux31:<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 |

**Table 9-79. OUTPUT8MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-79. OUTPUT8MUX16TO31CFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.17 OUTPUT1MUXENABLE Register (Offset = 20h) [reset = 0h]

OUTPUT1MUXENABLE is shown in Figure 9-72 and described in Table 9-80.

Return to Summary Table.

Output X-BAR Mux Enable for Output 1

**Figure 9-72. OUTPUT1MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | 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. 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 |

**Table 9-80. OUTPUT1MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of Mux27 to drive OUTPUT1 of OUTPUT-XBAR<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 |

**Table 9-80. OUTPUT1MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-80. OUTPUT1MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-80. OUTPUT1MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.18 OUTPUT2MUXENABLE Register (Offset = 22h) [reset = 0h]

OUTPUT2MUXENABLE is shown in Figure 9-73 and described in Table 9-81.

Return to Summary Table.

Output X-BAR Mux Enable for Output 2

**Figure 9-73. OUTPUT2MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | 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. 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 |

**Table 9-81. OUTPUT2MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of Mux27 to drive OUTPUT2 of OUTPUT-XBAR<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 |

**Table 9-81. OUTPUT2MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-81. OUTPUT2MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-81. OUTPUT2MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.19 OUTPUT3MUXENABLE Register (Offset = 24h) [reset = 0h]

OUTPUT3MUXENABLE is shown in Figure 9-74 and described in Table 9-82.

Return to Summary Table.

Output X-BAR Mux Enable for Output 3

**Figure 9-74. OUTPUT3MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | 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. 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 |

**Table 9-82. OUTPUT3MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of Mux27 to drive OUTPUT3 of OUTPUT-XBAR<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 |

**Table 9-82. OUTPUT3MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-82. OUTPUT3MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-82. OUTPUT3MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.20 OUTPUT4MUXENABLE Register (Offset = 26h) [reset = 0h]

OUTPUT4MUXENABLE is shown in Figure 9-75 and described in Table 9-83.

Return to Summary Table.

Output X-BAR Mux Enable for Output 4

**Figure 9-75. OUTPUT4MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | 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. 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 |

**Table 9-83. OUTPUT4MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of Mux27 to drive OUTPUT4 of OUTPUT-XBAR<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 |

**Table 9-83. OUTPUT4MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-83. OUTPUT4MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-83. OUTPUT4MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.21 OUTPUT5MUXENABLE Register (Offset = 28h) [reset = 0h]

OUTPUT5MUXENABLE is shown in Figure 9-76 and described in Table 9-84.

Return to Summary Table.

Output X-BAR Mux Enable for Output 5

**Figure 9-76. OUTPUT5MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | 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. 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 |

**Table 9-84. OUTPUT5MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of Mux27 to drive OUTPUT5 of OUTPUT-XBAR<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 |

**Table 9-84. OUTPUT5MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-84. OUTPUT5MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-84. OUTPUT5MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.22 OUTPUT6MUXENABLE Register (Offset = 2Ah) [reset = 0h]

OUTPUT6MUXENABLE is shown in Figure 9-77 and described in Table 9-85.

Return to Summary Table.

Output X-BAR Mux Enable for Output 6

**Figure 9-77. OUTPUT6MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MUX23 | MUX22 | MUX21 | MUX20 | MUX19 | MUX18 | MUX17 | MUX16 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MUX15 | MUX14 | MUX13 | MUX12 | MUX11 | MUX10 | MUX9 | MUX8 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MUX7 | MUX6 | MUX5 | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-85. 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 |

**Table 9-85. OUTPUT6MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of Mux27 to drive OUTPUT6 of OUTPUT-XBAR<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 |

**Table 9-85. OUTPUT6MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-85. OUTPUT6MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-85. OUTPUT6MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.23 OUTPUT7MUXENABLE Register (Offset = 2Ch) [reset = 0h]

OUTPUT7MUXENABLE is shown in Figure 9-78 and described in Table 9-86.

Return to Summary Table.

Output X-BAR Mux Enable for Output 7

**Figure 9-78. OUTPUT7MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | 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 9-86. OUTPUT7MUXENABLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MUX31 | R/W | 0h | Selects the output of Mux31 to drive OUTPUT7 of OUTPUT-XBAR<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 |

**Table 9-86. OUTPUT7MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of Mux27 to drive OUTPUT7 of OUTPUT-XBAR<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 |

**Table 9-86. OUTPUT7MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-86. OUTPUT7MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-86. OUTPUT7MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.24 OUTPUT8MUXENABLE Register (Offset = 2Eh) [reset = 0h]

OUTPUT8MUXENABLE is shown in Figure 9-79 and described in Table 9-87.

Return to Summary Table.

Output X-BAR Mux Enable for Output 8

**Figure 9-79. OUTPUT8MUXENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MUX31 | MUX30 | MUX29 | MUX28 | MUX27 | MUX26 | MUX25 | MUX24 |
| R/W-0h | 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 9-87. OUTPUT8MUXENABLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MUX31 | R/W | 0h | Selects the output of Mux31 to drive OUTPUT8 of OUTPUT-XBAR<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 |

**Table 9-87. OUTPUT8MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 27 | MUX27 | R/W | 0h | Selects the output of Mux27 to drive OUTPUT8 of OUTPUT-XBAR<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 |

**Table 9-87. OUTPUT8MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-87. OUTPUT8MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 9-87. OUTPUT8MUXENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.25 OUTPUTLATCH Register (Offset = 30h) [reset = 0h]

OUTPUTLATCH is shown in Figure 9-80 and described in Table 9-88.

Return to Summary Table.

Output X-BAR Output Latch

**Figure 9-80. OUTPUTLATCH Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OUTPUT8 | OUTPUT7 | OUTPUT6 | OUTPUT5 | OUTPUT4 | OUTPUT3 | OUTPUT2 | OUTPUT1 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 9-88. OUTPUTLATCH Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R=0 | 0h | Reserved |
| 15-8 | RESERVED | R=0 | 0h | Reserved |
| 7 | OUTPUT8 | R | 0h | Records the OUTPUT8 of OUTPUT-XBAR.<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 |

**Table 9-88. OUTPUTLATCH Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.26 OUTPUTLATCHCLR Register (Offset = 32h) [reset = 0h]

OUTPUTLATCHCLR is shown in Figure 9-81 and described in Table 9-89.

Return to Summary Table.

Output X-BAR Output Latch Clear

**Figure 9-81. OUTPUTLATCHCLR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OUTPUT8 | OUTPUT7 | OUTPUT6 | OUTPUT5 | OUTPUT4 | OUTPUT3 | OUTPUT2 | OUTPUT1 |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 9-89. OUTPUTLATCHCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R=0 | 0h | Reserved |
| 15-8 | RESERVED | R=0 | 0h | Reserved |
| 7 | OUTPUT8 | R=0/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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 |

**Table 9-89. OUTPUTLATCHCLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | OUTPUT4 | R=0/W=1 | 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 |
| 2 | OUTPUT3 | R=0/W=1 | 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/W=1 | 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/W=1 | 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 |

#### 9.3.1.4.27 OUTPUTLATCHFRC Register (Offset = 34h) [reset = 0h]

OUTPUTLATCHFRC is shown in Figure 9-82 and described in Table 9-90.

Return to Summary Table.

Output X-BAR Output Latch Clear

**Figure 9-82. OUTPUTLATCHFRC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OUTPUT8 | OUTPUT7 | OUTPUT6 | OUTPUT5 | OUTPUT4 | OUTPUT3 | OUTPUT2 | OUTPUT1 |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 9-90. OUTPUTLATCHFRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R=0 | 0h | Reserved |
| 15-8 | RESERVED | R=0 | 0h | Reserved |
| 7 | OUTPUT8 | R=0/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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 |

**Table 9-90. OUTPUTLATCHFRC Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | OUTPUT4 | R=0/W=1 | 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 |
| 2 | OUTPUT3 | R=0/W=1 | 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/W=1 | 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/W=1 | 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 |

#### 9.3.1.4.28 OUTPUTLATCHENABLE Register (Offset = 36h) [reset = 0h]

OUTPUTLATCHENABLE is shown in Figure 9-83 and described in Table 9-91.

Return to Summary Table.

Output X-BAR Output Latch Enable

**Figure 9-83. OUTPUTLATCHENABLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OUTPUT8 | OUTPUT7 | OUTPUT6 | OUTPUT5 | OUTPUT4 | OUTPUT3 | OUTPUT2 | OUTPUT1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-91. OUTPUTLATCHENABLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R=0 | 0h | Reserved |
| 15-8 | RESERVED | R=0 | 0h | Reserved |
| 7 | OUTPUT8 | R/W | 0h | Selects the output latch to drive OUTPUT8 for OUTPUT-XBAR<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 |

**Table 9-91. OUTPUTLATCHENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | OUTPUT4 | R/W | 0h | Selects the output latch to drive OUTPUT4 for OUTPUT-XBAR<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 |
| 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 |

#### 9.3.1.4.29 OUTPUTINV Register (Offset = 38h) [reset = 0h]

OUTPUTINV is shown in Figure 9-84 and described in Table 9-92.

Return to Summary Table.

Output X-BAR Output Inversion

**Figure 9-84. OUTPUTINV Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OUTPUT8 | OUTPUT7 | OUTPUT6 | OUTPUT5 | OUTPUT4 | OUTPUT3 | OUTPUT2 | OUTPUT1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 9-92. OUTPUTINV Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R=0 | 0h | Reserved |
| 15-8 | RESERVED | R=0 | 0h | Reserved |
| 7 | OUTPUT8 | R/W | 0h | Selects polarity for OUTPUT8 of OUTPUT-XBAR<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 |

**Table 9-92. OUTPUTINV Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 9.3.1.4.30 OUTPUTLOCK Register (Offset = 3Eh) [reset = 0h]

OUTPUTLOCK is shown in Figure 9-85 and described in Table 9-93.

Return to Summary Table.

Output X-BAR Configuration Lock register

**Figure 9-85. OUTPUTLOCK Register**

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

**Table 9-93. OUTPUTLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | KEY | R=0/W=1 | 0h | Bit-0 of this register can be set only if KEY= 0x5a5a<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 |

# Direct Memory Access (DMA)

The direct memory access (DMA) module provides a hardware method of transferring data between peripherals and/or memory without intervention from the CPU, thereby freeing up bandwidth for other system functions. Additionally, the DMA has the capability to orthogonally rearrange the data as it is transferred as well as “ping-pong” data between buffers. These features are useful for structuring data into blocks for optimal CPU processing.

**Topic** **Page**

## 10.1 Introduction

The strength of a controller is not measured purely in processor speed, but in total system capabilities. As a part of the equation, any time the CPU bandwidth for a given function can be reduced, the greater the system capabilities. Many times applications spend a significant amount of their bandwidth moving data, whether it is from off-chip memory to on-chip memory, or from a peripheral such as an analog-to-digital converter (ADC) to RAM, or even from one peripheral to another. Furthermore, many times this data comes in a format that is not conducive to the optimum processing powers of the CPU. The DMA module described in this reference guide has the ability to free up CPU bandwidth and rearrange the data into a pattern for more streamlined processing.

The DMA module is an event-based machine, meaning it requires a peripheral or software trigger to start a DMA transfer. Although it can be made into a periodic time-driven machine by configuring a timer as the interrupt trigger source, there is no mechanism within the module itself to start memory transfers periodically. The interrupt trigger source for each of the six DMA channels can be configured separately and each channel contains its own independent PIE interrupt to let the CPU know when a DMA transfer has either started or completed. Five of the six channels are exactly the same, while Channel 1 has the ability to be configured at a higher priority than the others. At the heart of the DMA is a state machine and tightly coupled address control logic. It is this address control logic that allows for rearrangement of the block of data during the transfer as well as the process of ping-ponging data between buffers. Each of these features, along with others, will be discussed in detail in this document.

DMA features include:

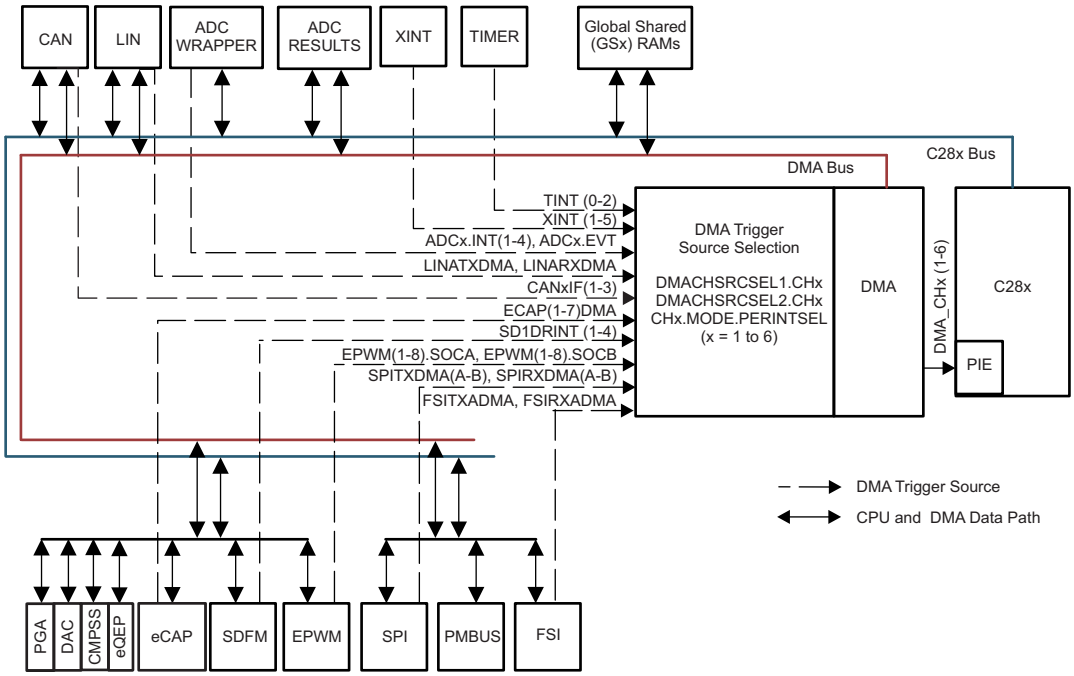
- Six channels with independent PIE interrupts
- Peripheral interrupt trigger sources
  - ADC interrupts and EVT signals
  - External Interrupts
  - ePWM SOC signals
  - CPU Timers
  - eCAP
  - Sigma-Delta Filter Module
  - SPI transmit and receive
  - CAN transmit and receive
  - LIN transmit and receive
  - FSI transmit and receive
- Data sources and destinations:
  - GSx RAM
  - ADC result registers
  - Control peripheral registers (ePWM, eQEP, eCAP, SDFM)
  - DAC and PGA registers
  - SPI, LIN, CAN, PMBus, and FSI registers
- Word Size: 16-bit or 32-bit (SPI limited to 16-bit)
- Throughput: 4 cycles/word without arbitration

## 10.2 Architecture

### 10.2.1 Block Diagram

Figure 10-1 shows a device-level block diagram of the DMA.

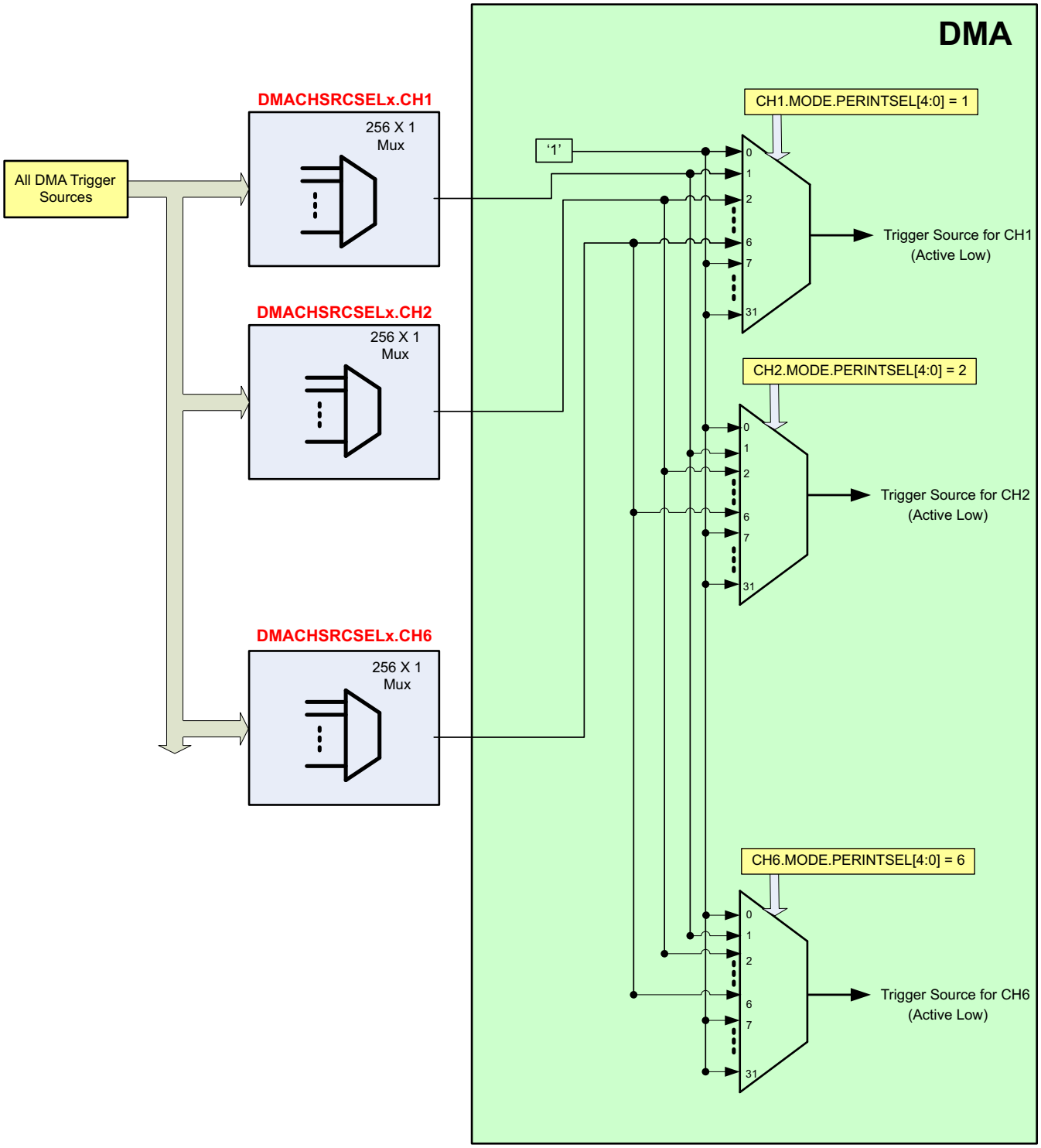
**Figure 10-1. DMA Block Diagram**




### 10.2.2 Peripheral Interrupt Event Trigger Sources

As shown in Figure 10-2 the peripheral interrupt event trigger can be independently configured as any of the sources from the DMACHSRCSELx register for each of the six DMA channels. Included in these sources are five external interrupt signals which can be connected to most of the general-purpose input/output (GPIO) pins on the device. This adds significant flexibility to the event trigger capabilities. The DMA trigger source is selected in the DMACHSRCSELx register for each channel. The PERINTSEL in the MODE register of each channel should be set to the channel number (for example, CH2.MODE.PERINTSEL[4:0] = 2). An active peripheral interrupt trigger will be latched into the PERINTFLG bit of the CONTROL register, and if the respective interrupt and DMA channel is enabled (see the MODE.CHx[PERINTE] and CONTROL.CHx[RUNSTS] bits), it will be serviced by the DMA channel. Upon receipt of a peripheral interrupt event signal, the DMA will automatically send a clear signal to the interrupt source so that subsequent interrupt events will occur.

**Figure 10-2. DMA Trigger Architecture**



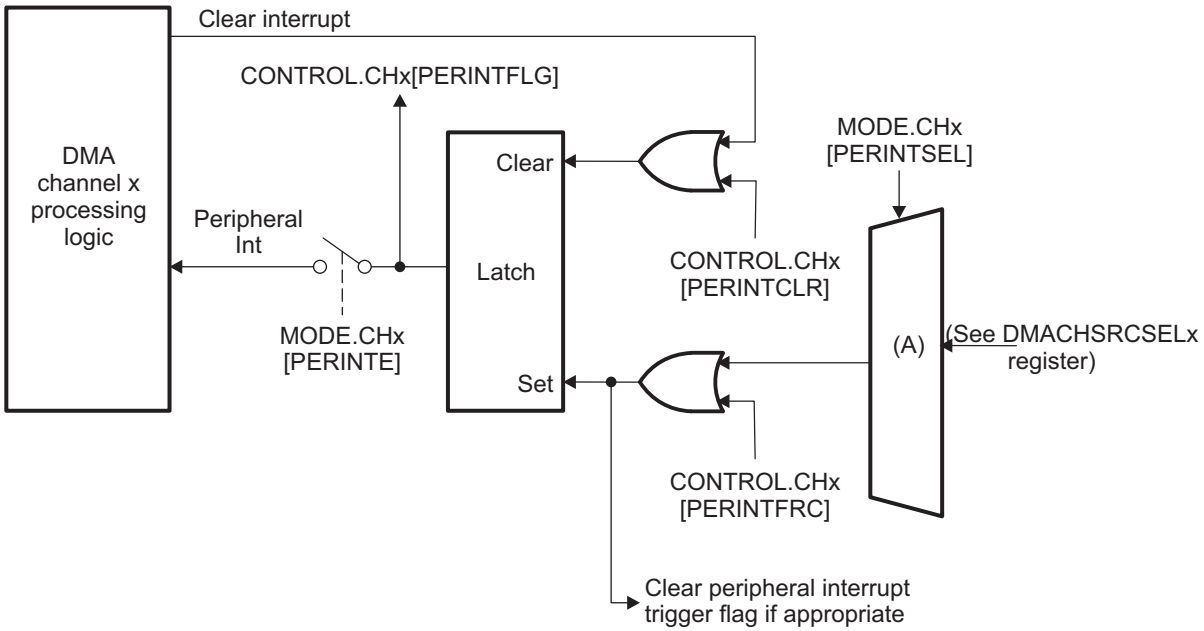

**NOTE:** To use the system level DMA Trigger source selection, the DMA internal trigger source selection configuration for each channel should be done using the DMACHSRCSELx register (), and the CHx.MODE.PERINTSEL register as shown here. See Table 10-1 or the DMACHSRCSELx register definition for a complete list of DMA trigger sources.

Regardless of the value of the MODE.CHx[PERINTSEL] bit field, software can always force a trigger by using the CONTROL.CHx[PERINTFRC] bit. Likewise, software can always clear a pending DMA trigger using the CONTROL.CHx[PERINTCLR] bit.

Once a particular interrupt trigger sets a channel's PERINTFLG bit, the bit stays pending until the priority logic of the state machine starts the burst transfer for that channel. Once the burst transfer starts, the flag is cleared. If a new interrupt trigger is generated while a burst is in progress, the burst will complete before responding to the new interrupt trigger (after proper prioritization). If a third interrupt trigger occurs before the pending interrupt is serviced, an error flag is set in the CONTROL.CHx[OVRFLG] bit. If a peripheral interrupt trigger occurs at the same time as the latched flag is being cleared, the peripheral interrupt trigger has priority and the PERINTFLG will remain set.

Figure 10-3 shows a diagram of the trigger select circuit. See the DMACHSRCSELx register description for the complete list of peripheral interrupt trigger sources.

**Figure 10-3. Peripheral Interrupt Trigger Input Diagram**




A See Figure 10-2.

Table 10-1 shows the interrupt trigger source options that are available for each channel.

**Table 10-1. Peripheral Interrupt Trigger Source Options**

| Select Value (8-bit) | DMA ChTrigger Source |
|---|---|
| 0 | No Peripheral |
| 1 | ADCA.1 |
| 2 | ADCA.2 |
| 3 | ADCA.3 |
| 4 | ADCA.4 |
| 5 | ADCAEVT |
| 6 | ADCB.1 |
| 7 | ADCB.2 |
| 8 | ADCB.3 |
| 9 | ADCB.4 |
| 10 | ADCBEVT |
| 11 | ADCC.1 |
| 12 | ADCC.2 |
| 13 | ADCC.3 |
| 14 | ADCC.4 |
| 15 | ADCCEVT |
| 16 - 28 | No Peripheral |
| 29 | XINT1 |
| 30 | XINT2 |
| 31 | XINT3 |
| 32 | XINT4 |
| 33 | XINT5 |
| 34 - 35 | No Peripheral |
| 36 | EPWM1.SOCA |
| 37 | EPWM1.SOCB |
| 38 | EPWM2.SOCA |
| 39 | EPWM2.SOCB |
| 40 | EPWM3.SOCA |
| 41 | EPWM3.SOCB |
| 42 | EPWM4.SOCA |
| 43 | EPWM4.SOCB |
| 44 | EPWM5.SOCA |
| 45 | EPWM5.SOCB |
| 46 | EPWM6.SOCA |
| 47 | EPWM6.SOCB |
| 48 | EPWM7.SOCA |
| 49 | EPWM7.SOCB |
| 50 | EPWM8.SOCA |
| 51 | EPWM8.SOCB |
| 52 - 67 | No Peripheral |
| 68 | TINT0 |
| 69 | TINT1 |
| 70 | TINT2 |
| 71 - 74 | No Peripheral |
| 75 | ECAP1DMA |
| 76 | ECAP2DMA |
| 77 | ECAP3DMA |
| 78 | ECAP4DMA |

**Table 10-1. Peripheral Interrupt Trigger Source Options (continued)**

| Select Value (8-bit) | DMA ChTrigger Source |
|---|---|
| 79 | ECAP5DMA |
| 80 | ECAP6DMA |
| 81 | ECAP7DMA |
| 82 - 95 | No Peripheral |
| 96 | SD1FLT1 |
| 97 | SD1FLT2 |
| 98 | SD1FLT3 |
| 99 | SD1FLT4 |
| 100 - 108 | No Peripheral |
| 109 | SPITXDMAA |
| 110 | SPIRXDMAA |
| 111 | SPITXDMAB |
| 112 | SPIRXDMAB |
| 113 - 116 | No Peripheral |
| 117 | LINATXDMA |
| 118 | LINARXDMA |
| 119 - 122 | No Peripheral |
| 123 | FSITXADMA |
| 124 | No Peripheral |
| 125 | FSIRXADMA |
| 126 | No Peripheral |
| 127 | Reserved |
| 128 | Reserved |
| 129 | Reserved |
| 130 | Reserved |
| 131 - 166 | No Peripheral |
| 167 | CANAIF1 |
| 168 | CANAIF2 |
| 169 | CANAIF3 |
| 170 | CANBIF1 |
| 171 | CANBIF2 |
| 172 | CANBIF3 |
| 173 - 255 | No Peripheral |

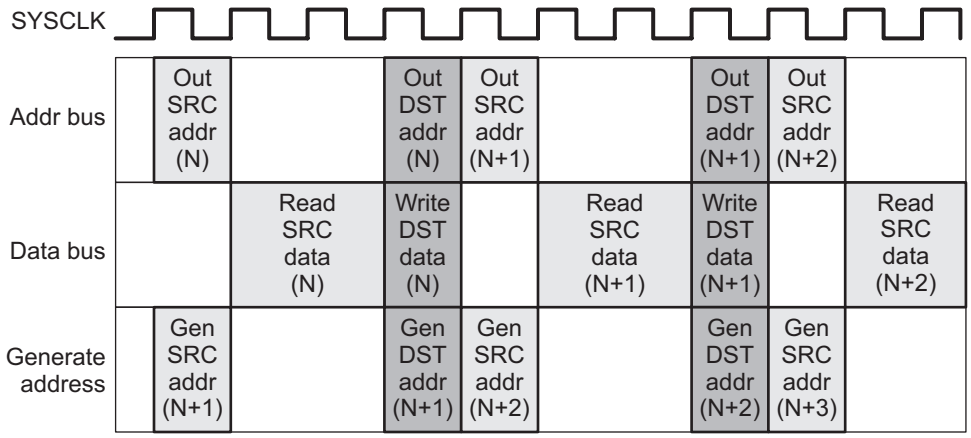
### *10.2.3 DMA Bus*

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 are via interfaces that sometimes share resources with the CPU memory or peripheral bus. Arbitration rules are defined in Section 10.4.

## 10.3 Pipeline Timing and Throughput

The DMA module consists of a 4-stage pipeline as shown in Figure 10-4.

**Figure 10-4. 4-Stage Pipeline DMA Transfer**




In addition to the pipeline there are a few other behaviors of the DMA that affect its total throughput:

- A 1-cycle delay is added at the beginning of each burst
- A 1-cycle delay is added when returning from a CH1 high priority interrupt
- Collisions with the CPU may add delay slots (see Section 10.4)
- 32-bit transfers run at double the speed of a 16-bit transfer (it takes the same amount of time to transfer a 32-bit word as it does a 16-bit word)

For example, to transfer 128 16-bit words from GS0 RAM to GS3 RAM, a channel can be configured to transfer 8 bursts of 16 words/burst. This will give:

8 bursts * [(4 cycles/word * 16 words/burst) + 1] = 520 cycles

If instead the channel were configured to transfer the same amount of data 32 bits at a time (the word size is configured to 32 bits) the transfer would take:

8 bursts * [(4 cycles/word * 8 words/burst) + 1] = 264 cycles

## 10.4 CPU and CLA Arbitration

Typically, DMA activity is independent of CPU and CLA activity. However, when the DMA and CPU (or CLA) try to access the same peripheral at the same time, an arbitration procedure is required to resolve the conflict. All instances of the same peripheral type conflict with each other. For instance, CAN-A and CAN-B conflict, as do the GS0 and GS2 RAMs. Different peripheral types can share a bus interface, which creates further opportunities for conflicts. These bus interfaces are:

- Peripheral frame 1: ePWM, eCAP, eQEP, SDFM, CMPSS, DAC, and PGA
- Peripheral frame 2: PMBus and SPI

**Conflict Example:** The CLA is accessing CAN-A while the DMA is simultaneously accessing CAN-B.

**Conflict Example:** The CPU is accessing an SPI FIFO while the DMA is simultaneously accessing a PMBus register.

**Non-conflict Example**: The CPU is accessing a shared ePWM while the DMA is accessing an SPI.

The exception to all this is the ADC result registers, which are duplicated for each bus master. The CPU, DMA, and CLA can all simultaneously read these registers with no stalls for any master.

A DMA transfer consists of four phases: send source address, read source data, send destination address, and write destination data (see Section 10.3). In the case of a block DMA transfer to and from the same memory interface the CPU is trying to access, the arbitration will stall CPU access until the DMA completes an access, not the entire transfer. This arbitration is based on a round robin priority scheme described in Section 10.5.

The following priority schemes are implemented for the various interfaces on the device.

- The fixed priority scheme for the peripheral frames is:
  - CLA/DMA Write
  - CLA/DMA Read
  - CPU Write
  - CPU Read
- The priority scheme for GSx RAM accesses is round-robin.
- All masters can access the ADC result registers simultaneously without delay.

**NOTE:** If the CPU is performing a read-modify-write operation and the DMA performs a write to the same location, the DMA write may be lost if the operation occurs in between the CPU read and the CPU write. Avoid mixing CPU writes with DMA writes to the same locations.

## 10.5 Channel Priority

Two priority schemes exist when determining channel priority: Round-robin mode and Channel 1 high-priority mode.

### *10.5.1 Round-Robin Mode*

In this mode, all channels have *equal* priority and each enabled channel is serviced in round-robin fashion as follows:

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. You can specify the size of the burst for each channel. Once CH6 (or the last enabled channel) has been serviced, and no other channels are pending, the round-robin state machine enters an idle state.

From the idle state, channel 1 (if enabled) is always serviced first. However, if the DMA is currently processing another channel x, all other pending channels between x and the end of the round are serviced before CH1. It is in this sense that all the channels are of *equal* priority. For instance, take an example where CH1, CH4, and CH5 are enabled in round-robin mode and CH4 is currently being processed. Then CH1 and CH5 both receive an interrupt trigger from their respective peripherals before CH4 completes. CH1 and CH5 are now both pending. When CH4 completes its burst, CH5 will be serviced next. Only after CH5 completes will CH1 be serviced. Upon completion of CH1, if there are no more channels pending, the round-robin state machine will enter an idle state.

A more complicated example is shown below:

- Assume all channels are enabled, and the DMA is in an idle state,
- Initially a trigger occurs on CH1, CH3, and CH5 on the same cycle,
- When the CH1 burst transfer starts, requests from CH3 and CH5 are pending,
- Before completion of the CH1 burst, the DMA receives a request from CH2. Now the pending requests are from CH2, CH3, and CH5,
- After completing the CH1 burst, CH2 will be serviced since it is next in the round-robin scheme after CH1.
- After the burst from CH2 is finished, the CH3 burst will be serviced, followed by CH5 burst.
- Now while the CH5 burst is being serviced, the DMA receives a request from CH1, CH3, and CH6.
- The burst from CH6 will start after the completion of the CH5 burst since it is the next channel after CH5 in the round-robin scheme.
- This will be followed by the CH1 burst and then the CH3 burst
- After the CH3 burst finishes, assuming no more triggers have occurred, the round-robin state machine will enter an idle state.

The round-robin state machine may be reset to the idle state via the DMACTRL[PRIORITYRESET] bit.

### 10.5.2 Channel 1 High Priority Mode

In this mode, if a CH1 trigger occurs, the current word transfer or the current + 1 word transfer (depends on which phase of the current DMA transfer the new CH1 trigger occurred) on any other channel is completed (not the complete burst), execution is halted, and CH1 is serviced for the complete burst count. When the CH1 burst is complete, execution returns to the channel that was active when the CH1 trigger occurred. All other channels have equal priority and each enabled channel is serviced in round-robin fashion as follows:

| | |
|---|---|
| Higher Priority: | CH1 |
| Lower priority: | CH2 → CH3 → CH4 → CH5 → CH6 → CH2 → … |

Given an example where CH1, CH4 and CH5 are enabled in Channel 1 High Priority Mode and CH4 is currently being processed. Then CH1 and CH5 both receive an interrupt trigger from their respective peripherals before CH4 completes. CH1 and CH5 are now both pending. When the current CH4 word transfer is completed, regardless of whether the DMA has completed the entire CH4 burst, CH4 execution will be suspended and CH1 will be serviced. After the CH1 burst completes, CH4 will resume execution.

Upon completion of CH4, CH5 will be serviced. After CH5 completes, if there are no more channels pending, the round-robin state machine will enter an idle state.

Typically Channel 1 would be used in this mode for the ADC, since its data rate is so high. However, Channel 1 High Priority Mode may be used in conjunction with any peripheral.

**NOTE:** High-priority mode and ONESHOT mode may not be used at the same time on channel 1. Other channels may use ONESHOT mode when channel 1 is in high-priority mode.

## 10.6 Address Pointer and Transfer Control

The DMA state machine is, at its most basic level, two nested loops. The inner loop transfers a burst of data when a peripheral interrupt trigger is received. A burst is the smallest amount of data that can be transferred at one time and its size is defined by the BURST_SIZE register for each channel. The BURST_SIZE register allows a maximum of 32 sixteen-bit words to be transferred in one burst. The outer loop, whose size is set by the TRANSFER_SIZE register for each channel, defines how many bursts are performed in the entire transfer. Since TRANSFER_SIZE is a 16-bit register, the total size of a transfer allowed is well beyond any practical requirement. One CPU interrupt is generated, if enabled, for each transfer. This interrupt can be configured to occur at the beginning or the end of the transfer via the MODE.CHx[CHINTMODE] bit.

In the default setting of the MODE.CHx[ONESHOT] bit, the DMA transfers one burst of data each time a peripheral interrupt trigger is received. After the burst is completed, the state machine moves on to the next pending channel in the priority scheme, even if another trigger for the channel just completed is pending. This feature keeps any single channel from monopolizing the DMA bus. If a transfer of more than the maximum number of words per burst is desired for a single trigger, the MODE.CHx[ONESHOT] bit can be set to complete the entire transfer when triggered. Be careful when using this mode, since this can create a condition where one trigger uses up the majority of the DMA bandwidth.

Each DMA channel contains a shadowed address pointer for the source and the destination address. These pointers, SRC_ADDR and DST_ADDR, can be independently controlled during the state machine operation. At the beginning of each transfer, the shadowed version of each pointer is copied into its respective active register. During the burst loop, after each word is transferred, the signed value contained in the appropriate source or destination BURST_STEP register is added to the active SRC/DST_ADDR register. During the transfer loop, after each burst is complete, there are two methods that can be used to modify the active address pointer. The first, and default, method is by adding the signed value contained in the SRC/DST_TRANSFER_STEP register to the appropriate pointer. The second is via a process called wrapping, where a wrap address is loaded into the active address pointer. When a wrap procedure occurs, the associated SRC/DST_TRANSFER_STEP register has no effect.

Address wrapping occurs when a number of bursts specified by the appropriate SRC/DST_WRAP_SIZE register completes. Each DMA channel contains two shadowed wrap address pointers, SRC_BEG_ADDR and DST_BEG_ADDR, allowing the source and destination wrapping to be independent of each other. Like the SRC_ADDR and DST_ADDR registers, the active SRC/DST_BEG_ADDR registers are loaded from their shadow counterpart at the beginning of a transfer. When the specified number of bursts has occurred, a two part wrap procedure takes place:

- The appropriate active SRC/DST_BEG_ADDR register is incremented by the signed value contained in the SRC/DST_WRAP_STEP register, then
- The new active SRC/DST_BEG_ADDR register is loaded into the active SRC/DST_ADDR register.

Additionally the wrap counter (SRC/DST_WRAP_COUNT) register is reloaded with the SRC/DST_WRAP_SIZE value to setup the next wrap period. This allows the channel to wrap multiple times within a single transfer. Combined with the first bullet above, this allows the channel to address multiple buffers within a single transfer.

The DMA contains both an active and shadow set of the following address pointers. When a DMA transfer begins, the shadow register set is copied to the active working set of registers. This allows you to program the values of the shadow registers for the next transfer while the DMA works with the active set. It also allows you to implement Ping-Pong buffer schemes without disrupting the DMA channel execution.

**Source/Destination Address Pointers (SRC/DST_ADDR)—** The value written into the shadow register is the start address of the first location where data is read or written to.

At the beginning of a transfer the shadow register is copied into the active register. The active register performs as the current address pointer.

**Source/Destination Begin Address Pointers (SRC/DST_BEG_ADDR)—** This is the wrap pointer.

The value written into the shadow register will be loaded into the active register at the start of a transfer. On a wrap condition, the active register will be incremented by the signed value in the appropriate SRC/DST_WRAP_STEP register prior to being loaded into the active SRC/DST_ADDR register.

For each channel, the transfer process can be controlled with the following size values:

**Source and Destination Burst Size (BURST_SIZE): —** This specifies the number of words to be transferred in a burst.

This value is loaded into the BURST_COUNT register at the beginning of each burst. The BURST_COUNT decrements each word that is transferred and when it reaches a zero value, the burst is complete, indicating that the next channel can be serviced. The behavior of the current channel is defined by the ONE_SHOT bit in the MODE register. The maximum size of the burst is dictated by the type of peripheral. For the ADC, the burst size could be all 16 registers (if all 16 registers are used). For RAM the burst size can be up to the maximum allowed by the BURST_SIZE register, which is 32.

**Source and Destination Transfer Size (TRANSFER_SIZE): —** This specifies the number of bursts to be transferred per CPU interrupt (if enabled).

Whether this interrupt is generated at the beginning or the end of the transfer is defined in the CHINTMODE bit in the MODE register. Whether the channel remains enabled or not after the transfer is completed is defined by the CONTINUOUS bit in the MODE register. The TRANSFER_SIZE register is loaded into the TRANSFER_COUNT register at the beginning of each transfer. The TRANSFER_COUNT register keeps track of how many bursts of data the channel has transferred and when it reaches zero, the DMA transfer is complete.

**Source/Destination Wrap Size (SRC/DST_WRAP_SIZE)—** This specifies the number of bursts to be transferred before the current address pointer wraps around to the beginning.

This feature is used to implement a circular addressing type function. This value is loaded into the appropriate SRC/DST_WRAP_COUNT register at the beginning of each transfer. The SRC/DST_WRAP_COUNT registers keep track of how many bursts of data the channel has transferred and when they reach zero, the wrap procedure is performed on the appropriate source or destination address pointer. A separate size and count register is allocated for source and destination pointers. To *disable* the wrap function, assign the value of these registers to be larger than the TRANSFER_SIZE.

**NOTE:** The value written to the SIZE registers is one less than the intended size. So, to transfer three 16-bit words, the value 2 should be placed in the SIZE register.

Regardless of the state of the DATASIZE bit, the value specified in the SIZE registers are for 16-bit addresses. So, to transfer three 32-bit words, the value 5 should be placed in the SIZE register.

For each source/destination pointer, the address changes can be controlled with the following step values:

**Source/Destination Burst Step (SRC/DST_BURST_STEP)—** Within each burst transfer, the address source and destination step sizes are specified by these registers.

This value is a signed 2's 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 should be set to zero.

**Source/Destination Transfer Step (SRC/DST_TRANSFER_STEP)—** This specifies the address offset to start the next burst transfer after completing the current burst transfer.

This is used in cases where registers or data memory locations are spaced at constant intervals. This value is a signed 2's 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 BEG_ADDR pointer and hence sets the new start address.

This implements a circular type of addressing mode, useful in many applications. This value is a signed 2's 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 should be placed in these registers.

Three modes are provided to control the way the state machine behaves during the burst loop and the transfer loop:

**One Shot Mode (ONESHOT)—** If one shot mode is enabled when an interrupt event trigger occurs, the DMA will continue transferring data in bursts until TRANSFER_COUNT is zero. If one shot mode is disabled, then an interrupt event trigger is required for each burst transfer and this will continue until TRANSFER_COUNT is zero.

**NOTE:** When ONESHOT mode is enabled, the DMA will continuously transfer bursts of data on the given channel until the TRANSFER_COUNT value is zero. This could potentially hog the bandwidth of a peripheral and cause long CPU stalls to occur. To avoid this, you could configure a CPU timer (or similar) and disable ONESHOT so as to avoid this situation.

High-priority mode and ONESHOT mode may not be used at the same time on channel 1. Other channels may use ONESHOT mode when channel 1 is in high-priority mode.

**Continuous Mode (CONTINUOUS)—** If continuous mode is disabled the RUNSTS bit in the CONTROL register is cleared at the end of the transfer, disabling the DMA channel.

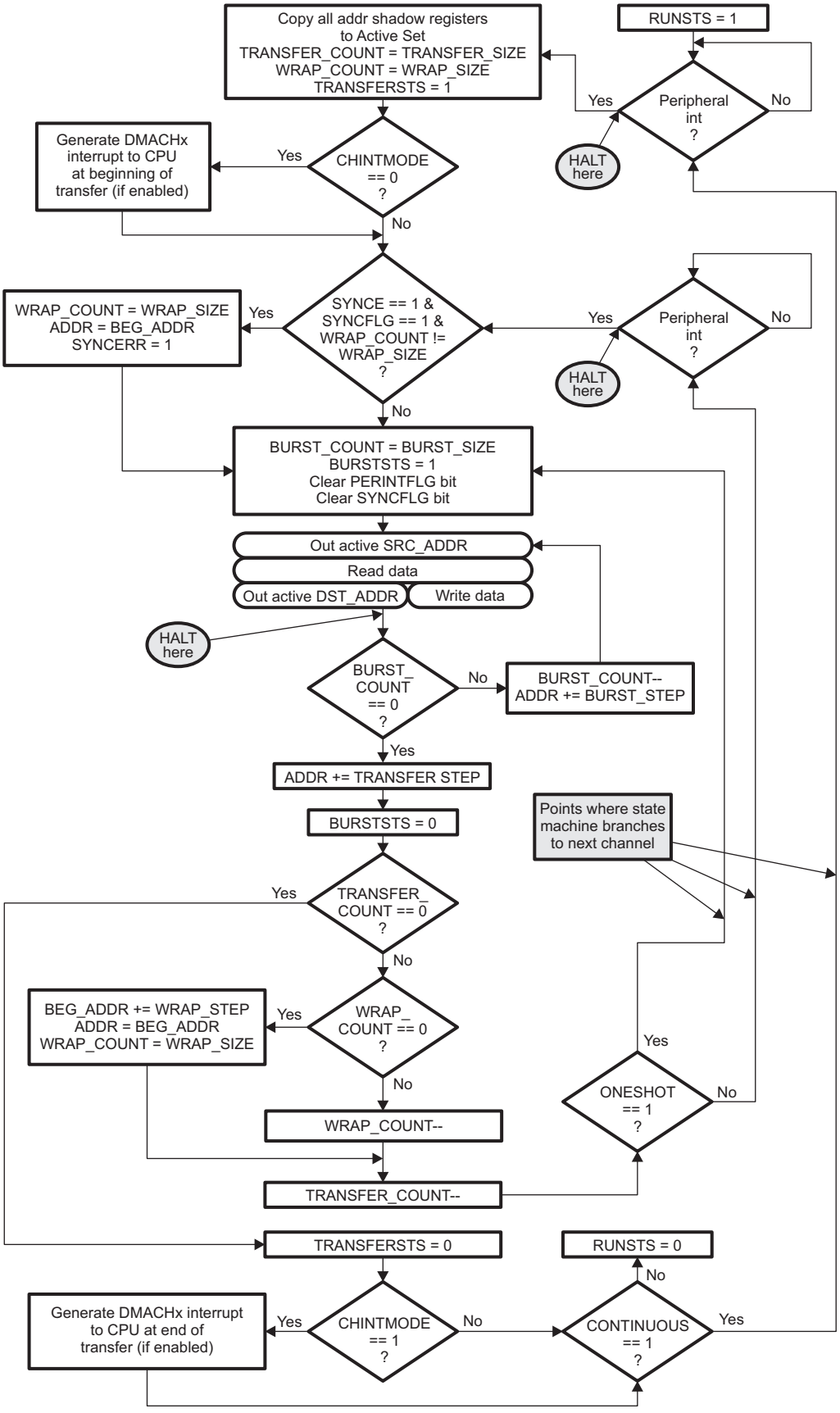
The channel must be re-enabled by setting the RUN bit in the CONTROL register before another transfer can be started on that channel. If the continuous mode is enabled the RUNSTS bit is not cleared at the end of the transfer.

**Channel Interrupt Mode (CHINTMODE)—** This mode bit selects whether the DMA interrupt from the respective channel is generated at the beginning of a new transfer or at the end of the transfer.

If implementing a ping-pong buffer scheme with continuous mode of operation, then the interrupt would be generated at the beginning, just after the working registers are copied to the shadow set. If the DMA does not operate in continuous mode, then the interrupt is typically generated at the end when the transfer is complete.

All of the above features and modes are shown in Figure 10-5.

**Figure 10-5. DMA State Diagram**



The following items are in reference to Figure 10-5.

- 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 ADDR registers are not affected by BEG_ADDR at the start of a transfer. BEG_ADDR only affects the ADDR registers on a wrap or sync error. Following is what happens to each of the ADDR registers when a transfer first starts:
  - BEG_ADDR_SHADOW remains unchanged.
  - ADDR_SHADOW remains unchanged.
  - BEG_ADDR = BEG_ADDR_SHADOW
  - ADDR = ADDR_SHADOW
- The active registers get updated when a wrap occurs. The shadow registers remain unchanged. Specifically:
  - BEG_ADDR_SHADOW remains unchanged.
  - ADDR_SHADOW remains unchanged.
  - BEG_ADDR += WRAP_STEP
  - ADDR = BEG_ADDR
- The active registers get updated when a sync error occurs. The shadow registers remain unchanged. Specifically:
  - BEG_ADDR_SHADOW remains unchanged.
  - ADDR_SHADOW remains unchanged.
  - BEG_ADDR remains unchanged.
  - ADDR = BEG_ADDR

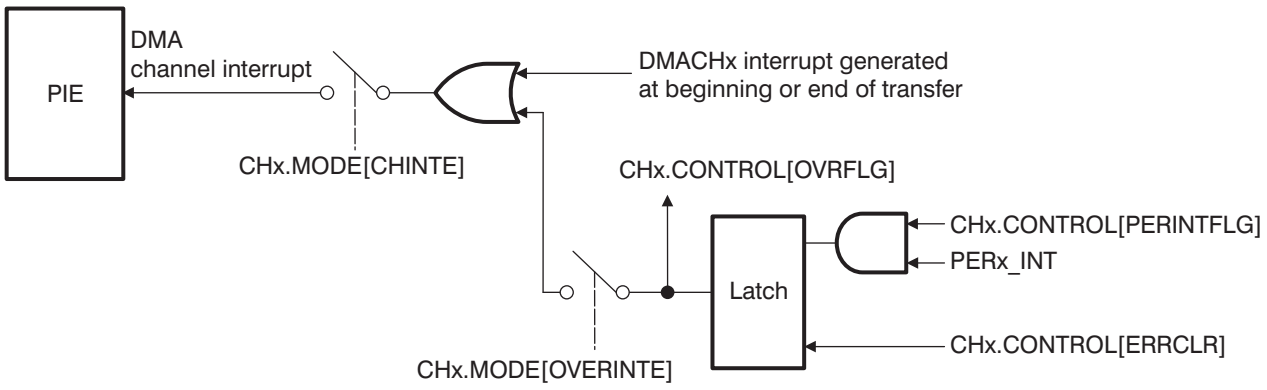
Probably the easiest way to remember all this is that:

- The shadow registers never change except by software.
- The active registers never change except by hardware, and a shadow register is only copied into its own active register, never an active register by another name.

## 10.7 Overrun Detection Feature

The DMA contains overrun detection logic. When a peripheral event trigger is received by the DMA, the PERINTFLG bit in the CONTROL register is set, pending the channel to the DMA state machine. When the burst for that channel is started, the PERINTFLG is cleared. If however, between the time that the PERINTFLG bit is set by an event trigger and cleared by the start of the burst, an additional event trigger arrives, the second trigger will be lost. This condition will set the OVRFLG bit in the CONTROL register as in Figure 10-6. If the overrun interrupt is enabled, the channel interrupt will be generated to the PIE module.

**Figure 10-6. Overrun Detection Logic**



## 10.8 Registers

### 10.8.1 DMA Base Addresses

**Table 10-2. DMA Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| DmaRegs | DMA_REGS | 0x0000_1000 | 0x0000_11FF |

### 10.8.1.1 DMA_REGS Registers

Table 10-3 lists the memory-mapped registers for the DMA_REGS. All register offset addresses not listed in Table 10-3 should be considered as reserved locations and the register contents should not be modified.

**Table 10-3. 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 10-4 shows the codes that are used for access types in this section.

**Table 10-4. DMA_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| W=1 | 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. |

#### 10.8.1.1.1 DMACTRL Register (Offset = 0h) [reset = 0h]

DMACTRL is shown in Figure 10-7 and described in Table 10-5.

Return to Summary Table.

DMA Control Register

**Figure 10-7. 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/W=1-0h | R=0/W=1-0h |

**Table 10-5. DMACTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-2 | RESERVED | R | 0h | Reserved |
| 1 | PRIORITYRESET | R=0/W=1 | 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/W=1 | 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 |

#### 10.8.1.1.2 DEBUGCTRL Register (Offset = 1h) [reset = 0h]

DEBUGCTRL is shown in Figure 10-8 and described in Table 10-6.

Return to Summary Table.

Debug Control Register

**Figure 10-8. DEBUGCTRL Register**

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

**Table 10-6. 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 |

#### 10.8.1.1.3 PRIORITYCTRL1 Register (Offset = 4h) [reset = 0h]

PRIORITYCTRL1 is shown in Figure 10-9 and described in Table 10-7.

Return to Summary Table.

Priority Control 1 Register

**Figure 10-9. PRIORITYCTRL1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | CH1PRIORITY |
| R-0h | | | | | | | R/W-0h |

**Table 10-7. 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 |

#### 10.8.1.1.4 PRIORITYSTAT Register (Offset = 6h) [reset = 0h]

PRIORITYSTAT is shown in Figure 10-10 and described in Table 10-8.

Return to Summary Table.

Priority Status Register

**Figure 10-10. PRIORITYSTAT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | ACTIVESTS_SHADOW | | | RESERVED | ACTIVESTS | | |
| R-0h | R-0h | | | R-0h | R-0h | | |

**Table 10-8. 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 |

### 10.8.1.2 DMA_CH_REGS Registers

Table 10-9 lists the memory-mapped registers for the DMA_CH_REGS. All register offset addresses not listed in Table 10-9 should be considered as reserved locations and the register contents should not be modified.

**Table 10-9. 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 10-10 shows the codes that are used for access types in this section.

**Table 10-10. DMA_CH_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| W=1 | W | Write |
| **Reset or Default Value** | | |
| *-n* | | Value after reset or the default value |
| **Register Array Variables** | | |

**Table 10-10. DMA_CH_REGS Access Type Codes (continued)**

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

#### 10.8.1.2.1 MODE Register (Offset = 0h) [reset = 0h]

MODE is shown in Figure 10-11 and described in Table 10-11.

Return to Summary Table.

Mode Register

**Figure 10-11. 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-0h | R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OVRINTE | RESERVED | | PERINTSEL | | | | |
| R/W-0h | R-0h | | R/W-0h | | | | |

**Table 10-11. 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 | 0h | Reserved |
| 12 | RESERVED | R | 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. |

**Table 10-11. MODE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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. |
| 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 |

#### 10.8.1.2.2 CONTROL Register (Offset = 1h) [reset = 0h]

CONTROL is shown in Figure 10-12 and described in Table 10-12.

Return to Summary Table.

Control Register

**Figure 10-12. 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/W=1-0h | R-0h | R-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 10-12. 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 |

**Table 10-12. 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/W=1 | 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 | 0h | Reserved |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | PERINTCLR | R=0/W=1 | 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/W=1 | 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/W=1 | 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>Reset type: SYSRSn |
| 1 | HALT | R=0/W=1 | 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/W=1 | 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 |

#### 10.8.1.2.3 BURST_SIZE Register (Offset = 2h) [reset = 0h]

BURST_SIZE is shown in Figure 10-13 and described in Table 10-13.

Return to Summary Table.

Burst Size Register

**Figure 10-13. BURST_SIZE Register**

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

**Table 10-13. 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 |

#### 10.8.1.2.4 BURST_COUNT Register (Offset = 3h) [reset = 0h]

BURST_COUNT is shown in Figure 10-14 and described in Table 10-14.

Return to Summary Table.

Burst Count Register

**Figure 10-14. BURST_COUNT Register**

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

**Table 10-14. 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 |

#### 10.8.1.2.5 SRC_BURST_STEP Register (Offset = 4h) [reset = 0h]

SRC_BURST_STEP is shown in Figure 10-15 and described in Table 10-15.

Return to Summary Table.

Source Burst Step Register

**Figure 10-15. SRC_BURST_STEP Register**

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

**Table 10-15. 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 |

#### 10.8.1.2.6 DST_BURST_STEP Register (Offset = 5h) [reset = 0h]

DST_BURST_STEP is shown in Figure 10-16 and described in Table 10-16.

Return to Summary Table.

Destination Burst Step Register

**Figure 10-16. DST_BURST_STEP Register**

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

**Table 10-16. 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 |

#### 10.8.1.2.7 TRANSFER_SIZE Register (Offset = 6h) [reset = 0h]

TRANSFER_SIZE is shown in Figure 10-17 and described in Table 10-17.

Return to Summary Table.

Transfer Size Register

**Figure 10-17. TRANSFER_SIZE Register**

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

**Table 10-17. 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 |

#### 10.8.1.2.8 TRANSFER_COUNT Register (Offset = 7h) [reset = 0h]

TRANSFER_COUNT is shown in Figure 10-18 and described in Table 10-18.

Return to Summary Table.

Transfer Count Register

**Figure 10-18. TRANSFER_COUNT Register**

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

**Table 10-18. TRANSFER_COUNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | TRANSFERCOUNT | R/W | 0h | These bits indicate the number of bursts left in the current transfer.<br>Reset type: SYSRSn |

#### 10.8.1.2.9 SRC_TRANSFER_STEP Register (Offset = 8h) [reset = 0h]

SRC_TRANSFER_STEP is shown in Figure 10-19 and described in Table 10-19.

Return to Summary Table.

Source Transfer Step Register

**Figure 10-19. SRC_TRANSFER_STEP Register**

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

**Table 10-19. 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 |

#### 10.8.1.2.10 DST_TRANSFER_STEP Register (Offset = 9h) [reset = 0h]

DST_TRANSFER_STEP is shown in Figure 10-20 and described in Table 10-20.

Return to Summary Table.

Destination Transfer Step Register

**Figure 10-20. DST_TRANSFER_STEP Register**

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

**Table 10-20. 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 |

#### 10.8.1.2.11 SRC_WRAP_SIZE Register (Offset = Ah) [reset = 0h]

SRC_WRAP_SIZE is shown in Figure 10-21 and described in Table 10-21.

Return to Summary Table.

Source Wrap Size Register

**Figure 10-21. SRC_WRAP_SIZE Register**

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

**Table 10-21. SRC_WRAP_SIZE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | WRAPSIZE | R/W | 0h | These bits specify the number of bursts to transfer before the source address wraps around to the beginning address. The actual number is equal to WRAPSIZE + 1. To disable the wrapping function, set WRAPSIZE to a value larger than TRANSFERSIZE.<br>Reset type: SYSRSn |

#### 10.8.1.2.12 SRC_WRAP_COUNT Register (Offset = Bh) [reset = 0h]

SRC_WRAP_COUNT is shown in Figure 10-22 and described in Table 10-22.

Return to Summary Table.

Source Wrap Count Register

**Figure 10-22. SRC_WRAP_COUNT Register**

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

**Table 10-22. SRC_WRAP_COUNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | WRAPSIZE | R/W | 0h | These bits indicate the number of bursts left before wrapping the source address.<br>Reset type: SYSRSn |

#### 10.8.1.2.13 SRC_WRAP_STEP Register (Offset = Ch) [reset = 0h]

SRC_WRAP_STEP is shown in Figure 10-23 and described in Table 10-23.

Return to Summary Table.

Source Wrap Step Register

**Figure 10-23. SRC_WRAP_STEP Register**

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

**Table 10-23. 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 |

#### 10.8.1.2.14 DST_WRAP_SIZE Register (Offset = Dh) [reset = 0h]

DST_WRAP_SIZE is shown in Figure 10-24 and described in Table 10-24.

Return to Summary Table.

Destination Wrap Size Register

**Figure 10-24. DST_WRAP_SIZE Register**

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

**Table 10-24. DST_WRAP_SIZE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | WRAPSIZE | R/W | 0h | These bits specify the number of bursts to transfer before the destination address wraps around to the beginning address. The actual number is equal to WRAPSIZE + 1. To disable the wrapping function, set WRAPSIZE to a value larger than TRANSFERSIZE.<br>Reset type: SYSRSn |

#### 10.8.1.2.15 DST_WRAP_COUNT Register (Offset = Eh) [reset = 0h]

DST_WRAP_COUNT is shown in Figure 10-25 and described in Table 10-25.

Return to Summary Table.

Destination Wrap Count Register

**Figure 10-25. DST_WRAP_COUNT Register**

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

**Table 10-25. DST_WRAP_COUNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | WRAPSIZE | R/W | 0h | These bits indicate the number of bursts left before wrapping the destination address.<br>Reset type: SYSRSn |

#### 10.8.1.2.16 DST_WRAP_STEP Register (Offset = Fh) [reset = 0h]

DST_WRAP_STEP is shown in Figure 10-26 and described in Table 10-26.

Return to Summary Table.

Destination Wrap Step Register

**Figure 10-26. DST_WRAP_STEP Register**

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

**Table 10-26. 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 |

#### 10.8.1.2.17 SRC_BEG_ADDR_SHADOW Register (Offset = 10h) [reset = 0h]

SRC_BEG_ADDR_SHADOW is shown in Figure 10-27 and described in Table 10-27.

Return to Summary Table.

Source Begin Address Shadow Register

**Figure 10-27. SRC_BEG_ADDR_SHADOW Register**

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

**Table 10-27. SRC_BEG_ADDR_SHADOW Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | BEGADDR | R/W | 0h | Shadow Source Beginning Address<br><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 |

#### 10.8.1.2.18 SRC_ADDR_SHADOW Register (Offset = 12h) [reset = 0h]

SRC_ADDR_SHADOW is shown in Figure 10-28 and described in Table 10-28.

Return to Summary Table.

Source Address Shadow Register

**Figure 10-28. SRC_ADDR_SHADOW Register**

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

**Table 10-28. SRC_ADDR_SHADOW Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | ADDR | R/W | 0h | Shadow Source Address<br><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 |

#### 10.8.1.2.19 SRC_BEG_ADDR_ACTIVE Register (Offset = 14h) [reset = 0h]

SRC_BEG_ADDR_ACTIVE is shown in Figure 10-29 and described in Table 10-29.

Return to Summary Table.

Source Begin Address Active Register

**Figure 10-29. SRC_BEG_ADDR_ACTIVE Register**

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

**Table 10-29. SRC_BEG_ADDR_ACTIVE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | BEGADDR | R/W | 0h | Active Source Beginning Address<br><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><br>When a transfer starts, this register is loaded with the shadow address from the SRC_BEG_ADDR_SHADOW register.<br>Reset type: SYSRSn |

#### 10.8.1.2.20 SRC_ADDR_ACTIVE Register (Offset = 16h) [reset = 0h]

SRC_ADDR_ACTIVE is shown in Figure 10-30 and described in Table 10-30.

Return to Summary Table.

Source Address Active Register

**Figure 10-30. SRC_ADDR_ACTIVE Register**

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

**Table 10-30. SRC_ADDR_ACTIVE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | ADDR | R/W | 0h | Active Source Address<br><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 |

#### 10.8.1.2.21 DST_BEG_ADDR_SHADOW Register (Offset = 18h) [reset = 0h]

DST_BEG_ADDR_SHADOW is shown in Figure 10-31 and described in Table 10-31.

Return to Summary Table.

Destination Begin Address Shadow Register

**Figure 10-31. DST_BEG_ADDR_SHADOW Register**

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

**Table 10-31. DST_BEG_ADDR_SHADOW Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | BEGADDR | R/W | 0h | Shadow Destination Beginning Address<br><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 |

#### 10.8.1.2.22 DST_ADDR_SHADOW Register (Offset = 1Ah) [reset = 0h]

DST_ADDR_SHADOW is shown in Figure 10-32 and described in Table 10-32.

Return to Summary Table.

Destination Address Shadow Register

**Figure 10-32. DST_ADDR_SHADOW Register**

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

**Table 10-32. DST_ADDR_SHADOW Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | ADDR | R/W | 0h | Shadow Destination Address<br><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 |

#### 10.8.1.2.23 DST_BEG_ADDR_ACTIVE Register (Offset = 1Ch) [reset = 0h]

DST_BEG_ADDR_ACTIVE is shown in Figure 10-33 and described in Table 10-33.

Return to Summary Table.

Destination Begin Address Active Register

**Figure 10-33. DST_BEG_ADDR_ACTIVE Register**

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

**Table 10-33. DST_BEG_ADDR_ACTIVE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | BEGADDR | R/W | 0h | Active Destination Beginning Address<br><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><br>When a transfer starts, this register is loaded with the shadow address from the DST_BEG_ADDR_SHADOW register.<br>Reset type: SYSRSn |

#### 10.8.1.2.24 DST_ADDR_ACTIVE Register (Offset = 1Eh) [reset = 0h]

DST_ADDR_ACTIVE is shown in Figure 10-34 and described in Table 10-34.

Return to Summary Table.

Destination Address Active Register

**Figure 10-34. DST_ADDR_ACTIVE Register**

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

**Table 10-34. DST_ADDR_ACTIVE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | ADDR | R/W | 0h | Active Destination Address<br><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 |

# Embedded Real-time Analysis and Diagnostic (ERAD)

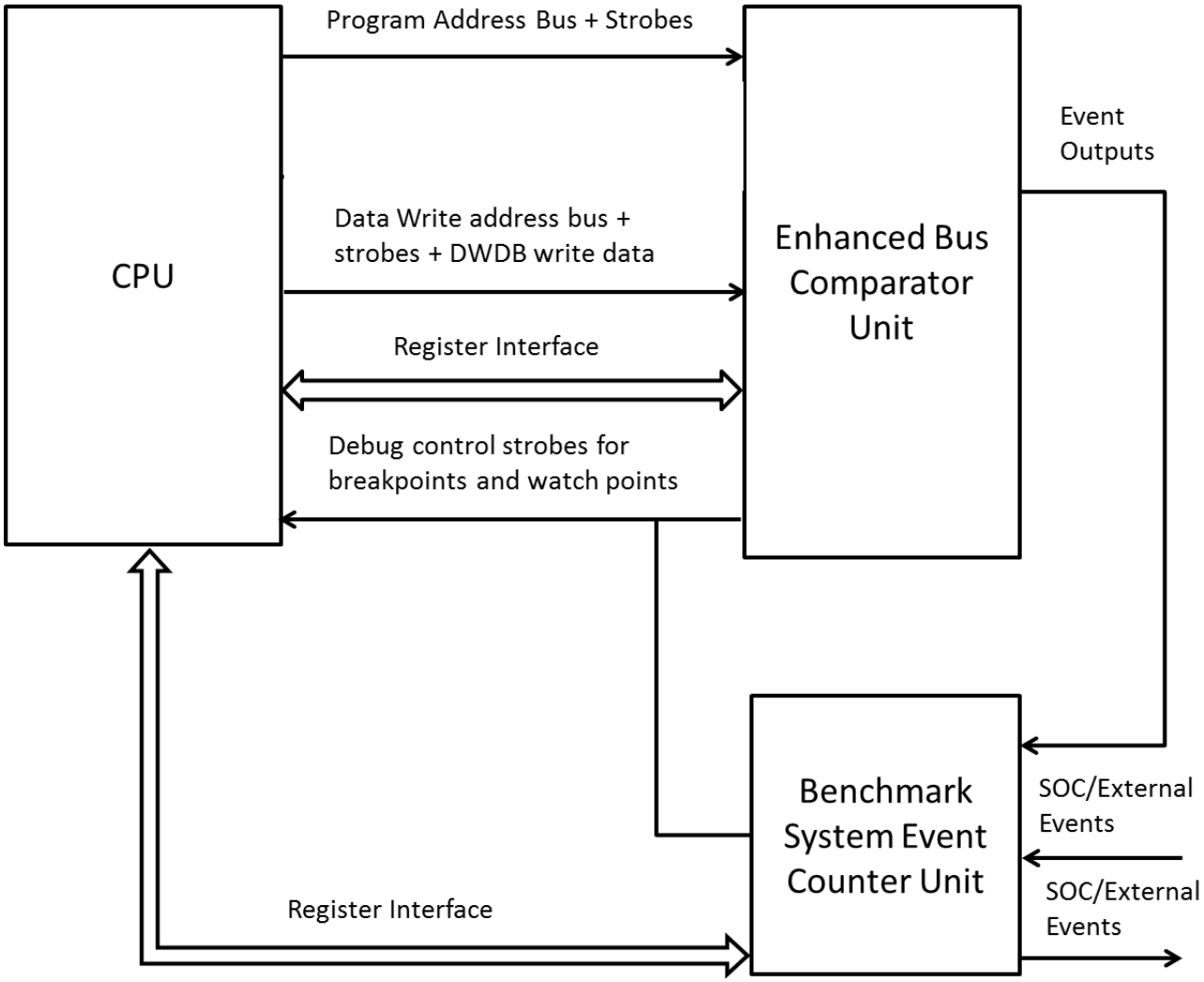
This chapter describes the features and operation of the embedded real-time analysis and diagnostic (ERAD) module. The ERAD module enhances the debug and system analysis capabilities of the device. The debug and system analysis enhancements provided by the ERAD module is done outside of the CPU. The ERAD module consists of the enhanced bus comparator units and the benchmark system event counter units. The enhanced bus comparator units are used to generate hardware breakpoints, hardware watch points and other output events. The benchmark system event counter units are used to analyze and profile the system. The ERAD module is accessible both by the debugger and the application software, which significantly increases the debug capabilities of many real-time systems, especially in situations where debuggers are not connected.

**Topic** **Page**

## 11.1 ERAD Module Overview

The ERAD module is shown in Figure 11-1.

**Figure 11-1. ERAD Overview**




ERAD enhances the debug and system analysis capabilities of the device from outside the CPU. The C28x CPU alone has two analysis resources. One resource can be used as a hardware breakpoint, hardware watch point, or counter. The other resource can be used as either a hardware breakpoint or hardware watch point. The ERAD module increases the debug capability significantly from outside the CPU by providing more hardware breakpoints, hardware watch points, counters for profiling, and other capabilities. The ERAD module can be utilized by the debugger, and also by the application software. For many real-time systems, it is not always possible to connect a debugger and perform intrusive debug. Under these situations, the user's code has the ability to set up and control the ERAD module in order to debug and profile the system without disturbing the end application.

The ERAD module consists of eight enhanced bus comparator units and four benchmark system event counter units. The enhanced bus comparator units monitor buses and generate output events. The benchmark system event counter units utilize their counter and possibly the output events of the enhanced bus comparator units to profile and analyze the system. These units are described in detail in the following sections.

## 11.2 Enhanced Bus Comparator Unit

The enhanced bus comparator units connect to the CPU similar to any standard direct memory interface connection. The program space buses, data space buses, debug qualifiers for memory access and debug-related strobes necessary for breakpoints, watch points and trace points, are connected between the CPU and these units. The enhanced bus comparator units are usually controlled by the debug software. However a CPU application code can also configure and use these units for interrupt and event generation. The ERAD module can be configured and used by either the application software or debug software. It is not possible for an enhanced bus comparator unit to be used by both the application software and the debug software simultaneously. The ERAD module will be owned by either the debugger or the application software through the memory mapped registers.

The enhanced bus comparator unit has the following capabilities:

- Generate hardware breakpoints
- Generate hardware watch points
- Generate trace tags for instruction fetch matches and generate RTOSINT
- Monitor data read address buses, data write address buses, data write data bus, and generate RTOSINT
- Generate an event output which can be used by other modules. This is done through monitoring any of the program address buses, VPC, or the Program Counter of the CPU

The following features are not supported by the enhanced bus comparator units:

- Chain breakpoints
- Ability to monitor DMA transfers
- Ability to monitor CLA buses

The enhanced bus comparator units have the capability to monitor a range of addresses by defining masks and generating outputs based on greater than, less than, or equal events.

The enhanced bus comparator units can be used with the benchmark system event counter units for profiling and analysis purposes.

### *11.2.1 Enhanced Bus Comparator Unit Operations*

The following operations are supported by the enhanced bus comparator unit:

- Operation of a hardware breakpoint: The bus comparator unit will generate the appropriate tags for an access made to a particular address. When the instruction is going to enter the DECODE 2 (D2) phase of the pipeline, the CPU is halted.
- Operation of a watch point: Unlike breakpoints which halt the CPU before a designated address is executed, watch points will detect the occurrence of a data memory read or write, and will halt the CPU. There is no precise definition as to when the CPU will halt, since it is entirely dependant on the current state of the CPU pipeline. The CPU will halt at the next interruptible boundary.
- Operation of program trace: This operation is very similar to the hardware breakpoint operation. The difference between the CPU behavior is that it will internally generate an RTOSINT when the tagged instruction reaches the DECODE 2 (D2) phase of the pipeline. If this instruction is discarded in the fetch buffer due to discontinuity, then no RTOSINT will be generated.
- Operation of RTOSINTn: This is an alternative to a watch point where an interrupt can be generated to the CPU on detecting the designated read and write access.

## 11.3 Benchmark System Event Counter Unit

The benchmark system event counter units provide better system profiling, analysis, and debug capability. The benchmark system event counter units contain counters which can enhance the debug and profiling process in various types of system scenarios such as:

- Profiling code segments
- Counting duration between specified memory reads and writes
- Counting system events (such as interrupts)

- Counting duration between system events
- System timer
- Measuring the number of wait states in code segments
- Measuring the maximum amount of time spent in between a pair of events, measured over multiple iterations
- Chaining counters in order to link events or create larger counters

Furthermore, the benchmark system event counter unit has the capability to:

- Function as a counter capable of counting:
  - Any of the match events generated by the enhanced bus comparator units
  - Events generated by the enhanced bus comparator units. These events can be used to start and stop the counting.
  - System events including the PIE interrupts, timer interrupts, and CLA task interrupts. These system events can be used to start and stop the counting.
- Generate an interrupt or a watch point if the count reaches a reference value.
- Operate in counting mode in one of the following two modes:
  - Duration mode: The counter will count the CPU cycles as long as the event is active.
  - Event mode: The counter will count only the positive edge of the event signal. This is effectively counting the number of times the event transitions from inactive to active.

### *11.3.1 Benchmark System Event Counter Modes*

The following are the key features of the counter module. The counters are initialized to zero and will always count up.

- Continuos Count: In this mode the counter continues to count as specified by the input selector. If the counter reaches the maximum value, it resets to zero and will continue to count up. A sticky overflow bit will be set to indicate that this counter had overflowed. The counter can count the CPU cycles without any events selected. In this mode the module can be used as a software controlled SYSCLK counter.
- Timer Mode Count: In this mode, the counter counts up to a set reference value and after reaching the reference value, it resets to zero. The counter will generate an event which can send an interrupt to the CPU or generate a watch point. The counter can be set up to either continue incrementing or reset when a match event occurs.
- Start-Stop Count: In this mode, two events are allowed to act as start and stop indicators to the counter. The counter will commence counting only when the defined start event occurs. The counter will then continue to count up until the stop event occurs. Once the first start event has occurred, further start events will be ignored till the stop event occurs.

#### 11.3.1.1 Counting Active Levels vs. Edges

The benchmark system event counters can be configured to either count active levels or edges of the selected inputs.

The counter module has eight inputs from the enhanced bus comparator units and 24 inputs from other events in the device. Each counter can be configured to count any of the input events or just count up on every cycle. For example if an input event occurs and is active for 25 cycles, the counter will increment only by 1 in event mode, whereas in the duration mode, it will increment by 25.

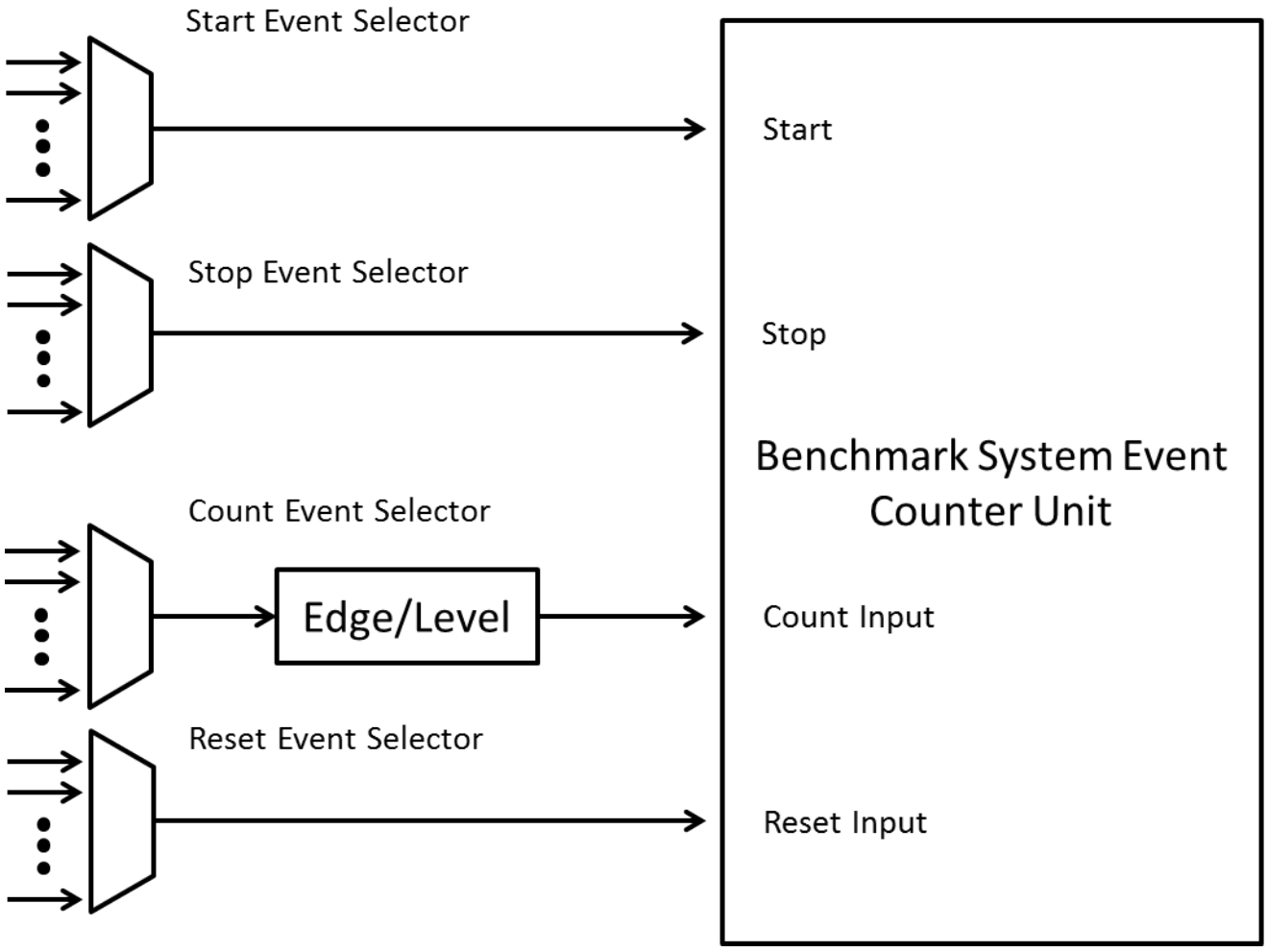
#### 11.3.1.2 Max mode

Max mode is also supported by the benchmark system event counter units. This mode allows the user to detect the maximum count that has occurred during various count iterations in start-stop mode. For example, a user can set up the counter in the start-stop count mode to count the duration of a critical code loop. Every time the stop event occurs and the counter stops, the counter value is checked against the current MAX_COUNT present in the register. If the new value is greater, then the MAX_COUNT register is updated. The counter will always reset to zero at the stop event and will be ready to start counting on the next start event. Therefore the MAX_COUNT will contain the maximum number of cycles that occurred between the start and stop condition over many iterations.

#### 11.3.1.3 Input Signal Selection

The benchmark system event counter inputs can be selected from various signals from in the system to enable debug and system analysis. Figure 11-2 shows the benchmark system event counter inputs.

**Figure 11-2. Benchmark System Event Counter Inputs**




Each event selector MUX can select from various signals on in the system. These signals are show in Table 11-1.

**Table 11-1. Event Selector Mux Signals**

| CTM_INP_SEL, STA_INP_SEL, STO_INP_SEL, RST_INP_SEL | EVENT_INPUT_SELECTED |
|---|---|
| INP_SEL[0] | BUS_COMP1_EVENT |
| INP_SEL[1] | BUS_COMP2_EVENT |
| INP_SEL[2] | BUS_COMP3_EVENT |
| INP_SEL[3] | BUS_COMP4_EVENT |
| INP_SEL[4] | BUS_COMP5_EVENT |
| INP_SEL[5] | BUS_COMP6_EVENT |
| INP_SEL[6] | BUS_COMP7_EVENT |
| INP_SEL[7] | BUS_COMP8_EVENT |
| INP_SEL[8] | COUNTER1_EVENT |
| INP_SEL[9] | COUNTER2_EVENT |
| INP_SEL[10] | COUNTER3_EVENT |
| INP_SEL[11] | COUNTER4_EVENT |
| INP_SEL[12] | PIE_INT1 |
| INP_SEL[13] | PIE_INT2 |
| INP_SEL[14] | PIE_INT3 |
| INP_SEL[15] | PIE_INT4 |
| INP_SEL[16] | PIE_INT5 |
| INP_SEL[17] | PIE_INT6 |
| INP_SEL[18] | PIE_INT7 |
| INP_SEL[19] | PIE_INT8 |
| INP_SEL[20] | PIE_INT9 |
| INP_SEL[21] | PIE_INT10 |
| INP_SEL[22] | PIE_INT11 |
| INP_SEL[23] | PIE_INT12 |
| INP_SEL[24] | TIMER1_TINT1 |
| INP_SEL[25] | TIMER2_TINT2 |
| INP_SEL[26] | CLA_INTERRUPT1 |
| INP_SEL[27] | CLA_INTERRUPT2 |
| INP_SEL[28] | CLA_INTERRUPT3 |
| INP_SEL[29] | CLA_INTERRUPT4 |
| INP_SEL[30] | CLA_INTERRUPT5 |
| INP_SEL[31] | CLA_INTERRUPT8 |

### 11.3.2 Reset on Event

It is also possible to reset the counters on external events. Additionally all the counter event outputs are fed back to each of the counter's input MUX which will select the event that can be used as a reset input. When enabled, an active high on the reset input will cause the counter to reset. This gives a powerful feature which allows the user to set up threshold monitors. This can be used to flag an interrupt or a watch point if the distance between two events crosses a certain threshold.

### 11.3.3 Operation Conditions

The benchmark event counter units will count accurately only when the CPU is operating in normal conditions. If the counters are running and capturing the CPU cycles while the CPU is controlled through the debugger in order to single-step through the code, then the result may differ from when the CPU was executing the code in normal conditions.

It is also important to note that if the counters are set up to use the value of the program counter register(VPC) as the source for the start and stop events, the value of the CPU cycles measured may be off by a few cycles when the CALL instruction is executed.

## 11.4 ERAD Ownership, Initialization and Reset

Even though the features of this module are meant for the debugger to use, the user application may also need to take advantage of the capabilities to monitor the buses and generate interrupts and events. It is an option to completely hand over the ownership of the ERAD module to the application software or the debugger. Another option is for both the application code and the debugger to use the ERAD module. Only the owner of the module (application code or debugger) is allowed to use the module at that time. The responsibility of resolving any ownership conflict, is on the software. The last option is to have no owner. In this mode, both the application code and the debugger have the capabillity to access the module at the same time. It is critical for the software, both on the application side and the debugger side, to resolve any conflicts. In this mode, it is possible for the debugger to use some of the enhanced bus comparator units and the benchmark system event counter units, while the application software uses the remaining units.

The ERAD module will initialize its internal states and all the registers to their initialized states under the following conditions:

- At power-on-reset (POR)
- With DCON and SYSRSN
  - Debug logic disconnected when the debugger owns the module
  - Functional reset when application owns the module

## 11.5 ERAD Programming Sequence

The ERAD module can be used to set hardware breakpoints and hardware watch points. The programming sequence to set hardware breakpoints, hardware watch points, or to use the timers to profile and analyze the system are described in this section. The same sequence can be used through the debugger software or the application code.

Please refer to the Driverlib example projects in C2000Ware for JavaScript files to configure the ERAD module. Example projects are also available to showcase the usage of these script files. These examples are included in the latest Driverlib install.

<C2000Ware installation>\driverlib\f28004x\examples\erad

### *11.5.1 Hardware Breakpoint and Hardware Watch Point Programming Sequence*

The programming sequence is identical when using the enhanced bus comparator units, regardless of whether it is a debug software or the application programing the units. A typical programming sequence for a unit is described below.

- Read and make sure the ownership is set as expected. If not, acquire the ownership before proceeding further if required.
- Ensure the unit is in IDLE mode.
- Set up the address reference, mask, bus select and stop bits.
- Enable the corresponding module bit in the global enable register.
- Once the usage is completed, write ‘1’ to Clear EVENT_FIRED sticky bit. This will take the module back to the enabled state.

The example programming sequences for hardware breakpointa and hardware watch points are shown below.

Set a hardware breakpoint on address 0x201000:

- Read GLBL_OWNER to confirm ownership.
- Read HWBP_STATUS to confirm the module is in IDLE state.
- Write 0x0 to HWBP_MASK.
- Write 0x201000 to HWBP_REF.
- Write 0x20 to HWBP_CNTL (STOP = 1, BUS_SEL = 000/PAB). If a trace is to be generated instead of a breakpoint, then set HWBP_CNTL to 0x40 instead of (STOP =0).
- Enable the corresponding module bit in the global enable register.

Set a hardware watch point on read of addresses from 0x121010 to 0x12101f:

- Read GLBL_OWNER to confirm ownership.
- Read HWBP_STATUS to confirm the module is in IDLE state.
- Write 0xf to HWBP_MASK.
- Write 0x121010 to HWBP_REF.
- Write 0x2C to HWBP_CNTL (STOP = 1, BUS_SEL = 011/DRAB). If an RTOSINTn is to be generated instead of a watch point, then set HWBP_CNTL to 0x4C instead (STOP =0).
- Enable the corresponding module bit in the global enable register.

Set a hardware watch point on write to address 0xff10101a:

- Read GLBL_OWNER to confirm ownership
- Read HWBP_STATUS to confirm the module is in IDLE state
- Write 0x0 to HWBP_MASK
- Write 0xff10101a to HWBP_REF
- Write 0x28 to HWBP_CNTL (STOP = 1, BUS_SEL = 010/DWAB). If an RTOSINTn is to be generated instead of a watch point, then set HWBP_CNTL to 0x48 instead (STOP =0)
- Enable the corresponding module bit in the global enable register.

### 11.5.2 Timer and Counter Programming Sequence

The programming sequence is identical when using the Benchmark System Event Counter units, regardless of whether it is a debug software, or the application, programing the units. Typical programming sequence for a unit is described below.

- Read and make sure the ownership is set as expected. If not, acquire the ownership before proceeding further if required.
- Ensure the unit is in IDLE mode.
- Set up the counter reference, counter registers (clear/reset if a clean start is required) and CTM_CNTL.
- Enable the corresponding module bit in the global enable register.
- Once the usage is completed, write ‘1’ to Clear EVENT_FIRED sticky bit. This will take the module back to the enabled state.

Set up a free running counter:

- Read and make sure the ownership is set as expected. If not, acquire the ownership before proceeding further, if required.
- Read CTM_STATUS to confirm that the module is in IDLE state
- Write 0x0 to CTM_INPUT_SEL
- Write 0x0 to CTM_CNTL
- Enable the module in the GLBL_ENABLE register.

Set up the counter to count the duration spent between addresses 0x1000 and 0x1210:

- Read and make sure the ownership is set as expected. If not, acquire the ownership before proceeding further, if required.
- Read CTM_STATUS to confirm that the module is in IDLE state.
- Set up the enhanced bus comparator unit 1 to generate an event for VPC = 0x1000.
- Set up the enhanced bus comparator unit 2 to generate an event for VPC = 0x1210.
- Set up the start and stop input selects to use comparator event 1 and 2, respectively. This is achieved by writing the value 0x0800 to CTM_INPUT_SEL register (bits 15:11 = 0b00001 and bits 10:6 = 0b00000 and all other bits are set to 0).
- Enable the counter in the START_STOP_MODE of operation. This is achieved by writing 0x4 to CTM_CNTL.
- Enable the module in the GLBL_ENABLE register.

Set up the counter to count the number of times a function at address 0x2010 is called and fire an RTOSINT if this count reaches 0x300:

- Read and make sure the ownership is set as expected. If not, acquire the ownership before proceeding further, if required.
- Read CTM_STATUS to confirm that the module is in IDLE state.
- Set up the enhanced bus comparator unit 1 to generate an event for VPC = 0x2010.
- Setup the counter to select comparator event 1 as the event to count. This is achieved by writing the value 0x0001 to CTM_INPUT_SEL register (Register bits 5:1 = 0b00000 and bit 0 = 1).
- Write 0x300 CTM_REF.
- Enable the counter in the EVENT_MODE, and also allow it to generate an RTOSINT when the count matches the reference. This is achieved by writing 0x88 to CTM_CNTL (bit 3 = 1 and bit 7 = 1).
- Enable the module in the GLBL_ENABLE register.

## 11.6 Registers

### 11.6.1 ERAD Base Addresses

**Table 11-2. ERAD Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| EnhancedDebugGlobalRegs | ERAD_GLOBAL_REGS | 0x0005_E800 | 0x0005_E80A |
| EnhancedDebugHWBP1Regs | ERAD_HWBP_REGS | 0x0005_E900 | 0x0005_E907 |
| EnhancedDebugHWBP2Regs | ERAD_HWBP_REGS | 0x0005_E908 | 0x0005_E90F |
| EnhancedDebugHWBP3Regs | ERAD_HWBP_REGS | 0x0005_E910 | 0x0005_E917 |
| EnhancedDebugHWBP4Regs | ERAD_HWBP_REGS | 0x0005_E918 | 0x0005_E91F |
| EnhancedDebugHWBP5Regs | ERAD_HWBP_REGS | 0x0005_E920 | 0x0005_E927 |
| EnhancedDebugHWBP6Regs | ERAD_HWBP_REGS | 0x0005_E928 | 0x0005_E92F |
| EnhancedDebugHWBP7Regs | ERAD_HWBP_REGS | 0x0005_E930 | 0x0005_E937 |
| EnhancedDebugHWBP8Regs | ERAD_HWBP_REGS | 0x0005_E938 | 0x0005_E93F |
| EnhancedDebugCounter1Regs | ERAD_COUNTER_REGS | 0x0005_E980 | 0x0005_E98F |
| EnhancedDebugCounter2Regs | ERAD_COUNTER_REGS | 0x0005_E990 | 0x0005_E99F |
| EnhancedDebugCounter3Regs | ERAD_COUNTER_REGS | 0x0005_E9A0 | 0x0005_E9AF |
| EnhancedDebugCounter4Regs | ERAD_COUNTER_REGS | 0x0005_E9B0 | 0x0005_E9BF |

### 11.6.1.1 ERAD_GLOBAL_REGS Registers

Table 11-3 lists the memory-mapped registers for the ERAD_GLOBAL_REGS. All register offset addresses not listed in Table 11-3 should be considered as reserved locations and the register contents should not be modified.

**Table 11-3. ERAD_GLOBAL_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | GLBL_EVENT_STAT | Global Event Status Register | | Go |
| 2h | GLBL_HALT_STAT | Global Halt Status Register | | Go |
| 4h | GLBL_ENABLE | Global Enable Register | EALLOW | Go |
| 6h | GLBL_CTM_RESET | Global Counter Reset | EALLOW | Go |
| Ah | GLBL_OWNER | Global Ownership | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 11-4 shows the codes that are used for access types in this section.

**Table 11-4. ERAD_GLOBAL_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | 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. |

#### 11.6.1.1.1 GLBL_EVENT_STAT Register (Offset = 0h) [reset = 0h]

GLBL_EVENT_STAT is shown in Figure 11-3 and described in Table 11-5.

Return to Summary Table.

This register contains one bit for each of the bus comparator modules and the counter modules that are present in a device. Each bit directly reflects the state of the EVENT_FIRED bit of the corresponding module. This facilitates software to just read one register and find out if any of the debug modules had fired.

**Figure 11-3. GLBL_EVENT_STAT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | CTM4 | CTM3 | CTM2 | CTM1 |
| R-0h | | | | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HWBP8 | HWBP7 | HWBP6 | HWBP5 | HWBP4 | HWBP3 | HWBP2 | HWBP1 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 11-5. GLBL_EVENT_STAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11 | CTM4 | R | 0h | This bit directly reflects the state of the EVENT_FIRED bit of the Counter unit 4.<br>0 No Event<br>1 Event Fired<br>Reset type: ERAD_RESET |
| 10 | CTM3 | R | 0h | This bit directly reflects the state of the EVENT_FIRED bit of the Counter unit 3.<br>0 No Event<br>1 Event Fired<br>Reset type: ERAD_RESET |
| 9 | CTM2 | R | 0h | This bit directly reflects the state of the EVENT_FIRED bit of the Counter unit 2.<br>0 No Event<br>1 Event Fired<br>Reset type: ERAD_RESET |
| 8 | CTM1 | R | 0h | This bit directly reflects the state of the EVENT_FIRED bit of the Counter unit 1.<br>0 No Event<br>1 Event Fired<br>Reset type: ERAD_RESET |
| 7 | HWBP8 | R | 0h | This bit directly reflects the state of the EVENT_FIRED bit of the Bus Comparator unit 8.<br>0 No Event<br>1 Event Fired<br>Reset type: ERAD_RESET |
| 6 | HWBP7 | R | 0h | This bit directly reflects the state of the EVENT_FIRED bit of the Bus Comparator unit 7.<br>0 No Event<br>1 Event Fired<br>Reset type: ERAD_RESET |

**Table 11-5. GLBL_EVENT_STAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | HWBP6 | R | 0h | This bit directly reflects the state of the EVENT_FIRED bit of the Bus Comparator unit 6.<br>0 No Event<br>1 Event Fired<br>Reset type: ERAD_RESET |
| 4 | HWBP5 | R | 0h | This bit directly reflects the state of the EVENT_FIRED bit of the Bus Comparator unit 5.<br>0 No Event<br>1 Event Fired<br>Reset type: ERAD_RESET |
| 3 | HWBP4 | R | 0h | This bit directly reflects the state of the EVENT_FIRED bit of the Bus Comparator unit 4.<br>0 No Event<br>1 Event Fired<br>Reset type: ERAD_RESET |
| 2 | HWBP3 | R | 0h | This bit directly reflects the state of the EVENT_FIRED bit of the Bus Comparator unit 3.<br>0 No Event<br>1 Event Fired<br>Reset type: ERAD_RESET |
| 1 | HWBP2 | R | 0h | This bit directly reflects the state of the EVENT_FIRED bit of the Bus Comparator unit 2.<br>0 No Event<br>1 Event Fired<br>Reset type: ERAD_RESET |
| 0 | HWBP1 | R | 0h | This bit directly reflects the state of the EVENT_FIRED bit of the Bus Comparator unit 1.<br>0 No Event<br>1 Event Fired<br>Reset type: ERAD_RESET |

#### 11.6.1.1.2 GLBL_HALT_STAT Register (Offset = 2h) [reset = 0h]

GLBL_HALT_STAT is shown in Figure 11-4 and described in Table 11-6.

Return to Summary Table.

This register contains one bit for each of the bus comparator modules and the counter modules that are present in a device. Each bit directly reflects the state of the Completed status bit. This facilitates software to just read one register and find out if any of the debug modules have fired.

**Figure 11-4. GLBL_HALT_STAT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | CTM4 | CTM3 | CTM2 | CTM1 |
| R-0h | | | | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HWBP8 | HWBP7 | HWBP6 | HWBP5 | HWBP4 | HWBP3 | HWBP2 | HWBP1 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 11-6. GLBL_HALT_STAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11 | CTM4 | R | 0h | This bit directly reflects the state of the completed bit of the Counter unit 4.<br>0 No Event<br>1 Event Fired<br>Reset type: ERAD_RESET |
| 10 | CTM3 | R | 0h | This bit directly reflects the state of the completed bit of the Counter unit 3.<br>0 No Event<br>1 Event Fired<br>Reset type: ERAD_RESET |
| 9 | CTM2 | R | 0h | This bit directly reflects the state of the completed bit of the Counter unit 2.<br>0 No Event<br>1 Event Fired<br>Reset type: ERAD_RESET |
| 8 | CTM1 | R | 0h | This bit directly reflects the state of the completed bit of the Counter unit 1.<br>0 No Event<br>1 Event Fired<br>Reset type: ERAD_RESET |
| 7 | HWBP8 | R | 0h | This bit directly reflects the state of the completed bit of the Bus Comparator unit 8.<br>0 Not Completed<br>1 Completed<br>Reset type: ERAD_RESET |
| 6 | HWBP7 | R | 0h | This bit directly reflects the state of the completed bit of the Bus Comparator unit 7.<br>0 Not Completed<br>1 Completed<br>Reset type: ERAD_RESET |

**Table 11-6. GLBL_HALT_STAT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | HWBP6 | R | 0h | This bit directly reflects the state of the completed bit of the Bus Comparator unit 6.<br>0 Not Completed<br>1 Completed<br>Reset type: ERAD_RESET |
| 4 | HWBP5 | R | 0h | This bit directly reflects the state of the completed bit of the Bus Comparator unit 5.<br>0 Not Completed<br>1 Completed<br>Reset type: ERAD_RESET |
| 3 | HWBP4 | R | 0h | This bit directly reflects the state of the completed bit of the Bus Comparator unit 4.<br>0 Not Completed<br>1 Completed<br>Reset type: ERAD_RESET |
| 2 | HWBP3 | R | 0h | This bit directly reflects the state of the completed bit of the Bus Comparator unit 3.<br>0 Not Completed<br>1 Completed<br>Reset type: ERAD_RESET |
| 1 | HWBP2 | R | 0h | This bit directly reflects the state of the completed bit of the Bus Comparator unit 2.<br>0 Not Completed<br>1 Completed<br>Reset type: ERAD_RESET |
| 0 | HWBP1 | R | 0h | This bit directly reflects the state of the completed bit of the Bus Comparator unit 1.<br>0 Not Completed<br>1 Completed<br>Reset type: ERAD_RESET |

#### 11.6.1.1.3 GLBL_ENABLE Register (Offset = 4h) [reset = 0h]

GLBL_ENABLE is shown in Figure 11-5 and described in Table 11-7.

Return to Summary Table.

This register contains one bit for each of the bus comparator modules and the counter modules that are present in a device. Each bit directly acts as a global enable for the corresponding module. This bit has to be set to 1 for the module to be functional.

**Figure 11-5. GLBL_ENABLE Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | CTM4 | CTM3 | CTM2 | CTM1 |
| R-0h | | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HWBP8 | HWBP7 | HWBP6 | HWBP5 | HWBP4 | HWBP3 | HWBP2 | HWBP1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 11-7. GLBL_ENABLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11 | CTM4 | R/W | 0h | This bit directly reflects the state of the ENABLE bit of the Counter unit 4.<br>0 Disabled<br>1 Enabled<br>Reset type: ERAD_RESET |
| 10 | CTM3 | R/W | 0h | This bit directly reflects the state of the ENABLE bit of the Counter unit 3.<br>0 Disabled<br>1 Enabled<br>Reset type: ERAD_RESET |
| 9 | CTM2 | R/W | 0h | This bit directly reflects the state of the ENABLE bit of the Counter unit 2.<br>0 Disabled<br>1 Enabled<br>Reset type: ERAD_RESET |
| 8 | CTM1 | R/W | 0h | This bit directly reflects the state of the ENABLE bit of the Counter unit 1.<br>0 Disabled<br>1 Enabled<br>Reset type: ERAD_RESET |
| 7 | HWBP8 | R/W | 0h | This bit directly reflects the state of the ENABLE bit of the Bus Comparator unit 8.<br>0 Disabled<br>1 Enabled<br>Reset type: ERAD_RESET |
| 6 | HWBP7 | R/W | 0h | This bit directly reflects the state of the ENABLE bit of the Bus Comparator unit 7.<br>0 Disabled<br>1 Enabled<br>Reset type: ERAD_RESET |

**Table 11-7. GLBL_ENABLE Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | HWBP6 | R/W | 0h | This bit directly reflects the state of the ENABLE bit of the Bus Comparator unit 6.<br>0 Disabled<br>1 Enabled<br>Reset type: ERAD_RESET |
| 4 | HWBP5 | R/W | 0h | This bit directly reflects the state of the ENABLE bit of the Bus Comparator unit 5.<br>0 Disabled<br>1 Enabled<br>Reset type: ERAD_RESET |
| 3 | HWBP4 | R/W | 0h | This bit directly reflects the state of the ENABLE bit of the Bus Comparator unit 4.<br>0 Disabled<br>1 Enabled<br>Reset type: ERAD_RESET |
| 2 | HWBP3 | R/W | 0h | This bit directly reflects the state of the ENABLE bit of the Bus Comparator unit 3.<br>0 Disabled<br>1 Enabled<br>Reset type: ERAD_RESET |
| 1 | HWBP2 | R/W | 0h | This bit directly reflects the state of the ENABLE bit of the Bus Comparator unit 2.<br>0 Disabled<br>1 Enabled<br>Reset type: ERAD_RESET |
| 0 | HWBP1 | R/W | 0h | This bit directly reflects the state of the ENABLE bit of the Bus Comparator unit 1.<br>0 Disabled<br>1 Enabled<br>Reset type: ERAD_RESET |

#### 11.6.1.1.4 GLBL_CTM_RESET Register (Offset = 6h) [reset = 0h]

GLBL_CTM_RESET is shown in Figure 11-6 and described in Table 11-8.

Return to Summary Table.

This register contains one bit for each of the counter modules that are present in a device. Each bit directly acts as a reset for the counters for the corresponding module.(It does not affect anything else except resetting the counter.
Example: If the counter was previously incrementing before reset, then on a reset event the counter gets reset and continues to increment again).

**Figure 11-6. GLBL_CTM_RESET Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | CTM4 | CTM3 | CTM2 | CTM1 |
| R-0h | | | | R=0/W-0h | R=0/W-0h | R=0/W-0h | R=0/W-0h |

**Table 11-8. GLBL_CTM_RESET Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | CTM4 | R=0/W | 0h | This bit directly resets the state the Counter unit 4.<br>0 No Effect<br>1 Reset<br>Reset type: ERAD_RESET |
| 2 | CTM3 | R=0/W | 0h | This bit directly resets the state the Counter unit 3.<br>0 No Effect<br>1 Reset<br>Reset type: ERAD_RESET |
| 1 | CTM2 | R=0/W | 0h | This bit directly resets the state the Counter unit 2.<br>0 No Effect<br>1 Reset<br>Reset type: ERAD_RESET |
| 0 | CTM1 | R=0/W | 0h | This bit directly resets the state the Counter unit 1.<br>0 No Effect<br>1 Reset<br>Reset type: ERAD_RESET |

#### 11.6.1.1.5 GLBL_OWNER Register (Offset = Ah) [reset = 0h]

GLBL_OWNER is shown in Figure 11-7 and described in Table 11-9.

Return to Summary Table.

Global Ownership

**Figure 11-7. GLBL_OWNER Register**

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

**Table 11-9. GLBL_OWNER Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-2 | RESERVED | R | 0h | Reserved |
| 1-0 | OWNER | R/W | 0h | This register determines whether Application Code or Debugger owns this module or it's kept in No Owner state where debugger or application can access the module.<br>00 No Owner<br>01 Application owned<br>10 Debugger owned<br>11 Reserved<br>Reset type: ERAD_RESET |

### 11.6.1.2 ERAD_HWBP_REGS Registers

Table 11-10 lists the memory-mapped registers for the ERAD_HWBP_REGS. All register offset addresses not listed in Table 11-10 should be considered as reserved locations and the register contents should not be modified.

**Table 11-10. ERAD_HWBP_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | HWBP_MASK | HWBP Mask Register | EALLOW | Go |
| 2h | HWBP_REF | HWBP Reference Register | EALLOW | Go |
| 4h | HWBP_CLEAR | HWBP Clear Register | EALLOW | Go |
| 6h | HWBP_CNTL | HWBP Control Register | EALLOW | Go |
| 7h | HWBP_STATUS | HWBP Status Register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 11-11 shows the codes that are used for access types in this section.

**Table 11-11. ERAD_HWBP_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | 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. |

#### 11.6.1.2.1 HWBP_MASK Register (Offset = 0h) [reset = 0h]

HWBP_MASK is shown in Figure 11-8 and described in Table 11-12.

Return to Summary Table.

HWBP Mask Register

**Figure 11-8. HWBP_MASK Register**

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

**Table 11-12. HWBP_MASK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | MASK | R/W | 0h | This register contains address mask for comparison. The contents of this register are used along with the reference register to determine the address match. The equation used to determine a match is as follows. Match is true if,<br>(address \| mask) == (ref \| mask)<br>This register is writable by CPU only if application owns the unit and if EALLOW is set. Otherwise, the writes are ignored. The register is writable by the debugger only if the debugger owns this unit. Otherwise, the writes are ignored.<br>Reset type: ERAD_RESET |

#### 11.6.1.2.2 HWBP_REF Register (Offset = 2h) [reset = 0h]

HWBP_REF is shown in Figure 11-9 and described in Table 11-13.

Return to Summary Table.

HWBP Reference Register

**Figure 11-9. HWBP_REF Register**

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

**Table 11-13. HWBP_REF Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | REF | R/W | 0h | This register contains the reference address for comparison. The contents of this register are used along with the mask register to determine the address match. The equation used to determine a match is as follows. Match is true if,<br>(address \| mask) == (ref \| mask)<br>This register is writable by CPU only if application owns the unit and if EALLOW is set. Otherwise, the writes are ignored. The register is writable by the debugger only if the debugger owns this unit. Otherwise, the writes are ignored.<br>Reset type: ERAD_RESET |

#### 11.6.1.2.3 HWBP_CLEAR Register (Offset = 4h) [reset = 0h]

HWBP_CLEAR is shown in Figure 11-10 and described in Table 11-14.

Return to Summary Table.

HWBP Clear Register

**Figure 11-10. HWBP_CLEAR Register**

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

**Table 11-14. HWBP_CLEAR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-1 | RESERVED | R | 0h | Reserved |
| 0 | EVENT_CLR | R=0/W | 0h | Event Clear register:<br>0 No action.<br>1 A write with this bit set to 1 will clear the sticky EVENT_FIRED bit in the HWBP_STATUS register and bring the Breakpoint Module statemachine status back to IDLE.<br>Reads of this bit position will always return a 0.<br>Reset type: ERAD_RESET |

#### 11.6.1.2.4 HWBP_CNTL Register (Offset = 6h) [reset = 0h]

HWBP_CNTL is shown in Figure 11-11 and described in Table 11-15.

Return to Summary Table.

HWBP Control Register

**Figure 11-11. HWBP_CNTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | RESERVED | RESERVED | COMP_MODE | |
| R-0h | | | | R-0h | R-0h | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| COMP_MODE | RTOSINT | STOP | BUS_SEL | | | RESERVED | |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | | | R-0h | |

**Table 11-15. HWBP_CNTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11 | RESERVED | R | 0h | Reserved |
| 10 | RESERVED | R | 0h | Reserved |
| 9-7 | COMP_MODE | R/W | 0h | Bus comparator compare modes:<br>000 Regular masked compare<br>HWBP_MSK will be ignored for the following modes:<br>100 Bus value GT HWBP_REF<br>101 Bus value GE HWBP_REF<br>110 Bus value LT HWBP_REF<br>111 Bus value LE HWBP_REF<br>GT means Greater Than<br>GE means Greater or Equal<br>LT means Less Than<br>LE means Lesser or Equal<br>Reset type: ERAD_RESET |
| 6 | RTOSINT | R/W | 0h | This bit decides whether the bus comparator unit will generate RTOSINTn interrupt when event matches occur.<br>Note that the event outputs will always be generated regardless of the state of this bit.<br>0 The bus comparator unit will not cause any action towards the CPU.<br>1 The bus comparator unit will assert RTOSINTn for matching data accesses and trace tags for matching program fetches.<br>Reset type: ERAD_RESET |
| 5 | STOP | R/W | 0h | This bit decides whether the bus comparator unit will generate CPU halting signals when event matches occur. Note that the event outputs will always be generated regardless of the state of this bit.<br>0 The bus comparator unit will not cause any action towards halting the CPU.<br>1 The bus comparator unit will assert ANASTOP for matching data accesses and break tags for matching program fetches. These can cause the CPU to HALT<br>Reset type: ERAD_RESET |

**Table 11-15. HWBP_CNTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4-2 | BUS_SEL | R/W | 0h | These bits are used to select which CPU buses will be used for comparison to generate the match events. For each bus selected, the corresponding strobes will automatically be selected to determine valid accesses.<br>000 PAB for instruction fetches<br>011 DRAB for data read accesses<br>010 DWAB for data write accesses<br>001 VPC for Program counter match<br>100 DWDB for write data match<br>All other combinations are RESERVED.<br>Reset type: ERAD_RESET |
| 1-0 | RESERVED | R | 0h | Reserved |

#### 11.6.1.2.5 HWBP_STATUS Register (Offset = 7h) [reset = 400h]

HWBP_STATUS is shown in Figure 11-12 and described in Table 11-16.

Return to Summary Table.

HWBP Status Register

**Figure 11-12. HWBP_STATUS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| STATUS | | MODULE_ID | | | | | |
| R-0h | | R-4h | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | EVENT_FIRED |
| R-0h | | | | | | | R-0h |

**Table 11-16. HWBP_STATUS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | STATUS | R | 0h | Bus comaparator status:<br>00 Idle<br>10 Enabled<br>11 Completed<br>Reset type: ERAD_RESET |
| 13-8 | MODULE_ID | R | 4h | These bits are always a constant representing a unique identification for the bus comparator unit.<br>Reset type: ERAD_RESET |
| 7-1 | RESERVED | R | 0h | Reserved |
| 0 | EVENT_FIRED | R | 0h | This is a sticky bit which gets set every time the HWBP unit generates a match event. This will be used by software to figure out whether this HWBP module fired an event or not. This bit will get cleared by writing a '1' to bit 0 of the HWBP_CLEAR register.<br>Reset type: ERAD_RESET |

### 11.6.1.3 ERAD_COUNTER_REGS Registers

Table 11-17 lists the memory-mapped registers for the ERAD_COUNTER_REGS. All register offset addresses not listed in Table 11-17 should be considered as reserved locations and the register contents should not be modified.

**Table 11-17. ERAD_COUNTER_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | CTM_CNTL | Counter Control Register | EALLOW | Go |
| 1h | CTM_STATUS | Counter Status Register | EALLOW | Go |
| 2h | CTM_REF | Counter Reference Register | EALLOW | Go |
| 4h | CTM_COUNT | Counter Current Value Register | EALLOW | Go |
| 6h | CTM_MAX_COUNT | Counter Max Count Value Register | EALLOW | Go |
| 8h | CTM_INPUT_SEL | Counter Input Select Register | EALLOW | Go |
| 9h | CTM_CLEAR | Counter Clear Register | EALLOW | Go |
| Ah | CTM_INPUT_SEL_MSB | Counter Input Select Extension Register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 11-18 shows the codes that are used for access types in this section.

**Table 11-18. ERAD_COUNTER_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | 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. |

#### 11.6.1.3.1 CTM_CNTL Register (Offset = 0h) [reset = 0h]

CTM_CNTL is shown in Figure 11-13 and described in Table 11-19.

Return to Summary Table.

Counter Control Register

**Figure 11-13. CTM_CNTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RST_INP_SEL | | | | | RST_EN | RESERVED | START_STOP_CUMULATIVE |
| R/W-0h | | | | | R/W-0h | R-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RTOSINT | STOP | RESERVED | RST_ON_MATCH | EVENT_MODE | START_STOP_MODE | RESERVED | |
| R/W-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h | R-0h | |

**Table 11-19. CTM_CNTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-11 | RST_INP_SEL | R/W | 0h | These 5 bits are used to select the event input that will be used as the reset input. These bits matter only if the Enable Reset bit is set to 1.<br>Reset type: ERAD_RESET |
| 10 | RST_EN | R/W | 0h | This bit decides if the reset input is enabled or not. Setting this to 1 will cause the counter to reset to zero whenever the selected reset input<br>goes active high. No event will be generated when the counter is reset. Setting this bit to 0 will cause the counter to ignore the reset inputs.<br>Reset type: ERAD_RESET |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | START_STOP_CUMULATIVE | R/W | 0h | This bit decides whether the counter counts to give the cumulative cycle count for 'n' number of successive start stop events or clears the counter on very stop event to record the MAX_COUNT across successive start stop sequences.<br>0 When in START_STOP mode counter gets cleared on every stop event and MAX_COUNT records the max value<br>1 When in START_STOP mode counter keeps counting between successive start stop events to generate a cumulative count w/o clearing the counter on any stop events. MAX_COUNT register is invalid when this bit is set.<br>Reset type: ERAD_RESET |
| 7 | RTOSINT | R/W | 0h | This bit decides whether the counter module will generate RTOSINTn interrupt when count value matches the reference.<br>Note that the event outputs will always be generated regardless of the state of this bit.<br>0 The counter unit will not cause any action towards the CPU.<br>1 The counter unit will assert RTOSINTn when the count value matches the reference value.<br>Reset type: ERAD_RESET |

**Table 11-19. CTM_CNTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | STOP | R/W | 0h | This bit decides whether the counter module will generate a watchpoint to the CPU when the count value matches the reference.<br>Note that the event outputs will always be generated regardless of the state of this bit.<br>0 The counter unit will not generate a watchpoint.<br>1 The counter unit will assert ANASTOP when the count value matches the reference.<br>Reset type: ERAD_RESET |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | RST_ON_MATCH | R/W | 0h | This bit is used to decide whether the counter will reset to zero once it reaches the reference value.<br>0 Counter will stay at the reference value and the counter will go to COMPLETED state and further counting will be stopped.<br>1 The counter will reset to zero once it reaches the match value and will stay enabled.<br>Reset type: ERAD_RESET |
| 3 | EVENT_MODE | R/W | 0h | This bit is used to decide whether the counter will count the level of the event or the edge of the event.<br>0 Counter will increment the count as long as the count input is active high.<br>1 The counter will count only on the rising edge of the count input.<br>Reset type: ERAD_RESET |
| 2 | START_STOP_MODE | R/W | 0h | This bit is used to decide whether the counter will count in the START_STOP mode or not.<br>0 Normal count mode. The counter will not depend on the START and STOP events<br>1 This is the START-STOP mode of the counter. The counter will start counting only after the START input has been asserted. It will continue to count the selected event till the STOP event is seen.<br>Reset type: ERAD_RESET |
| 1-0 | RESERVED | R | 0h | Reserved |

#### 11.6.1.3.2 CTM_STATUS Register (Offset = 1h) [reset = 10h]

CTM_STATUS is shown in Figure 11-14 and described in Table 11-20.

Return to Summary Table.

Counter Status Register

**Figure 11-14. CTM_STATUS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| STATUS | | | | MODULE_ID | | | |
| R-0h | | | | R-4h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MODULE_ID | | | | | | OVERFLOW | EVENT_FIRED |
| R-4h | | | | | | R-0h | R-0h |

**Table 11-20. CTM_STATUS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | STATUS | R | 0h | Counter unit status,<br>00 Idle<br>10 Enabled<br>11 Completed<br>Reset type: ERAD_RESET |
| 11-2 | MODULE_ID | R | 4h | These bits are always a constant representing a unique identification for the trigger unit.<br>Reset type: ERAD_RESET |
| 1 | OVERFLOW | R | 0h | This is a sticky bit which gets set every time the counter overflows and wraps around after reaching 0xffffffff.This bit will get cleared by writing a '1' to bit 9 of the CTM_CNTL register.<br>Reset type: ERAD_RESET |
| 0 | EVENT_FIRED | R | 0h | This is a sticky bit which gets set every time the CTM unit generates a match event. This will be used by software to figure out whether this CTM module fired an event or not. This bit will get cleared by writing a '1' to bit 9 of the CTM_CNTL register.<br>Reset type: ERAD_RESET |

#### 11.6.1.3.3 CTM_REF Register (Offset = 2h) [reset = 0h]

CTM_REF is shown in Figure 11-15 and described in Table 11-21.

Return to Summary Table.

Counter Reference Register

**Figure 11-15. CTM_REF Register**

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

**Table 11-21. CTM_REF Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | REF | R/W | 0h | This register contains the counter reference value for comparison. The counter will generate an event if the count value matches the reference register (considering both upper and lower half of the register).<br>This register is writable by CPU only if application owns the unit and if EALLOW is set. Otherwise, the writes are ignored. The register is writable by the debugger only if the debugger owns this unit. Otherwise, the writes are ignored.<br>Refrence match is enabled only when a non zero value is programmed on one of the REF_HIGH or REF_LOW register.<br>Reset type: ERAD_RESET |

#### 11.6.1.3.4 CTM_COUNT Register (Offset = 4h) [reset = 0h]

CTM_COUNT is shown in Figure 11-16 and described in Table 11-22.

Return to Summary Table.

Counter Current Value Register

**Figure 11-16. CTM_COUNT Register**

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

**Table 11-22. CTM_COUNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | COUNT | R/W | 0h | This register contains the current count value.<br>The counter will generate an event if the count value matches the reference register (considering both upper and lower half of the register).<br>This register is writable by CPU only if application owns the unit and if EALLOW is set. Otherwise, the writes are ignored.<br>The register is writable by the debugger only if the debugger owns this unit. Otherwise, the writes are ignored.<br>Reset type: ERAD_RESET |

#### 11.6.1.3.5 CTM_MAX_COUNT Register (Offset = 6h) [reset = 0h]

CTM_MAX_COUNT is shown in Figure 11-17 and described in Table 11-23.

Return to Summary Table.

Counter Max Count Value Register

**Figure 11-17. CTM_MAX_COUNT Register**

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

**Table 11-23. CTM_MAX_COUNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | MAX_COUNT | R/W | 0h | This register contains the maximum recorded counter value. This is relevant only in the Start Stop mode of operation.<br>This register is writable by CPU only if application owns the unit and if EALLOW is set. Otherwise, the writes are ignored.<br>The register is writable by the debugger only if the debugger owns this unit. Otherwise, the writes are ignored.<br>Reset type: ERAD_RESET |

#### 11.6.1.3.6 CTM_INPUT_SEL Register (Offset = 8h) [reset = 0h]

CTM_INPUT_SEL is shown in Figure 11-18 and described in Table 11-24.

Return to Summary Table.

Counter Input Select Register

**Figure 11-18. CTM_INPUT_SEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| STO_INP_SEL | | | | | STA_INP_SEL | | |
| R/W-0h | | | | | R/W-0h | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| STA_INP_SEL | | CNT_INP_SEL | | | | | CTM_INP_SEL_EN |
| R/W-0h | | R/W-0h | | | | | R/W-0h |

**Table 11-24. CTM_INPUT_SEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-11 | STO_INP_SEL | R/W | 0h | These 5 bits decide which of the 32 inputs will be selected as the STOP event for the counter. The 32 inputs will be hooked up to the event outputs from the breakpoint module and to other system events. The usage of these bits are relevant only in the START_STOP mode of counting.<br>Reset type: ERAD_RESET |
| 10-6 | STA_INP_SEL | R/W | 0h | These 5 bits decide which of the 32 inputs will be selected as the START event for the counter. The 32 inputs will be hooked up to the event outputs from the breakpoint module and to other system events. The usage of these bits are relevant only in the START_STOP mode of counting.<br>Reset type: ERAD_RESET |
| 5-1 | CNT_INP_SEL | R/W | 0h | These 5 bits decide which of the 32 inputs will be selected to enable counting. The 32 inputs will be hooked up to the event outputs from the breakpoint module and to other system events.<br>Reset type: ERAD_RESET |
| 0 | CTM_INP_SEL_EN | R/W | 0h | Counter input select enable:<br>0 Disable using the input_select register for the count input. The counter will always count CPU cycles.<br>1 Enable using the input_select register for the count input. The counter will count the event selected by the count input register.<br>Reset type: ERAD_RESET |

#### 11.6.1.3.7 CTM_CLEAR Register (Offset = 9h) [reset = 0h]

CTM_CLEAR is shown in Figure 11-19 and described in Table 11-25.

Return to Summary Table.

Counter Clear Register

**Figure 11-19. CTM_CLEAR Register**

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

**Table 11-25. CTM_CLEAR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-2 | RESERVED | R | 0h | Reserved |
| 1 | OVERFLOW_CLEAR | R=0/W | 0h | Clear OVERFLOW:<br>0 No action.<br>1 A write with this bit set to 1 will clear the sticky OVERFLOW bit in the CTM_STATUS register.<br>Reads of this bit position will always return a 0.<br>Reset type: ERAD_RESET |
| 0 | EVENT_CLEAR | R=0/W | 0h | Clear EVENT_FIRED:<br>0 No action.<br>1 A write with this bit set to 1 will clear the sticky EVENT_FIRED bit in the CTM_STATUS register and bring the Breakpoint Module statemachine status back to IDLE.<br>Reads of this bit position will always return a 0.<br>Reset type: ERAD_RESET |

#### 11.6.1.3.8 CTM_INPUT_SEL_MSB Register (Offset = Ah) [reset = 0h]

CTM_INPUT_SEL_MSB is shown in Figure 11-20 and described in Table 11-26.

Return to Summary Table.

Counter Input Select Extension Register

**Figure 11-20. CTM_INPUT_SEL_MSB Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | RST_INP_SEL_MSB | RESERVED | | | STO_INP_SEL_MSB |
| R-0h | | | R/W-0h | R-0h | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | STA_INP_SEL_MSB | RESERVED | | | CTM_INP_SEL_MSB |
| R-0h | | | R/W-0h | R-0h | | | R/W-0h |

**Table 11-26. CTM_INPUT_SEL_MSB Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12 | RST_INP_SEL_MSB | R/W | 0h | Reset type: ERAD_RESET |
| 11-9 | RESERVED | R | 0h | Reserved |
| 8 | STO_INP_SEL_MSB | R/W | 0h | Reset type: ERAD_RESET |
| 7-5 | RESERVED | R | 0h | Reserved |
| 4 | STA_INP_SEL_MSB | R/W | 0h | Reset type: ERAD_RESET |
| 3-1 | RESERVED | R | 0h | Reserved |
| 0 | CTM_INP_SEL_MSB | R/W | 0h | Reset type: ERAD_RESET |

# Analog Subsystem

This analog subsystem module is described in this chapter.

**Topic** **Page**

## 12.1 Analog Subsystem

The analog modules on this device include the Analog-to-Digital Converters (ADCs), Temperature Sensor, Buffered Digital-to-Analog Converter (DAC), Programmable Gain Amplifier (PGA), and Comparator Subsystem (CMPSS).

### 12.1.1 Features

The analog subsystem has the following features:

- Flexible voltage references
  - The ADCs are referenced to VREFHIx and VREFLOx pins
    - VREFHIx pin voltage can be driven in externally or can be generate by an internal bandgap voltage reference
    - The internal voltage reference range can be selected to be 0V to 3.3V or 0V to 2.5V
  - The buffered DACs are referenced to VREFHIx and VREFLOx
    - Alternately, these DACs can be referenced to the VDAC pin and VSSA
  - The comparator DACs are referenced to VDDA and VSSA
    - Alternately, these DACs can be referenced to the VDAC pin and VSSA
- Flexible pin usage
  - Buffered DAC outputs, comparator subsystem inputs, PGA functions, and digital inputs are multiplexed with ADC inputs
  - Internal connection to $V_{REFLO}$ on all ADCs for offset self-calibration

### 12.1.2 Block Diagram

The subsystem block diagrams show the connections between the different integrated analog modules and to the device pins. These pins fall into two categories: analog module inputs/outputs and reference pins.

The analog pins are organized into analog groups around a PGA and CMPSS module. The block diagram shows which pins connect to each group, but does not show the specific connections from the pins to the ADC, DAC, CMPSS, or PGA modules.

The block diagrams for the analog subsystem are presented in the following graphics.

Figure 12-1 shows the Analog Subsystem Block Diagram for the 100-pin PZ LQFP.

Figure 12-2 shows the Analog Subsystem Block Diagram for the 64-pin PM LQFP.

Figure 12-3 shows the Analog Subsystem Block Diagram for the 56-pin RSH VQFN.

**Figure 12-1. Analog Subsystem Block Diagram (100-Pin PZ LQFP)**

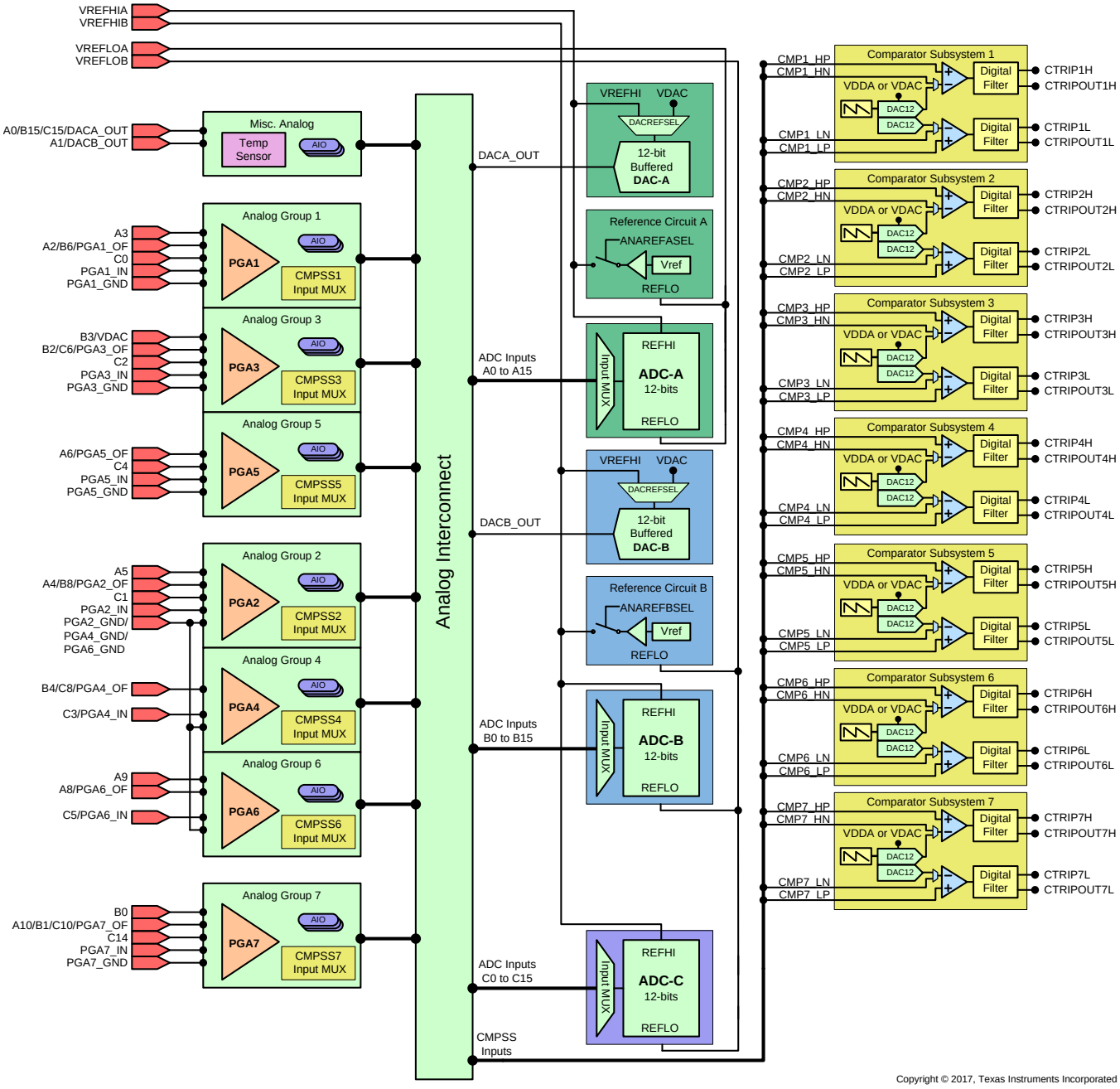


**Figure 12-2. Analog Subsystem Block Diagram (64-Pin PM LQFP)**

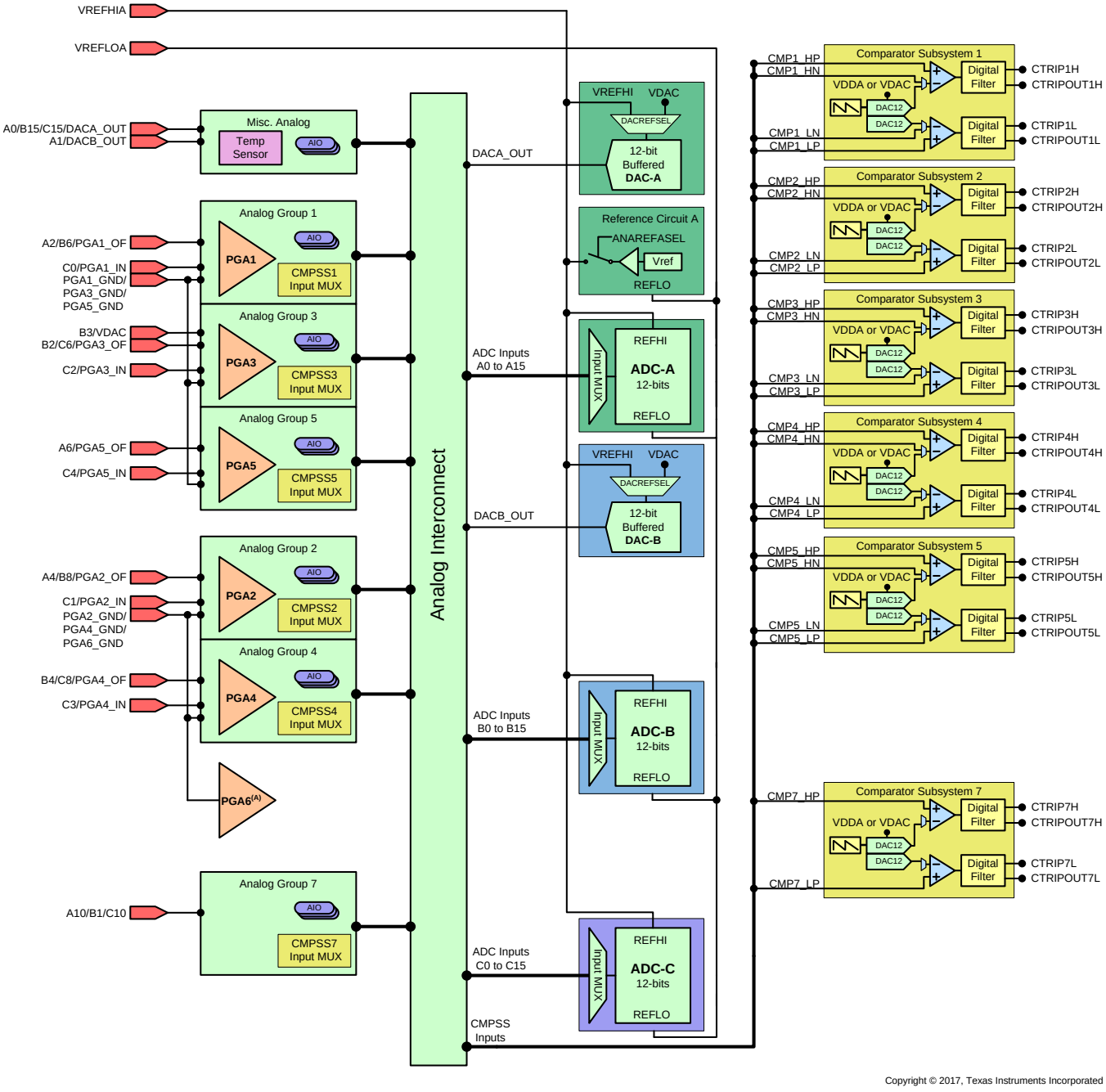


**Figure 12-3. Analog Subsystem Block Diagram (56-Pin RSH VQFN)**

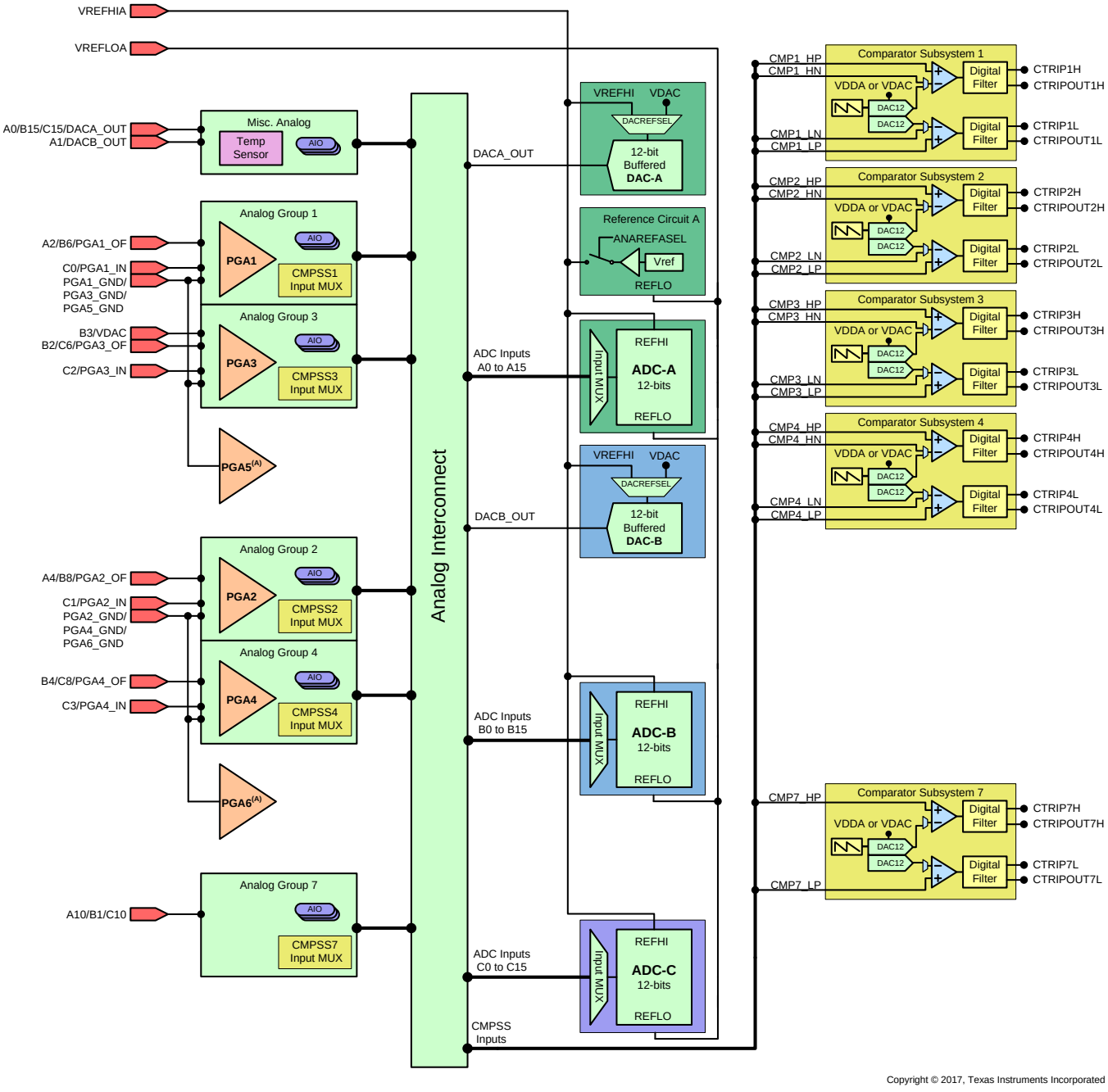
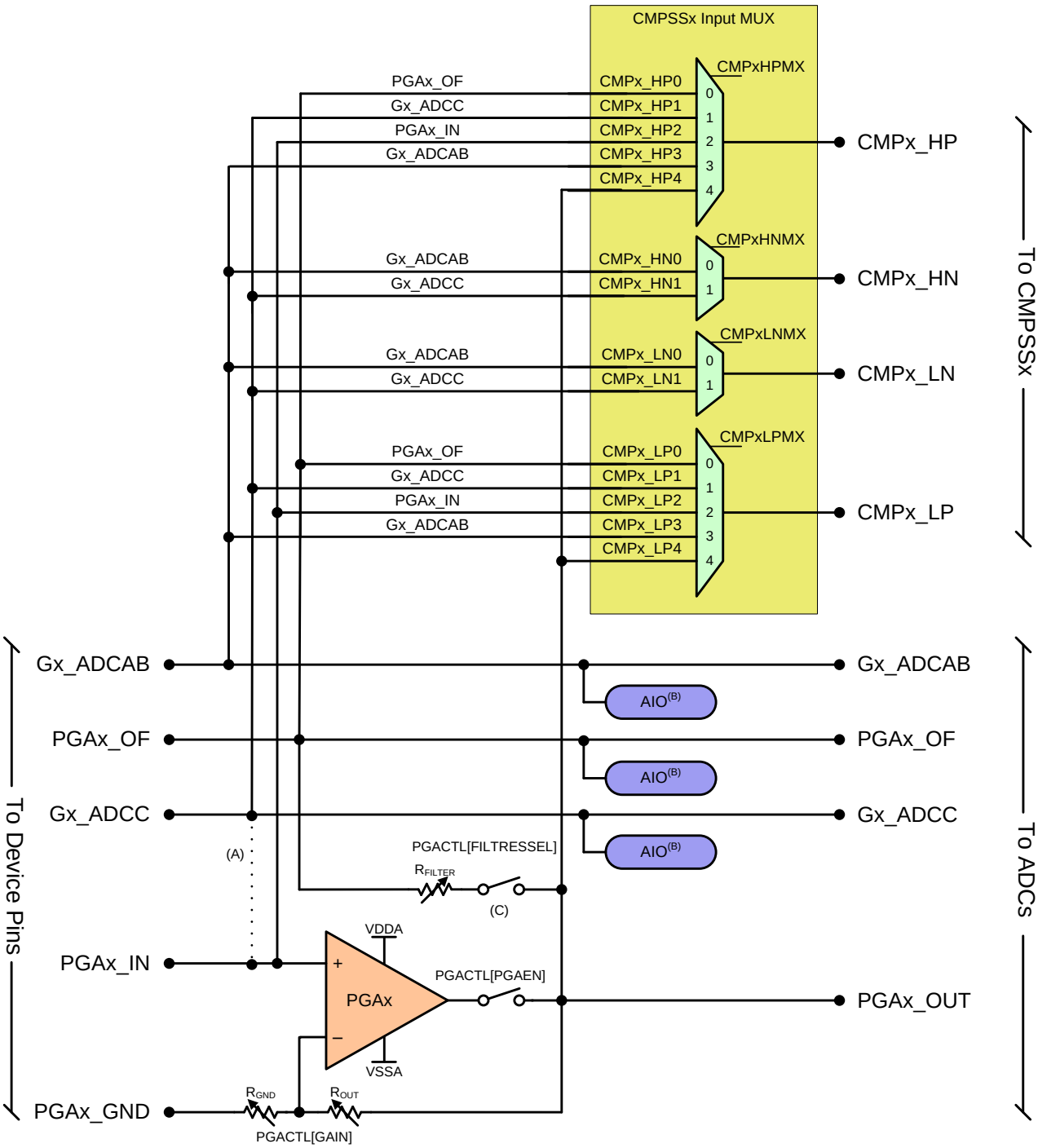


Figure 12-4 shows how each analog group is structured.

**Figure 12-4. Analog Group Connections**




A On lower pin-count packages, the input to ADCC will share a pin with the PGA input. If the PGA input is unused, then the ADCC input can allow the pin to be used as an ADC input, a negative comparator input, or a digital input.

B AIOs support digital input mode only.

## Table 12-1. Analog Pins and Internal Connections

| PIN NAME | GROUP NAME | PACKAGE | | | ALWAYS CONNECTED (NO MUX) | | | | | COMPARATOR SUBSYSTEM (MUX) | | | | AIO INPUT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 100 PZ | 64 PM | 56 RSH | ADCA | ADCB | ADCC | PGA | DAC | HIGH POSITIVE | HIGH NEGATIVE | LOW POSITIVE | LOW NEGATIVE | |
| VREFHIA | - | 25 | 16 | 14 | | | | | | | | | | |
| VREFHIB | - | 24 | | | | | | | | | | | | |
| VREFHIC | - | | | | | | | | | | | | | |
| VREFLOA | - | 27 | 17 | 15 | A13 | | | | | | | | | |
| VREFLOB | - | 26 | | | | B13 | | | | | | | | |
| VREFLOC | - | | | | | | C13 | | | | | | | |
| **Analog Group 1** | | | | | | | | | | **CMP1** | | | | |
| A3 | G1_ADCAB | 10 | | | A3 | | | | | HPMXSEL = 3 | HNMXSEL = 0 | LPMXSEL = 3 | LNMXSEL = 0 | AIO233 |
| A2/B6/PGA1_OF | PGA1_OF | 9 | 9 | 8 | A2 | B6 | | PGA1_OF | | HPMXSEL = 0 | | LPMXSEL = 0 | | AIO224 |
| C0 | G1_ADCC | 19 | 12 | 10 | | | C0 | | | HPMXSEL = 1 | HNMXSEL = 1 | LPMXSEL = 1 | LNMXSEL = 1 | AIO237 |
| PGA1_IN | PGA1_IN | 18 | | | | | | PGA1_IN | | HPMXSEL = 2 | | LPMXSEL = 2 | | |
| PGA1_GND | PGA1_GND | 14 | 10 | 9 | | | | PGA1_GND | | | | | | |
| - | PGA1_OUT(1) | | | | A11 | B7 | | PGA1_OUT | | HPMXSEL = 4 | | LPMXSEL = 4 | | |
| **Analog Group 2** | | | | | | | | | | **CMP2** | | | | |
| A5 | G2_ADCAB | 35 | | | A5 | | | | | HPMXSEL = 3 | HNMXSEL = 0 | LPMXSEL = 3 | LNMXSEL = 0 | AIO234 |
| A4/B8/PGA2_OF | PGA2_OF | 36 | 23 | 21 | A4 | B8 | | PGA2_OF | | HPMXSEL = 0 | | LPMXSEL = 0 | | AIO225 |
| C1 | G2_ADCC | 29 | 18 | 16 | | | C1 | | | HPMXSEL = 1 | HNMXSEL = 1 | LPMXSEL = 1 | LNMXSEL = 1 | AIO238 |
| PGA2_IN | PGA2_IN | 30 | | | | | | PGA2_IN | | HPMXSEL = 2 | | LPMXSEL = 2 | | |
| PGA2_GND | PGA2_GND | 32 | 20 | 18 | | | | PGA2_GND | | | | | | |
| - | PGA2_OUT(1) | | | | A12 | B9 | | PGA2_OUT | | HPMXSEL = 4 | | LPMXSEL = 4 | | |
| **Analog Group 3** | | | | | | | | | | **CMP3** | | | | |
| B3/VDAC | G3_ADCAB | 8 | 8 | 7 | | B3 | | | VDAC | HPMXSEL = 3 | HNMXSEL = 0 | LPMXSEL = 3 | LNMXSEL = 0 | AIO242 |
| B2/C6/PGA3_OF | PGA3_OF | 7 | 7 | 6 | | B2 | C6 | PGA3_OF | | HPMXSEL = 0 | | LPMXSEL = 0 | | AIO226 |
| C2 | G3_ADCC | 21 | 13 | 11 | | | C2 | | | HPMXSEL = 1 | HNMXSEL = 1 | LPMXSEL = 1 | LNMXSEL = 1 | AIO244 |
| PGA3_IN | PGA3_IN | 20 | | | | | | PGA3_IN | | HPMXSEL = 2 | | LPMXSEL = 2 | | |
| PGA3_GND | PGA3_GND | 15 | 10 | 9 | | | | PGA3_GND | | | | | | |
| - | PGA3_OUT(1) | | | | | B10 | C7 | PGA3_OUT | | HPMXSEL = 4 | | LPMXSEL = 4 | | |
| **Analog Group 4** | | | | | | | | | | **CMP4** | | | | |
| B5 | G4_ADCAB | | | | | B5 | | | | HPMXSEL = 3 | HNMXSEL = 0 | LPMXSEL = 3 | LNMXSEL = 0 | AIO243 |
| B4/C8/PGA4_OF | PGA4_OF | 39 | 24 | 22 | | B4 | C8 | PGA4_OF | | HPMXSEL = 0 | | LPMXSEL = 0 | | AIO227 |
| C3 | G4_ADCC | 31 | 19 | 17 | | | C3 | | | HPMXSEL = 1 | HNMXSEL = 1 | LPMXSEL = 1 | LNMXSEL = 1 | AIO245 |
| PGA4_IN | PGA4_IN | | | | | | | PGA4_IN | | HPMXSEL = 2 | | LPMXSEL = 2 | | |
| PGA4_GND | PGA4_GND | 32 | 20 | 18 | | | | PGA4_GND | | | | | | |
| - | PGA4_OUT(1) | | | | | B11 | C9 | PGA4_OUT | | HPMXSEL = 4 | | LPMXSEL = 4 | | |

(1) Internal connection only; does not come to a device pin.

## Table 12-1. Analog Pins and Internal Connections (continued)

| PIN NAME | GROUP NAME | PACKAGE | | | ALWAYS CONNECTED (NO MUX) | | | | | COMPARATOR SUBSYSTEM (MUX) | | | | AIO INPUT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 100 PZ | 64 PM | 56 RSH | ADCA | ADCB | ADCC | PGA | DAC | HIGH POSITIVE | HIGH NEGATIVE | LOW POSITIVE | LOW NEGATIVE | |
| **Analog Group 5** | | | | | | | | | | **CMP5** | | | | |
| A7 | G5_ADCAB | | | | A7 | | | | | HPMXSEL = 3 | HNMXSEL = 0 | LPMXSEL = 3 | LNMXSEL = 0 | AIO235 |
| A6/PGA5_OF | PGA5_OF | 6 | 6 | | A6 | | | PGA5_OF | | HPMXSEL = 0 | | LPMXSEL = 0 | | AIO228 |
| C4 | G5_ADCC | 17 | 11 | | | | C4 | | | HPMXSEL = 1 | HNMXSEL = 1 | LPMXSEL = 1 | LNMXSEL = 1 | AIO239 |
| PGA5_IN | PGA5_IN | 16 | | | | | | PGA5_IN | | HPMXSEL = 2 | | LPMXSEL = 2 | | |
| PGA5_GND | PGA5_GND | 13 | 10 | 9 | | | | PGA5_GND | | | | | | |
| - | PGA5_OUT[2] | | | | A14 | | | PGA5_OUT | | HPMXSEL = 4 | | LPMXSEL = 4 | | |
| **Analog Group 6** | | | | | | | | | | **CMP6** | | | | |
| A9 | G6_ADCAB | 38 | | | A9 | | | | | HPMXSEL = 3 | HNMXSEL = 0 | LPMXSEL = 3 | LNMXSEL = 0 | AIO236 |
| A8/PGA6_OF | PGA6_OF | 37 | | | A8 | | | PGA6_OF | | HPMXSEL = 0 | | LPMXSEL = 0 | | AIO229 |
| C5 | G6_ADCC | 28 | | | | | C5 | | | HPMXSEL = 1 | HNMXSEL = 1 | LPMXSEL = 1 | LNMXSEL = 1 | AIO240 |
| PGA6_IN | PGA6_IN | | | | | | | PGA6_IN | | HPMXSEL = 2 | | LPMXSEL = 2 | | |
| PGA6_GND | PGA6_GND | 32 | 20 | 18 | | | | PGA6_GND | | | | | | |
| - | PGA6_OUT[2] | | | | A15 | | | PGA6_OUT | | HPMXSEL = 4 | | LPMXSEL = 4 | | |
| **Analog Group 7** | | | | | | | | | | **CMP7** | | | | |
| B0 | G7_ADCAB | 41 | | | | B0 | | | | HPMXSEL = 3 | HNMXSEL = 0 | LPMXSEL = 3 | LNMXSEL = 0 | AIO241 |
| A10/B1/C10/PGA7_OF | PGA7_OF[3] | 40 | 25 | 23 | A10 | B1 | C10 | PGA7_OF | | HPMXSEL = 0 | | LPMXSEL = 0 | | AIO230 |
| C14 | G7_ADCC | 44 | | | | | C14 | | | HPMXSEL = 1 | HNMXSEL = 1 | LPMXSEL = 1 | LNMXSEL = 1 | AIO246 |
| PGA7_IN | PGA7_IN | 43 | | | | | | PGA7_IN | | HPMXSEL = 2 | | LPMXSEL = 2 | | |
| PGA7_GND | PGA7_GND | 42 | | | | | | PGA7_GND | | | | | | |
| - | PGA7_OUT[2] | | | | | B12 | C11 | PGA7_OUT | | HPMXSEL = 4 | | LPMXSEL = 4 | | |
| **Other Analog** | | | | | | | | | | | | | | |
| A0/B15/C15/DACA_OUT | | 23 | 15 | 13 | A0 | B15 | C15 | | DACA_OUT | | | | | AIO231 |
| A1/DACB_OUT | | 22 | 14 | 12 | A1 | | | | DACB_OUT | | | | | AIO232 |
| C12 | | | | | | | C12 | | | | | | | AIO247 |
| - | TempSensor[2] | | | | | B14 | | | | | | | | |

(2) Internal connection only; does not come to a device pin.
(3) PGA functionality not available on 64-pin and 56-pin packages.

**Table 12-2. Analog Signal Descriptions**

| SIGNAL NAME | DESCRIPTION |
|---|---|
| AIOx | Digital input on ADC pin |
| Ax | ADC A Input |
| Bx | ADC B Input |
| Cx | ADC C Input |
| CMPx_DACH | Comparator subsystem high DAC output |
| CMPx_DACL | Comparator subsystem low DAC output |
| CMPx_HNy | Comparator subsystem high comparator negative input |
| CMPx_HPy | Comparator subsystem high comparator positive input |
| CMPx_LNy | Comparator subsystem low comparator negative input |
| CMPx_LPy | Comparator subsystem low comparator positive input |
| DACx_OUT | Buffered DAC Output |
| PGAx_GND | PGA Ground |
| PGAx_IN | PGA Input |
| PGAx_OF | PGA Output for filter |
| PGAx_OUT | PGA Output to internal ADC |
| TempSensor | Internal temperature sensor |
| VDAC | Optional external reference voltage for on-chip DACs. There is a 100-pF capacitor to VSSA on this pin whether used for ADC input or DAC reference which cannot be disabled. If this pin is being used as a reference for the on-chip DACs, place at least a 1-uF capacitor on this pin. |

**Table 12-3. Reference Summary**

| Module | Reference Option | Configured Where? | Register | Notes |
|---|---|---|---|---|
| ADC | Internal or External | Analog System | AnalogSubsysRegs. ANAREFCTL.bit.ANAREFxSEL | Both options require use of the VREFHI pin |
| | 3.3V or 2.5V Internal Reference Range | Analog System | AnalogSubsysRegs. ANAREFCTL.bit.ANAREFx2P5SEL | Only applicable when using internal reference mode. |
| Buffered DAC | VREFHI or VDAC | DAC Module | DacxRegs. DACCTL.bit.DACREFSEL | |
| | Internal or External | Analog System | (follows ADC configuration) | Only applicable when using VREFHI |
| | 3.3V or 2.5V Internal Reference Range | Analog System | (follows ADC configuration) | Only applicable when using VREFHI with internal reference |
| | | DAC Module | DacxRegs. DACCTL.bit.MODE | Gain = 2 should be set when using 3.3V internal reference mode. |
| 3.3V or 2.5V Internal Reference Range | VDDA or VDAC | CMPSS Module | CmpssxRegs. COMPDACCTL.bit.SELREF | |

## 12.2 Registers

### 12.2.1 Analog Subsystem Base Addresses

**Table 12-4. Analog Subsystem Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| AnalogSubsysRegs | ANALOG_SUBSYS_REGS | 0x0005_D700 | 0x0005_D7FF |

### 12.2.1.1 ANALOG_SUBSYS_REGS Registers

Table 12-5 lists the memory-mapped registers for the ANALOG_SUBSYS_REGS. All register offset addresses not listed in Table 12-5 should be considered as reserved locations and the register contents should not be modified.

**Table 12-5. ANALOG_SUBSYS_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 1Eh | ANAREFPP | ADC Analog Reference Peripheral Properties register | EALLOW | Go |
| 60h | TSNSCTL | Temperature Sensor Control Register | EALLOW | Go |
| 68h | ANAREFCTL | Analog Reference Control Register | EALLOW | Go |
| 70h | VMONCTL | Voltage Monitor Control Register | EALLOW | Go |
| 78h | DCDCCTL | DC-DC control register. | EALLOW | Go |
| 7Ah | DCDCSTS | DC-DC status register. | | Go |
| 82h | CMPHPMXSEL | Bits to select one of the many sources on CopmHP inputs. Refer to Pimux diagram for details. | EALLOW | Go |
| 84h | CMPLPMXSEL | Bits to select one of the many sources on CopmLP inputs. Refer to Pimux diagram for details. | EALLOW | Go |
| 86h | CMPHNMXSEL | Bits to select one of the many sources on CopmHN inputs. Refer to Pimux diagram for details. | EALLOW | Go |
| 87h | CMPLNMXSEL | Bits to select one of the many sources on CopmLN inputs. Refer to Pimux diagram for details. | EALLOW | Go |
| 8Eh | LOCK | Lock Register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 12-6 shows the codes that are used for access types in this section.

**Table 12-6. ANALOG_SUBSYS_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| W=1 | W | Write |
| WOnce | W | Write |
| WSOnce | SOnce<br>W | Set once<br>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. |

**Table 12-6. ANALOG_SUBSYS_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. |

#### 12.2.1.1.1 ANAREFPP Register (Offset = 1Eh) [reset = 0h]

ANAREFPP is shown in Figure 12-5 and described in Table 12-7.

Return to Summary Table.

ADC Analog Reference Peripheral Properties register

**Figure 12-5. ANAREFPP Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | ANAREFCDIS | ANAREFBDIS |
| R-0h | | | | | | R/WOnce-0h | R/WOnce-0h |

**Table 12-7. ANAREFPP Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-2 | RESERVED | R | 0h | Reserved |
| 1 | ANAREFCDIS | R/WOnce | 0h | ANAREFC Disable. This bit field determines, whether ANAREFC is disabled or enabled.<br>0 ANAREFC is enabled.<br>1 ANAREFC is disabled.<br>Note: This bit should be programmed to 1 in parts where VREFHIA and VREFHIB are double bonded.<br>Reset type: XRSn |
| 0 | ANAREFBDIS | R/WOnce | 0h | ANAREFB Disable. This bit field determines, whether ANAREFB is disabled or enabled.<br>0 ANAREFB is enabled.<br>1 ANAREFB is disabled.<br>Note: This bit should be programmed to 1 in parts where VREFHIA, VREFHIB and VREFHIC are triple bonded.<br>Reset type: XRSn |

#### 12.2.1.1.2 TSNSCTL Register (Offset = 60h) [reset = 0h]

TSNSCTL is shown in Figure 12-6 and described in Table 12-8.

Return to Summary Table.

Temperature Sensor Control Register

**Figure 12-6. TSNSCTL Register**

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

**Table 12-8. TSNSCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-1 | RESERVED | R | 0h | Reserved |
| 0 | ENABLE | R/W | 0h | Temperature Sensor Enable. This bit enables the temperature sensor output to the ADC.<br>0 Disabled<br>1 Enabled<br>Reset type: SYSRSn |

#### 12.2.1.1.3 ANAREFCTL Register (Offset = 68h) [reset = Fh]

ANAREFCTL is shown in Figure 12-7 and described in Table 12-9.

Return to Summary Table.

Analog Reference Control Register

**Figure 12-7. ANAREFCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | ANAREFC2P5SEL | ANAREFB2P5SEL | ANAREFA2P5SEL |
| R-0h | | | | | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | ANAREFCSEL | ANAREFBSEL | ANAREFASEL |
| R-1h | | | | | R/W-1h | R/W-1h | R/W-1h |

**Table 12-9. ANAREFCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-11 | RESERVED | R | 0h | Reserved |
| 10 | ANAREFC2P5SEL | R/W | 0h | Analog referenc C 2.5V source select. In internal reference mode, this bit selects which voltage the internal reference buffer drives onto the VREFHI pin. The buffer can drive either 1.65V onto the pin, resulting in a reference range of 0 to 3.3V, or the buffer can drive 2.5V onto the pin, resulting in a reference range of 0 to 2.5V. If switching between these two modes, the user must allow adequate time for the external capacitor to charge to the new voltage before using the ADC or buffered DAC. If multiple VREFHI pins are ganged together (for lower pin-count packages), then the reference voltage select for the ganged pins should always be configured to the same setting.<br>0 Internal 1.65V reference mode (3.3V reference range)<br>1 Internal 2.5V reference mode (2.5V reference range)<br>Reset type: XRSn |
| 9 | ANAREFB2P5SEL | R/W | 0h | Analog referenc B 2.5V source select. In internal reference mode, this bit selects which voltage the internal reference buffer drives onto the VREFHI pin. The buffer can drive either 1.65V onto the pin, resulting in a reference range of 0 to 3.3V, or the buffer can drive 2.5V onto the pin, resulting in a reference range of 0 to 2.5V. If switching between these two modes, the user must allow adequate time for the external capacitor to charge to the new voltage before using the ADC or buffered DAC. If multiple VREFHI pins are ganged together (for lower pin-count packages), then the reference voltage select for the ganged pins should always be configured to the same setting.<br>0 Internal 1.65V reference mode (3.3V reference range)<br>1 Internal 2.5V reference mode (2.5V reference range)<br>Reset type: XRSn |

**Table 12-9. ANAREFCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 8 | ANAREFA2P5SEL | R/W | 0h | Analog referenc A 2.5V source select. In internal reference mode, this bit selects which voltage the internal reference buffer drives onto the VREFHI pin. The buffer can drive either 1.65V onto the pin, resulting in a reference range of 0 to 3.3V, or the buffer can drive 2.5V onto the pin, resulting in a reference range of 0 to 2.5V. If switching between these two modes, the user must allow adequate time for the external capacitor to charge to the new voltage before using the ADC or buffered DAC. If multiple VREFHI pins are ganged together (for lower pin-count packages), then the reference voltage select for the ganged pins should always be configured to the same setting.<br>0 Internal 1.65V reference mode (3.3V reference range)<br>1 Internal 2.5V reference mode (2.5V reference range)<br>Reset type: XRSn |
| 7-3 | RESERVED | R | 1h | Reserved |
| 2 | ANAREFCSEL | R/W | 1h | Analog reference C mode select. This bit selects whether the VREFHIC pin uses internal reference mode (the device drives a voltage onto the VREFHI pin) or external reference mode (the system is expected to drive a voltage into the VREFHI pin). If multiple VREFHI pins are ganged together (for lower pin-count packages), then the mode select for the ganged pins should always be configured to the same setting<br>0 Internal reference mode<br>1 External reference mode<br>Reset type: XRSn |
| 1 | ANAREFBSEL | R/W | 1h | Analog reference B mode select. This bit selects whether the VREFHIB pin uses internal reference mode (the device drives a voltage onto the VREFHI pin) or external reference mode (the system is expected to drive a voltage into the VREFHI pin). If multiple VREFHI pins are ganged together (for lower pin-count packages), then the mode select for the ganged pins should always be configured to the same setting<br>0 Internal reference mode<br>1 External reference mode<br>Reset type: XRSn |
| 0 | ANAREFASEL | R/W | 1h | Analog reference A mode select. This bit selects whether the VREFHIA pin uses internal reference mode (the device drives a voltage onto the VREFHI pin) or external reference mode (the system is expected to drive a voltage into the VREFHI pin). If multiple VREFHI pins are ganged together (for lower pin-count packages), then the mode select for the ganged pins should always be configured to the same setting<br>0 Internal reference mode<br>1 External reference mode<br>Reset type: XRSn |

#### 12.2.1.1.4 VMONCTL Register (Offset = 70h) [reset = 0h]

VMONCTL is shown in Figure 12-8 and described in Table 12-10.

Return to Summary Table.

Voltage Monitor Control Register

**Figure 12-8. VMONCTL Register**

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

**Table 12-10. VMONCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-9 | RESERVED | R | 0h | Reserved |
| 8 | BORLVMONDIS | R/W | 0h | BORL disable on VDDIO.<br>0 BORL is enabled on VDDIO, i.e BOR circuit will be triggered if VDDIO goes lower than the lower BOR threshold of VDDIO.<br>1 BORL is disabled on VDDIO, i.e BOR circuit will not be triggered if VDDIO goes lower than the lower BOR threshold of VDDIO.<br>Reset type: SYSRSn |
| 7-2 | RESERVED | R | 0h | Reserved |
| 1 | RESERVED | R | 0h | Reserved |
| 0 | RESERVED | R | 0h | Reserved |

#### 12.2.1.1.5 DCDCCTL Register (Offset = 78h) [reset = 80000000h]

DCDCCTL is shown in Figure 12-9 and described in Table 12-11.

Return to Summary Table.

DC-DC control register.

**Figure 12-9. DCDCCTL Register**

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

**Table 12-11. DCDCCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | RESERVED | R | 0h | Reserved |
| 30-1 | RESERVED | R | 0h | Reserved |
| 0 | DCDCEN | R/W | 0h | Enable DC-DC.<br>0 : Disables DC-DC and the device would work of internal VREG.<br>1 : Enables DC-DC.<br>Reset type: XRSn |

#### 12.2.1.1.6 DCDCSTS Register (Offset = 7Ah) [reset = 0h]

DCDCSTS is shown in Figure 12-10 and described in Table 12-12.

Return to Summary Table.

DC-DC status register.

**Figure 12-10. DCDCSTS Register**

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

**Table 12-12. DCDCSTS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1 | SWSEQDONE | R | 0h | DC-DC switch sequence done.<br>0 : Indicates that the sequence to switch to DC-DC is not complete.<br>1 : Indicates that the sequence to switch to DC-DC is complete.<br>When DCDCCTL.DCDCEN is set, PMM does the necessary sequencing to switch to DC-DC, and at the end of the sequence this bit will be set. However the power source will be switched to DC-DC only if inductor functionality check passes, else the device will continue to work of VREG.<br>Reset type: XRSn |
| 0 | INDDETECT | R/W=1 | 0h | Inductor Detected Status.<br>1 : Indicates that the external inductor connected to DC-DC is functional.<br>0 : Indicates that the external inductor connected to DC-DC is faulty.<br>When DCDCCTL.DCDCEN is set, PMM checks for proper functioning of the external inductor. If the check shows that inductor is functional then this bit will be set.<br>This status of this bit should be checked after DCDCSTS.SWSEQDONE is set.<br>Reset type: XRSn |

#### 12.2.1.1.7 CMPHPMXSEL Register (Offset = 82h) [reset = 0h]

CMPHPMXSEL is shown in Figure 12-11 and described in Table 12-13.

Return to Summary Table.

Bits to select one of the many sources on CopmHP inputs. Refer to Pimux diagram for details.

**Figure 12-11. CMPHPMXSEL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | CMP7HPMXSEL | | | CMP6HPMXSEL | | |
| R-0h | | R/W-0h | | | R/W-0h | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | CMP5HPMXSEL | | | CMP4HPMXSEL | | | CMP3HPMXSEL |
| R-0h | R/W-0h | | | R/W-0h | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CMP3HPMXSEL | | CMP2HPMXSEL | | | CMP1HPMXSEL | | |
| R/W-0h | | R/W-0h | | | R/W-0h | | |

**Table 12-13. CMPHPMXSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-22 | RESERVED | R | 0h | Reserved |
| 21-19 | CMP7HPMXSEL | R/W | 0h | CMP-4HPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Note: Only values 0 to 4 are valid, rest are reserved<br>Reset type: XRSn |
| 18-16 | CMP6HPMXSEL | R/W | 0h | CMP-5HPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Note: Only values 0 to 4 are valid, rest are reserved<br>Reset type: XRSn |
| 15 | RESERVED | R | 0h | Reserved |
| 14-12 | CMP5HPMXSEL | R/W | 0h | CMP5HPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Note: Only values 0 to 4 are valid, rest are reserved<br>Reset type: XRSn |
| 11-9 | CMP4HPMXSEL | R/W | 0h | CMP4HPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Note: Only values 0 to 4 are valid, rest are reserved<br>Reset type: XRSn |
| 8-6 | CMP3HPMXSEL | R/W | 0h | CMP3HPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Note: Only values 0 to 4 are valid, rest are reserved<br>Reset type: XRSn |

**Table 12-13. CMPHPMXSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5-3 | CMP2HPMXSEL | R/W | 0h | CMP2HPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Note: Only values 0 to 4 are valid, rest are reserved<br>Reset type: XRSn |
| 2-0 | CMP1HPMXSEL | R/W | 0h | CMP1HPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Note: Only values 0 to 4 are valid, rest are reserved<br>Reset type: XRSn |

#### 12.2.1.1.8 CMPLPMXSEL Register (Offset = 84h) [reset = 0h]

CMPLPMXSEL is shown in Figure 12-12 and described in Table 12-14.

Return to Summary Table.

Bits to select one of the many sources on CopmLP inputs. Refer to Pimux diagram for details.

**Figure 12-12. CMPLPMXSEL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | CMP7LPMXSEL | | | CMP6LPMXSEL | | |
| R-0h | | R/W-0h | | | R/W-0h | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | CMP5LPMXSEL | | | CMP4LPMXSEL | | | CMP3LPMXSEL |
| R-0h | R/W-0h | | | R/W-0h | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CMP3LPMXSEL | | CMP2LPMXSEL | | | CMP1LPMXSEL | | |
| R/W-0h | | R/W-0h | | | R/W-0h | | |

**Table 12-14. CMPLPMXSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-22 | RESERVED | R | 0h | Reserved |
| 21-19 | CMP7LPMXSEL | R/W | 0h | CMP7LPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Note: Only values 0 to 4 are valid, rest are reserved<br>Reset type: XRSn |
| 18-16 | CMP6LPMXSEL | R/W | 0h | CMP6LPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Note: Only values 0 to 4 are valid, rest are reserved<br>Reset type: XRSn |
| 15 | RESERVED | R | 0h | Reserved |
| 14-12 | CMP5LPMXSEL | R/W | 0h | CMP5LPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Note: Only values 0 to 4 are valid, rest are reserved<br>Reset type: XRSn |
| 11-9 | CMP4LPMXSEL | R/W | 0h | CMP4LPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Note: Only values 0 to 4 are valid, rest are reserved<br>Reset type: XRSn |
| 8-6 | CMP3LPMXSEL | R/W | 0h | CMP3LPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Note: Only values 0 to 4 are valid, rest are reserved<br>Reset type: XRSn |

**Table 12-14. CMPLPMXSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5-3 | CMP2LPMXSEL | R/W | 0h | CMP2LPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Note: Only values 0 to 4 are valid, rest are reserved<br>Reset type: XRSn |
| 2-0 | CMP1LPMXSEL | R/W | 0h | CMP1LPMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Note: Only values 0 to 4 are valid, rest are reserved<br>Reset type: XRSn |

#### 12.2.1.1.9 CMPHNMXSEL Register (Offset = 86h) [reset = 0h]

CMPHNMXSEL is shown in Figure 12-13 and described in Table 12-15.

Return to Summary Table.

Bits to select one of the many sources on CopmHN inputs. Refer to Pimux diagram for details.

**Figure 12-13. CMPHNMXSEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | CMP7HNMXSEL | CMP6HNMXSEL | CMP5HNMXSEL | CMP4HNMXSEL | CMP3HNMXSEL | CMP2HNMXSEL | CMP1HNMXSEL |
| R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 12-15. CMPHNMXSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-7 | RESERVED | R | 0h | Reserved |
| 6 | CMP7HNMXSEL | R/W | 0h | CMP7HNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Reset type: XRSn |
| 5 | CMP6HNMXSEL | R/W | 0h | CMP6HNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Reset type: XRSn |
| 4 | CMP5HNMXSEL | R/W | 0h | CMP5HNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Reset type: XRSn |
| 3 | CMP4HNMXSEL | R/W | 0h | CMP4HNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Reset type: XRSn |
| 2 | CMP3HNMXSEL | R/W | 0h | CMP3HNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Reset type: XRSn |
| 1 | CMP2HNMXSEL | R/W | 0h | CMP2HNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Reset type: XRSn |
| 0 | CMP1HNMXSEL | R/W | 0h | CMP1HNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Reset type: XRSn |

#### 12.2.1.1.10 CMPLNMXSEL Register (Offset = 87h) [reset = 0h]

CMPLNMXSEL is shown in Figure 12-14 and described in Table 12-16.

Return to Summary Table.

Bits to select one of the many sources on CopmLN inputs. Refer to Pimux diagram for details.

**Figure 12-14. CMPLNMXSEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | CMP7LNMXSEL | CMP6LNMXSEL | CMP5LNMXSEL | CMP4LNMXSEL | CMP3LNMXSEL | CMP2LNMXSEL | CMP1LNMXSEL |
| R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 12-16. CMPLNMXSEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-7 | RESERVED | R | 0h | Reserved |
| 6 | CMP7LNMXSEL | R/W | 0h | CMP7LNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Reset type: XRSn |
| 5 | CMP6LNMXSEL | R/W | 0h | CMP6LNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Reset type: XRSn |
| 4 | CMP5LNMXSEL | R/W | 0h | CMP5LNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Reset type: XRSn |
| 3 | CMP4LNMXSEL | R/W | 0h | CMP4LNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Reset type: XRSn |
| 2 | CMP3LNMXSEL | R/W | 0h | CMP3LNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Reset type: XRSn |
| 1 | CMP2LNMXSEL | R/W | 0h | CMP2LNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Reset type: XRSn |
| 0 | CMP1LNMXSEL | R/W | 0h | CMP1LNMXSEL bits, Please refer to the analog group connections diagram for the general structure of the CMPSS input mux. Please refer to the Analog Pins and Internal Connections table for specific connections for the CMPSS input mux.<br>Reset type: XRSn |

#### 12.2.1.1.11 LOCK Register (Offset = 8Eh) [reset = 0h]

LOCK is shown in Figure 12-15 and described in Table 12-17.

Return to Summary Table.

Lock Register

**Figure 12-15. LOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | VREGCTL | CMPLNMXSEL |
| R-0h | | | | | | R/WSOnce-0h | R/WSOnce-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CMPHNMXSEL | CMPLPMXSEL | CMPHPMXSEL | ADCINMXSEL | DCDCCTL | VMONCTL | ANAREFCTL | TSNSCTL |
| 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 12-17. LOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-10 | RESERVED | R | 0h | Reserved |
| 9 | VREGCTL | R/WSOnce | 0h | VREGCTL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset.<br>Reset type: SYSRSn |
| 8 | CMPLNMXSEL | R/WSOnce | 0h | CMPLNMXSEL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset.<br>Reset type: SYSRSn |
| 7 | CMPHNMXSEL | R/WSOnce | 0h | CMPHNMXSEL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset.<br>Reset type: SYSRSn |
| 6 | CMPLPMXSEL | R/WSOnce | 0h | CMPLPMXSEL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset.<br>Reset type: SYSRSn |
| 5 | CMPHPMXSEL | R/WSOnce | 0h | CMPHPMXSEL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset.<br>Reset type: SYSRSn |
| 4 | ADCINMXSEL | R/WSOnce | 0h | ADCINMXSEL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset.<br>Reset type: SYSRSn |
| 3 | DCDCCTL | R/WSOnce | 0h | DCDCCTL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset.<br>Reset type: SYSRSn |
| 2 | VMONCTL | R/WSOnce | 0h | VMONCTL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset.<br>Reset type: SYSRSn |

**Table 12-17. LOCK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | ANAREFCTL | R/WSOnce | 0h | ANAREFCTL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset.<br>Reset type: SYSRSn |
| 0 | TSNSCTL | R/WSOnce | 0h | TSNSCTL Register Lock. Setting this bit will disable any future write to the respective register. This bit can only be cleared by a reset.<br>Reset type: SYSRSn |

# Analog-to-Digital Converter (ADC)

The analog-to-digital converter module described in this chapter is a Type 5 ADC. See the *TMS320C28xx, 28xxx DSP Peripheral Reference Guide* (SPRU566) for a list of all devices with modules of the same type, to determine the differences between the types, and for a list of device-specific differences within a type.

**Topic** **Page**

## 13.1 Analog-to-Digital Converter (ADC)

The ADC module described here is a successive approximation (SAR) style ADC with resolution of 12 bits. This chapter refers to the analog circuits of the converter as the “core,” and includes the channel select MUX, the sample-and-hold (S/H) circuit, the successive approximation circuits, voltage reference circuits, and other analog support circuits. The digital circuits of the converter are referred to as the “wrapper” and includes logic for programmable conversions, result registers, interfaces to analog circuits, interfaces to the peripheral buses, post-processing circuits, and interfaces to other on-chip modules.

### *13.1.1 Features*

Each ADC module consists of a single sample-and-hold (s/h) circuit. The ADC module is designed to be duplicated multiple times on the same chip, allowing simultaneous sampling or independent operation of multiple ADCs. The ADC wrapper is start-of-conversion (SOC) based (see Section 13.1.4).

Each ADC has the following features:

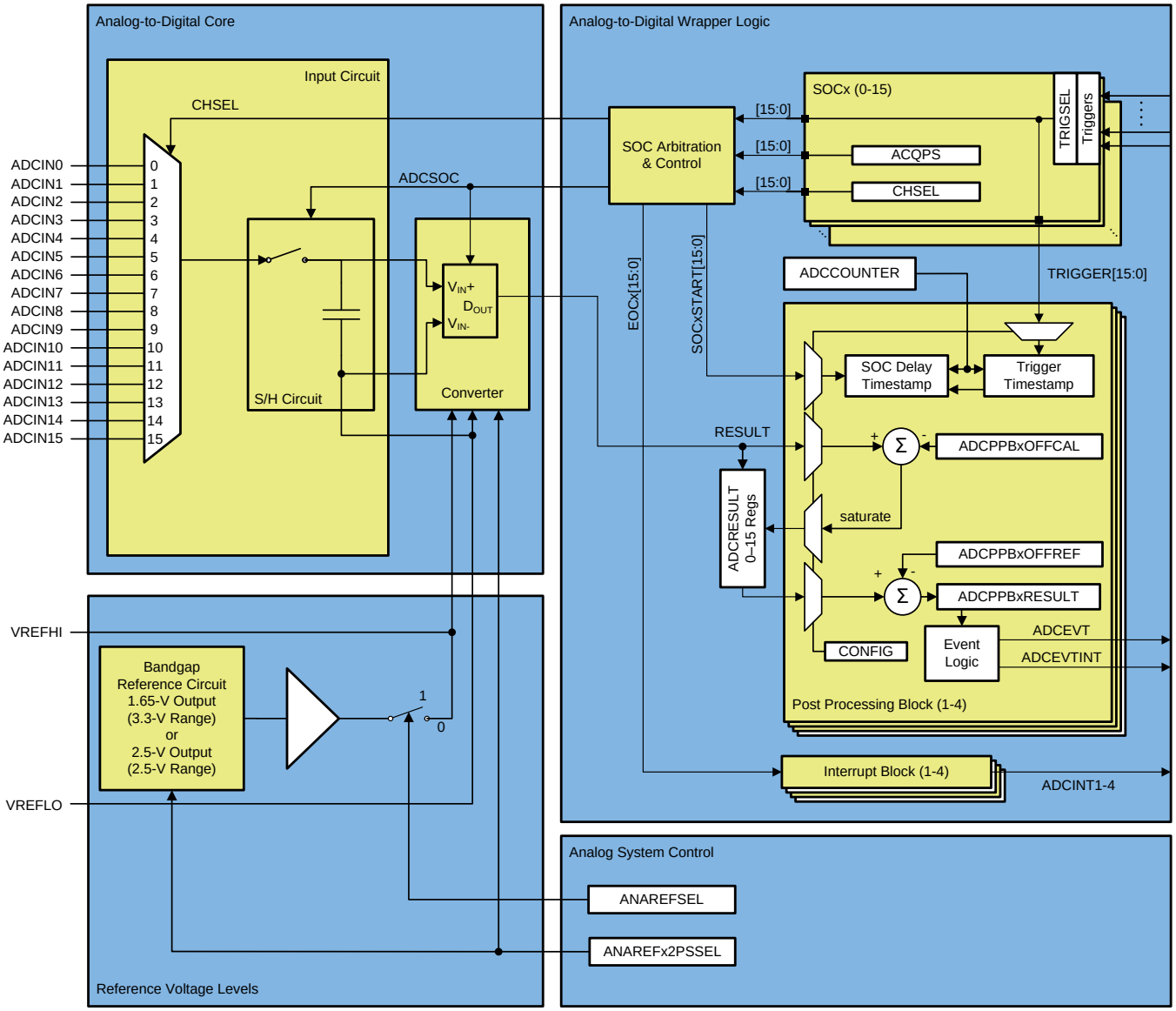
- Resolution of 12 bits
- Single-ended signaling
- Ratiometric external reference set by VREFHI/VREFLO
- Selectable internal reference of 2.5v or 3.3v
- Input multiplexer with up to 16 channels
- 16 configurable SOCs
- 16 individually addressable result registers
- Multiple trigger sources
  - S/W - software immediate start
  - All ePWMs - ADCSOC A or B
  - GPIO XINT2
  - CPU Timers 0/1/2
  - ADCINT1/2
- Four flexible PIE interrupts
- Burst mode
- Four post-processing blocks, each with:
  - Saturating offset calibration
  - Error from setpoint calculation
  - High, low, and zero-crossing compare, with interrupt and ePWM trip capability
  - Trigger-to-sample delay capture

**NOTE:** Not every channel may be pinned out from all ADCs. Check the datasheet for your device to determine which channels are available.

### 13.1.2 ADC Block Diagram

The block diagram for the ADC core and ADC wrapper are presented in Figure 13-1.

**Figure 13-1. ADC Module Block Diagram**



### 13.1.3 ADC Configurability

Some ADC configurations are individually controlled by the SOCs, while others are globally controlled per ADC module. Table 13-1 summarizes the basic ADC options and their level of configurability. The subsequent sections discuss these configurations.

**Table 13-1. ADC Options and Configuration Levels**

| Options | Configurability |
|---|---|
| Clock | Per module(1) |
| Resolution | Not configurable (12-bit resolution only) |
| Signal mode | Not configurable (single-ended signal mode only) |
| Reference voltage source | Per module(2) |
| Trigger source | Per SOC(1) |
| Converted channel | Per SOC |
| Acquisition window duration | Per SOC(1) |
| EOC location | Per module |
| Burst Mode | Per module(1) |

(1) Writing these values differently to different ADC modules could cause the ADCs to operate asynchronously. See Section 13.3.1 for guidance on when the ADCs are operating synchronously or asynchronously.

(2) Lower pin count packages may share one VREFHI pin among multiple ADCs. In this case, the ADCs that share a reference pin must have their reference modes configured identically

#### 13.1.3.1 Clock Configuration

The base ADC clock is provided directly by the system clock (SYSCLK). This clock is used to generate the ADC acquisition window. The register ADCCTL2 has a PRESCALE field which determines the ADCCLK. The ADCCLK is used to clock the converter.

The core requires approximately 10.5 ADCCLK cycles to process a voltage into a conversion result in 12-bit mode. Refer to Section 13.3.2 to determine the necessary duration of the acquisition window.

**NOTE:** To determine an appropriate value for ADCCTL2.PRESCALE, consult the datasheet of your device to determine the maximum SYSCLK and ADCCLK frequency.

#### 13.1.3.2 Voltage Reference

##### 13.1.3.2.1 External Reference Mode

Each ADC has a VREFHI input and a VREFLO input, which is used as a ratiometric reference.

See Section 13.3.4 for information on how to supply the reference voltage.

**NOTES:**

- On devices with no external VREFLO signals, VREFLO has been internally connected to the device analog ground, VSSA.
- Consult the datasheet for your device to determine the allowable voltage range for VREFHI and VREFLO.
- The external reference mode requires an external capacitor on the VREFHI pin. Consult the device datasheet for the specific value required.

##### 13.1.3.2.2 Internal Reference Mode

There are two software selectable internal reference mode voltages, 3.3V and 2.5V. Using internal reference mode requires an external capacitor on VREFHI pin. Consult device datasheet for the recommended capacitance value.

#### 13.1.3.2.3 Ganged References

On some packages, the voltage reference pins for multiple ADCs may be combined. In this case, it is necessary to configure the ganged references identically when selecting external vs. internal reference mode and for selecting an internal reference voltage range of 3.3V or 2.5V.

For example, if ADC B and ADC C reference pins are combined and the desired reference mode is 2.5V internal reference mode, the following reference configuration code should be run:

```
//ADCB VREFHI and ADCC VREFHI share a pin
//ADCB VREFLO and ADCC VREFLO share a pin
//Both references must be explicitly configured
//Both references must be configured identically
SetVREF(ADC_ADCB, ADC_INTERNAL, ADC_VREF2P5);
SetVREF(ADC_ADCC, ADC_INTERNAL, ADC_VREF2P5);
```

### 13.1.3.3 Signal Mode

This ADC only supports single-ended mode. In this mode, the input voltage to the converter is sampled through a single pin (ADCINx), referenced to VREFLO.

### 13.1.3.4 Expected Conversion Results

Based on a given analog input voltage, the ideal expected digital conversion is given by the tables below. Fractional values are truncated.

**Table 13-2. Analog to 12-bit Digital Formulas**

| Analog Input | Digital Result |
|---|---|
| when ADCINy ≤ VREFLO | ADCRESULTx = 0 |
| when VREFLO < ADCINy < VREFHI | $ADCRESULTx = 4096\left(\frac{ADCINy - VREFLO}{VREFHI - VREFLO}\right)$ |
| when ADCINy ≥ VREFHI | ADCRESULTx = 4095 |

### 13.1.3.5 Interpreting Conversion Results

Based on a given ADC conversion result, the ideal corresponding analog input is given by the below tables. This corresponds to the center of the possible range of analog voltages that could have produced this conversion result.

**Table 13-3. 12-Bit Digital-to-Analog Formulas**

| Digital Value | Analog Equivalent |
|---|---|
| when ADCRESULTy = 0 | ADCINx ≤ VREFLO |
| when 0 < ADCRESULTy < 4095 | $ADCINx = (VREFHI - VREFLO)\left(\frac{ADCRESULTy}{4096}\right) + VREFLO$ |
| when ADCRESULTy = 4095 | ADCINx ≥ VREFHI |

### 13.1.3.6 Shared Reference Pins

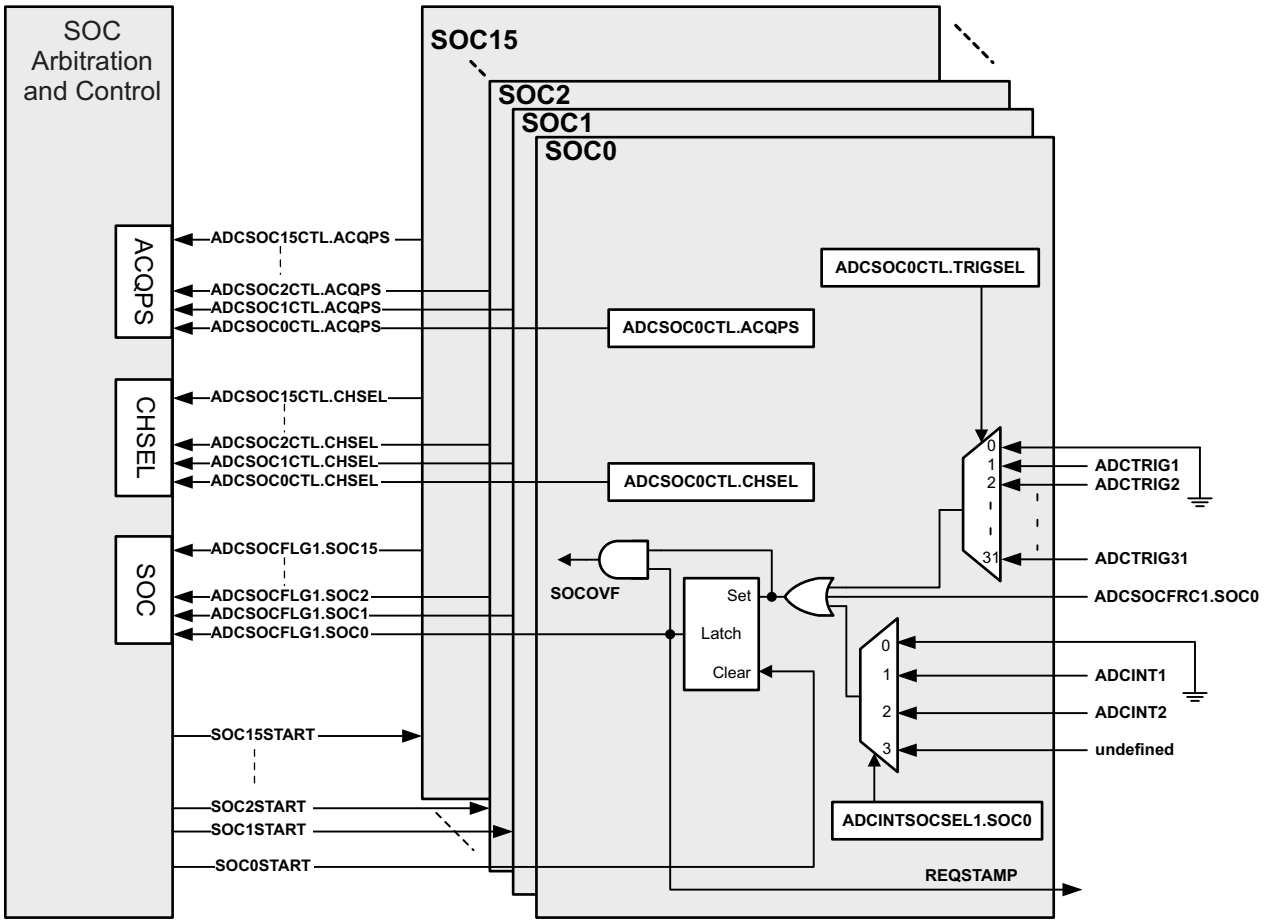
Lower pin count packages may share one VREFHI pin among multiple ADCs. In this case, the ADCs that share a reference pin must have their reference modes configured identically. If configured in internal reference mode, device hardware automatically ensures that multiple reference buffers do not drive contending voltages onto the same pin.

### 13.1.4 SOC Principle of Operation

The ADC triggering and conversion sequencing is accomplished through configurable start-of-conversions (SOCs). Each SOC is a configuration set defining the single conversion of a single channel. In that set there are three configurations: the trigger source that starts the conversion, the channel to convert, and the acquisition (sample) window duration. Upon receiving the trigger configured for a SOC, the wrapper will ensure that the specified channel is captured using the specified acquisition window duration.

Multiple SOCs can be configured for the same trigger, channel, and/or acquisition window as desired. Configuring multiple SOCs to use the same trigger will allow the trigger to generate a sequence of conversions. Configuring multiple SOCs to use the same trigger and channel will allow for oversampling.

**Figure 13-2. SOC Block Diagram**




### 13.1.4.1 SOC Configuration

Each SOC has its own configuration register, ADCSOCxCTL. Within this register, SOCx can be configured for trigger source, channel to convert, and acquisition (sample) window duration.

### 13.1.4.2 Trigger Operation

Each SOC can be configured to start on one of many input triggers. The primary trigger select for SOCx is in the ADCSOCxCTL.TRIGSEL register, which can select between:

- Disabled (software only)
- CPU Timers 0/1/2
- GPIO: Input X-Bar INPUT5
- ePWM1 to ePWM8, ADCSOCA or ADCSOCB

In addition, each SOC can also be triggered when the ADCINT1 flag or ADCINT2 flag is set. This is achieved by configuring the ADCINTSOCSEL1 register (for SOC0 to SOC7) or the ADCINTSOCSEL2 register (for SOC8 to SOC15). This is useful for creating continuous conversions.

### 13.1.4.3 ADC Acquisition (Sample and Hold) Window

External signal sources vary in their ability to drive an analog signal quickly and effectively. In order to achieve rated resolution, the signal source needs to charge the sampling capacitor in the ADC core to within 0.5LSBs of the signal voltage. The acquisition window is the amount of time the sampling capacitor is allowed to charge and is configurable for SOCx by the ADCSOCxCTL.ACQPS register.

ACQPS is a 9-bit register that can be set to a value between 0 and 511, resulting in an acquisition window duration of:

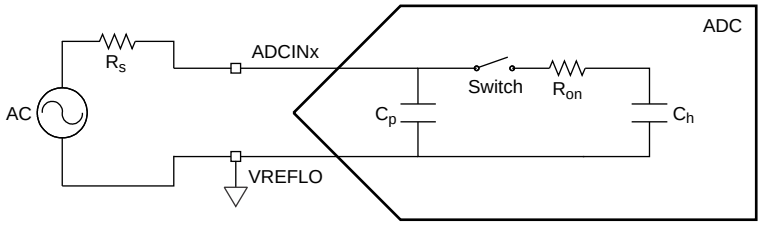
Acquisition window = (ACQPS + 1)·(System Clock (SYSCLK) cycle time)

- The acquisition window duration is based on the System Clock (SYSCLK), not the ADC clock (ADCCLK).
- The selected acquisition window duration must be at least as long as one ADCCLK cycle.
- The datasheet will specify a minimum acquisition window duration (in nanoseconds). The user is responsible for selecting an acquisition window duration that meets this requirement.

#### 13.1.4.4 Input Model

For single-ended operation, the ADC input characteristics for values in the single-ended input model (see Figure 13-3) can be found in the device data manual.

**Figure 13-3. Input Model**




These input models should be used along with actual signal source impedance to determine the acquisition window duration. See Section 13.3.2 for more information.

#### 13.1.4.5 Channel Selection

Each SOC can be configured to convert any of the ADC channels. This behavior is selected for SOCx by the ADCSOCxCTL.CHSEL register. This is summarized in Table 13-4.

**Table 13-4. Channel Selection of Input Pins**

| CHSEL | Input |
|---|---|
| 0 | ADCIN0 |
| 1 | ADCIN1 |
| 2 | ADCIN2 |
| 3 | ADCIN3 |
| 4 | ADCIN4 |
| 5 | ADCIN5 |
| 6 | ADCIN6 |
| 7 | ADCIN7 |
| 8 | ADCIN8 |
| 9 | ADCIN9 |
| 10 | ADCIN10 |
| 11 | ADCIN11 |
| 12 | ADCIN12 |
| 13 | ADCIN13 |
| 14 | ADCIN14 |
| 15 | ADCIN15 |

### 13.1.5 SOC Configuration Examples

The following sections provide some specific examples of how to configure the SOCs to produce some conversions.

#### 13.1.5.1 Single Conversion from ePWM Trigger

To configure ADCA to perform a single conversion on channel ADCIN1 when the ePWM timer reaches its period match, a few things are necessary. First, ePWM3 must be configured to generate an SOCA, SOCB, SOCC, or SOCD signal (in this statement, SOC refers to a signal in the ePWM module). See the *Enhanced Pulse Width Modulator Module (ePWM)* chapter on how to do this. Assume that SOCB was chosen.

SOC5 is chosen arbitrarily. Any of the 16 SOCs could be used.

Assuming a 100ns sample window is desired with a SYSCLK frequency of 100MHz, then the acquisition window duration should be 100ns/10ns = 10 cycles. The ACQPS field should therefore be set to 10 –1 = 9.

```
AdcaRegs.ADCSOC5CTL.bit.CHSEL = 1;      //SOC5 will convert ADCINA1
AdcaRegs.ADCSOC5CTL.bit.ACQPS = 9;      //SOC5 will use sample duration of 10 SYSCLK cycles
AdcaRegs.ADCSOC5CTL.bit.TRIGSEL = 10;   //SOC5 will begin conversion on ePWM3 SOCB
```

As configured, when ePWM3 matches its period and generates the SOCB signal, the ADC will begin sampling channel ADCINA1 (SOC5) immediately if the ADC is idle. If the ADC is busy, ADCINA1 will begin sampling when SOC5 gains priority (see Section 13.1.6).The ADC control logic will sample ADCINA1 with the specified acquisition window width of 100 ns. Immediately after the acquisition is complete, the ADC will begin converting the sampled voltage to a digital value. When the ADC conversion is complete, the results will be available in the ADCRESULT5 register (see Section 13.2 for exact sample, conversion, and result latch timings).

#### 13.1.5.2 Oversampled Conversion from ePWM Trigger

To configure the ADC to oversample ADCINA1 4 times, we use the same configurations as the previous example, but apply them to SOC5, SOC6, SOC7, and SOC8.

```
AdcaRegs.ADCSOC5CTL.bit.CHSEL = 1;        //SOC5 will convert ADCINA1
AdcaRegs.ADCSOC5CTL.bit.ACQPS = 11;       //SOC5 will use sample duration of 12 SYSCLK cycles
AdcaRegs.ADCSOC5CTL.bit.TRIGSEL = 10;     //SOC5 will begin conversion on ePWM3 SOCB
AdcaRegs.ADCSOC6CTL.bit.CHSEL = 1;        //SOC6 will convert ADCINA1
AdcaRegs.ADCSOC6CTL.bit.ACQPS = 11;       //SOC6 will use sample duration of 12 SYSCLK cycles
AdcaRegs.ADCSOC6CTL.bit.TRIGSEL = 10;     //SOC6 will begin conversion on ePWM3 SOCB
AdcaRegs.ADCSOC7CTL.bit.CHSEL = 1;        //SOC7 will convert ADCINA1
AdcaRegs.ADCSOC7CTL.bit.ACQPS = 11;       //SOC7 will use sample duration of 12 SYSCLK cycles
AdcaRegs.ADCSOC7CTL.bit.TRIGSEL = 10;     //SOC7 will begin conversion on ePWM3 SOCB
AdcaRegs.ADCSOC8CTL.bit.CHSEL = 1;        //SOC8 will convert ADCINA1
AdcaRegs.ADCSOC8CTL.bit.ACQPS = 11;       //SOC8 will use sample duration of 12 SYSCLK cycles
AdcaRegs.ADCSOC8CTL.bit.TRIGSEL = 10;     //SOC8 will begin conversion on ePWM3 SOCB
```

As configured, when ePWM3 matches its period and generates the SOCB signal, the ADC will begin sampling channel ADCINA1 (SOC5) immediately if the ADC is idle. If the ADC is busy, ADCINA1 will begin sampling when SOC5 gains priority (see ADC Conversion Priority). Once the conversion is complete for SOC5, SOC6 will begin converting ADCINA1 and the results for SOC5 will be placed in the ADCRESULT5 register. All four conversions will eventually be completed sequentially, with the results in ADCRESULT5, ADCRESULT6, ADCRESULT7, and ADCRESULT8 for SOC5, SOC6, SOC7, and SOC8, respectively.

**NOTE:** It is possible, but unlikely, that the ADC could begin converting SOC6, SOC7, or SOC8 before SOC5 depending on the position of the round-robin pointer when the ePWM trigger is received. See ADC Conversion Priority to understand how the next SOC to be converted is chosen.

### 13.1.5.3 Multiple Conversions from CPU Timer Trigger

This example will show how to sample multiple signals with different acquisition window requirements. CPU Timer 2 will be used to generate the trigger. To see how to configure the CPU timer, see the *System Control and Interrupts* chapter.

A good first step when designing a sampling scheme with many signals is to list out the signals and their required acquisition window. From this, calculate the necessary number of SYSCLK cycles for each signal, then the ACQPS register setting. This is shown in Table 13-5, where a SYCLK of 120MHz is assumed (8.333ns cycle time).

**Table 13-5. Example Requirements for Multiple Signal Sampling**

| Signal Name | Acquisition Window Requirement (ns) | Acquisition Window SYSCLK Cycles | ACQPS Register Value |
|---|---|---|---|
| Signal 1 | >200ns | 200ns/8.333ns = 24 | 24 – 1 = 23 |
| Signal 2 | >740ns | 740ns/8.333ns = 89 (round up) | 89 – 1 = 88 |
| Signal 3 | >183.33 ns | 183.333ns/8.333ns = 22 | 22 – 1 = 21 |
| Signal 4 | >485ns | 485ns/8.333ns = 59 (round up) | 59 – 1 = 58 |

Next decide which ADC pins to connect to each signal. This will be highly dependent on application board layout. Once the pins are selected, determining the value of CHSEL is straightforward (see Table 13-6).

**Table 13-6. Example Connections for Multiple Signal Sampling**

| Signal Name | ADC PIN | CHSEL Register Value |
|---|---|---|
| Signal 1 | ADCINA5 | 5 |
| Signal 2 | ADCINA0 | 0 |
| Signal 3 | ADCINA3 | 3 |
| Signal 4 | ADCINA2 | 2 |

With the information tabulated, it is easy to generate the SOC configurations:

```
AdcaRegs.ADCSOC0CTL.bit.CHSEL = 5;        //SOC0 will convert ADCINA5
AdcaRegs.ADCSOC0CTL.bit.ACQPS = 23;       //SOC0 will use sample duration of 24 SYSCLK cycles
AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 3;      //SOC0 will begin conversion on CPU Timer 2
AdcaRegs.ADCSOC1CTL.bit.CHSEL = 0;        //SOC1 will convert ADCINA0
AdcaRegs.ADCSOC1CTL.bit.ACQPS = 88;       //SOC1 will use sample duration of 89 SYSCLK cycles
AdcaRegs.ADCSOC1CTL.bit.TRIGSEL = 3;      //SOC1 will begin conversion on CPU Timer 2
AdcaRegs.ADCSOC2CTL.bit.CHSEL = 3;        //SOC2 will convert ADCINA3
AdcaRegs.ADCSOC2CTL.bit.ACQPS = 21;       //SOC2 will use sample duration of 22 SYSCLK cycles
AdcaRegs.ADCSOC2CTL.bit.TRIGSEL = 3;      //SOC2 will begin conversion on CPU Timer 2
AdcaRegs.ADCSOC3CTL.bit.CHSEL = 2;        //SOC3 will convert ADCINA2
AdcaRegs.ADCSOC3CTL.bit.ACQPS = 58;       //SOC3 will use sample duration of 59 SYSCLK cycles
AdcaRegs.ADCSOC3CTL.bit.TRIGSEL = 3;      //SOC3 will begin conversion on CPU Timer 2
```

As configured, when CPU Timer 2 generates an event, SOC0, SOC1, SOC2, and SOC3 will eventually be sampled and converted, in that order. The conversion results for ACINA5 (Signal 1) will be in ADCRESULT0. Similarly, The results for ADCINA0 (Signal 2), ADCINA3 (Signal 3), and ADCINA2 (Signal 4) will be in ADCRESULT1, ADCRESULT2, and ADCRESULT3, respectively.

**NOTE:** It is possible, but unlikely, that the ADC could begin converting SOC1, SOC2, or SOC3 before SOC0 depending on the position of the round-robin pointer when the CPU Timer trigger is received. See ADC Conversion Priority to understand how the next SOC to be converted is chosen.

#### 13.1.5.4 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 CPU Timer 2 to generate the trigger could be accomplished by the statement:

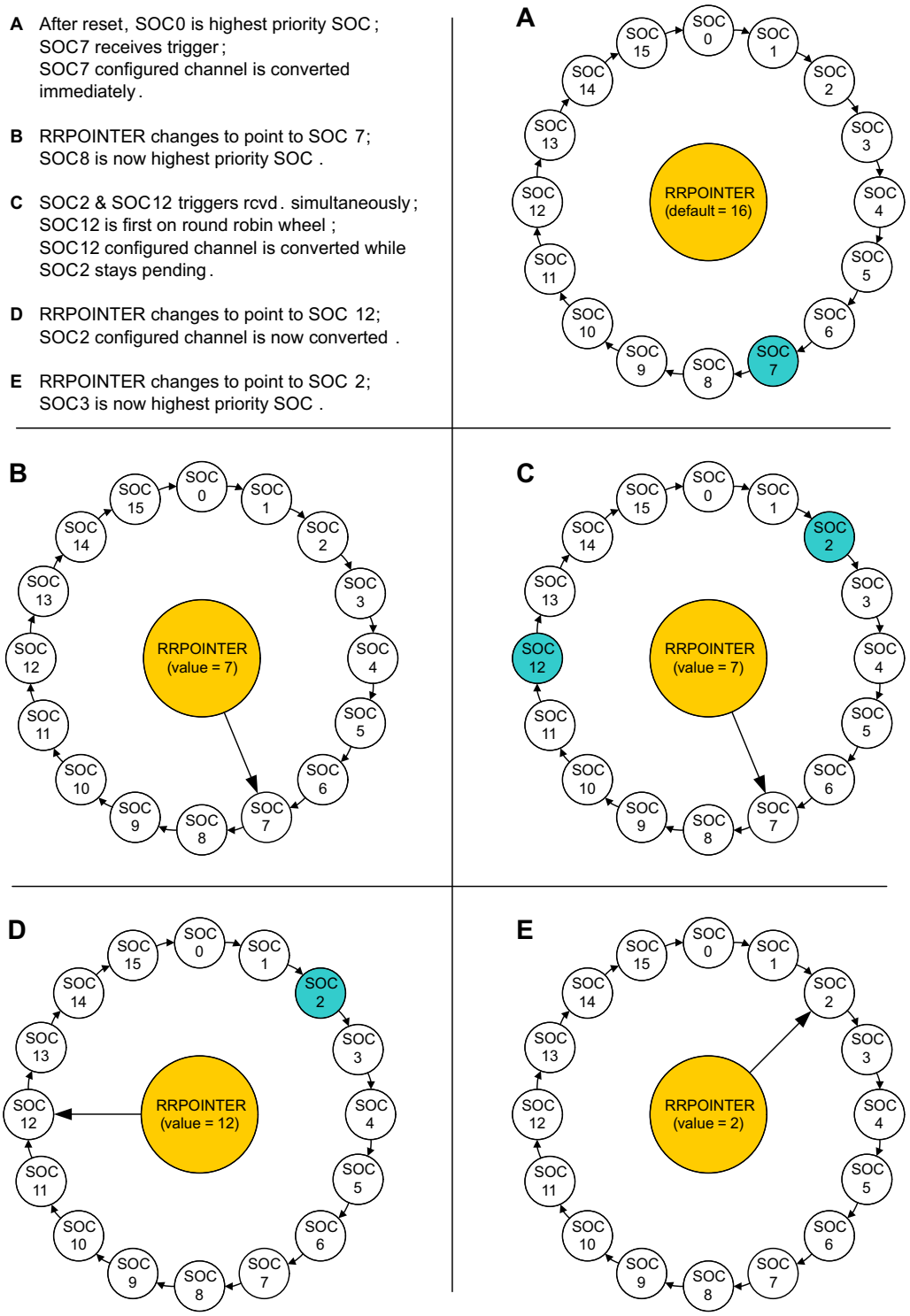
```
AdcaRegs.ADCSOCFRC1.all = 0x000F;        //set SOC flags for SOC0 to SOC3
```

### *13.1.6 ADC Conversion Priority*

When multiple SOC flags are set at the same time, one of two forms of priority determines the order in which they are converted. The default priority method is round robin. In this scheme, no SOC has an inherent higher priority than another. Priority depends on the round robin pointer (RRPOINTER). The RRPOINTER reflected in the ADCSOCPRIORITYCTL register points to the last SOC converted. The highest priority SOC is given to the next value greater than the RRPOINTER value, wrapping around back to SOC0 after SOC15. At reset the value is 16 since 0 indicates a conversion has already occurred. When RRPOINTER equals 16 the highest priority is given to SOC0. The RRPOINTER is reset by a device reset, when the ADCCTL1.RESET bit is set, or when the SOCPRICTL register is written.

An example of the round robin priority method is given in Figure 13-4 .

**Figure 13-4. Round Robin Priority Example**

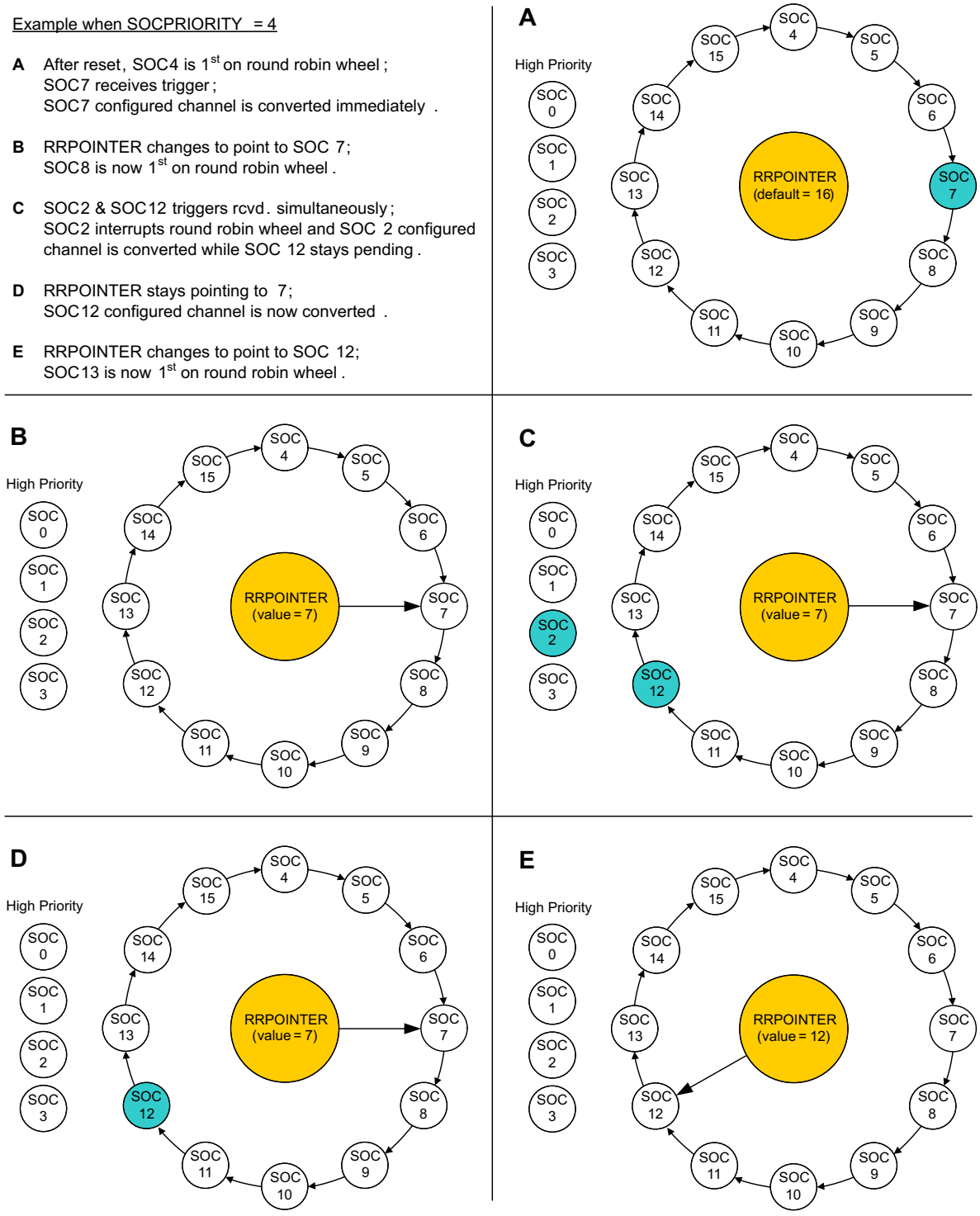



The SOCPRIORITY field in the ADCSOCPRIORITYCTL register can be used to assign high priority from a single to all of the SOCs. When configured as high priority, an SOC will interrupt the round robin wheel after any current conversion completes and insert itself in as the next conversion. After its conversion completes, the round robin wheel will continue where it was interrupted. If two high priority SOC's are triggered at the same time, the SOC with the lower number will take precedence.

High priority mode is assigned first to SOC0, then in increasing numerical order. The value written in the 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 13-5 .

**Figure 13-5. High Priority Example**



### 13.1.7 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 will not set all round-robin SOCs to be converted, but only (ADCBURSTCTL.BURSTSIZE + 1) SOCs. The first SOC to be set will be that with the highest priority based on the round-robin pointer, and subsequent SOCs will be 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.

#### 13.1.7.1 Burst Mode Example

Burst mode can be used to sample a different set of signals on every other trigger. In the following example, ADCIN7 and ADCIN5 are converted on the first trigger from CPU Timer 2 and every other trigger thereafter. ADCIN2 and ACIN3 are converted on the second trigger from CPU Timer 2 and every other trigger thereafter. All signals are converted with 20 SYSCLK cycle wide acquisition windows, but different durations could be configured for each SOC as desired.

```
AdcaRegs.BURSTCTL.BURSTEN = 1;            //Enable ADC burst mode
AdcaRegs.BURSTCTL.BURSTTRIG = 3;          //CPU Timer 2 will trigger 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 will convert ADCINA7
AdcaRegs.ADCSOC12CTL.bit.ACQPS = 19;      //SOC12 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC13CTL.bit.CHSEL = 5;       //SOC13 will convert ADCINA5
AdcaRegs.ADCSOC13CTL.bit.ACQPS = 19;      //SOC13 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC14CTL.bit.CHSEL = 2;       //SOC14 will convert ADCINA2
AdcaRegs.ADCSOC14CTL.bit.ACQPS = 19;      //SOC14 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC15CTL.bit.CHSEL = 3;       //SOC15 will convert ADCINA3
AdcaRegs.ADCSOC15CTL.bit.ACQPS = 19;      //SOC15 will use sample duration of 20 SYSCLK cycles
```

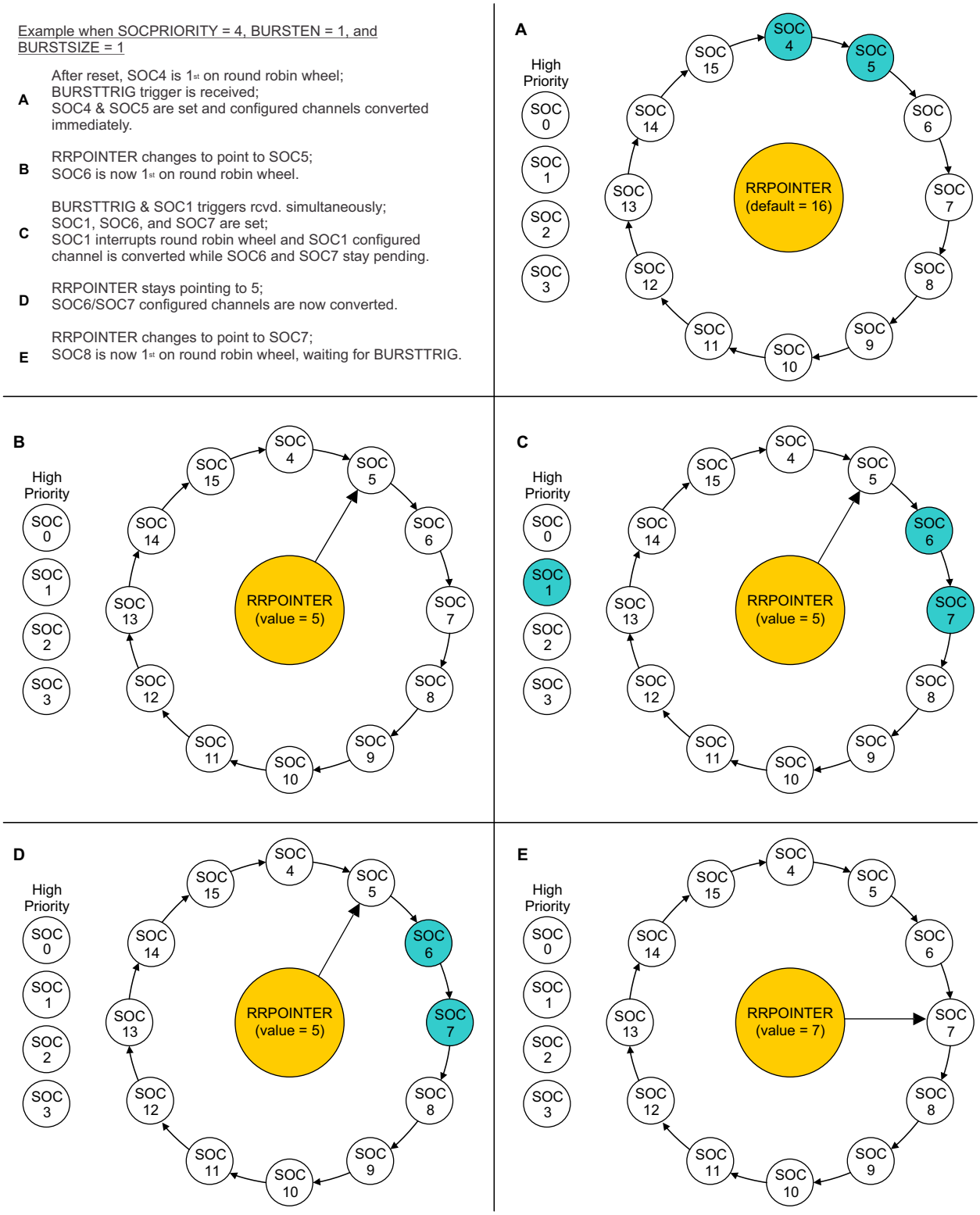
When the first CPU Timer 2 trigger is received, SOC12 and SOC13 will be converted immediately if the ADC is idle. If the ADC is busy, SOC12 and SOC13 will be converted once their SOCs gain priority. The results for SOC12 and SOC13 will be in ADCRESULT12 and ADCRESULT13, respectively. After SOC13 completes, the round robin pointer will give highest priority to SOC14. Because of this, when the next CPU Timer 2 trigger is received, SOC14 and SOC15 will be set as pending and eventually be converted. The results for SOC14 and SOC15 will be in ADCRESULT14 and ADCRESULT15, respectively. Subsequent triggers will continue to toggle between converting SOC12 and SOC13, and converting SOC14 and SOC15.

While the above example toggles between two sets of conversions, three or more different sets of conversions could be achieved using a similar approach.

#### 13.1.7.2 Burst Mode Priority Example

An example of priority resolution using burst mode and high-priority SOCs is presented in Figure 13-6.

**Figure 13-6. Burst Priority Example**



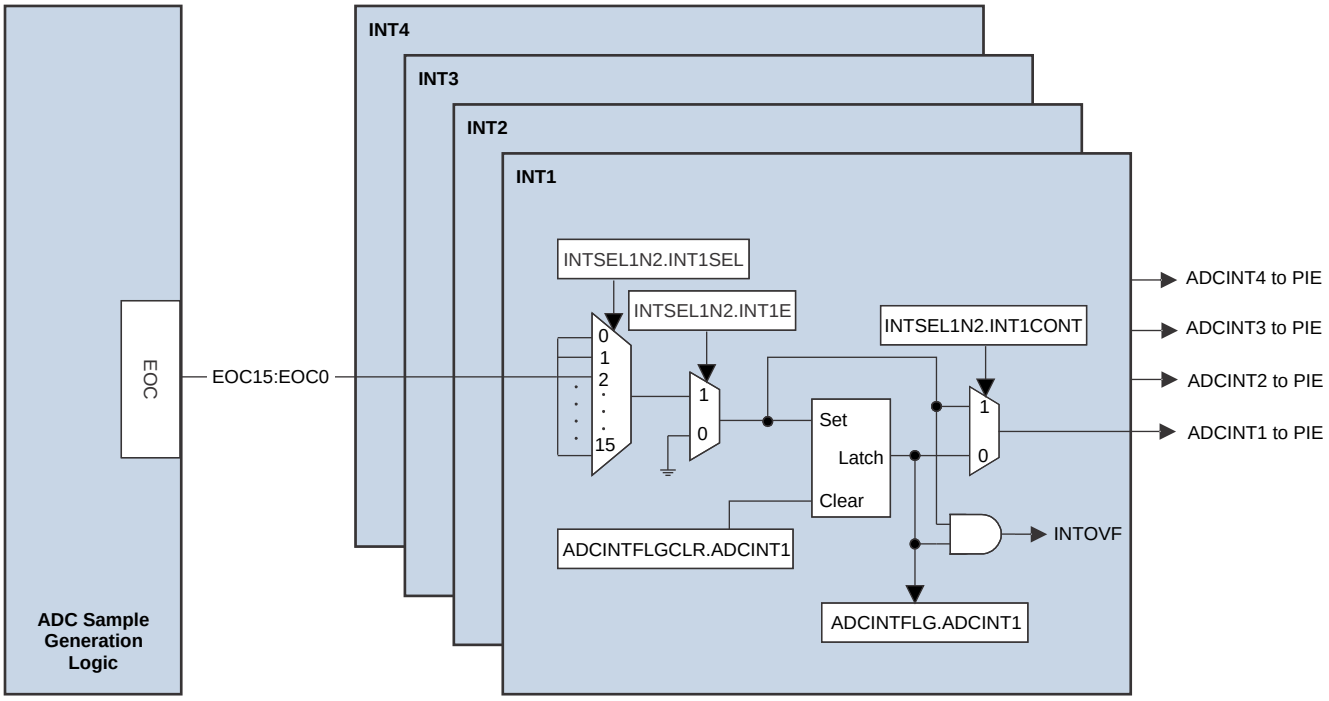
### 13.1.8 EOC and Interrupt Operation

Each SOC has a corresponding end-of-conversion (EOC) signal. This EOC signal can be used to trigger an ADC interrupt. The ADC can be configured to generate the EOC pulse at either the end of the acquisition window or at the end of the voltage conversion. This is configured using the bit INTPULSEPOS in the ADCCTL1 register. See Section 13.2, for exact EOC pulse location.

Each ADC module has 4 configurable ADC interrupts. These interrupts can be triggered by any of the 16 EOC signals. The flag bit for each ADCINT can be read directly or the interrupt can be passed on to the PIE.

Figure 13-7 shows a block diagram of the ADC interrupt structure.

**Figure 13-7. ADC EOC Interrupts**




#### 13.1.8.1 Early Interrupt Configuration Mode

In order to enable a lower latency from end of SOC to interrupt service routine, early interrupt generation mode may be invoked as described below:

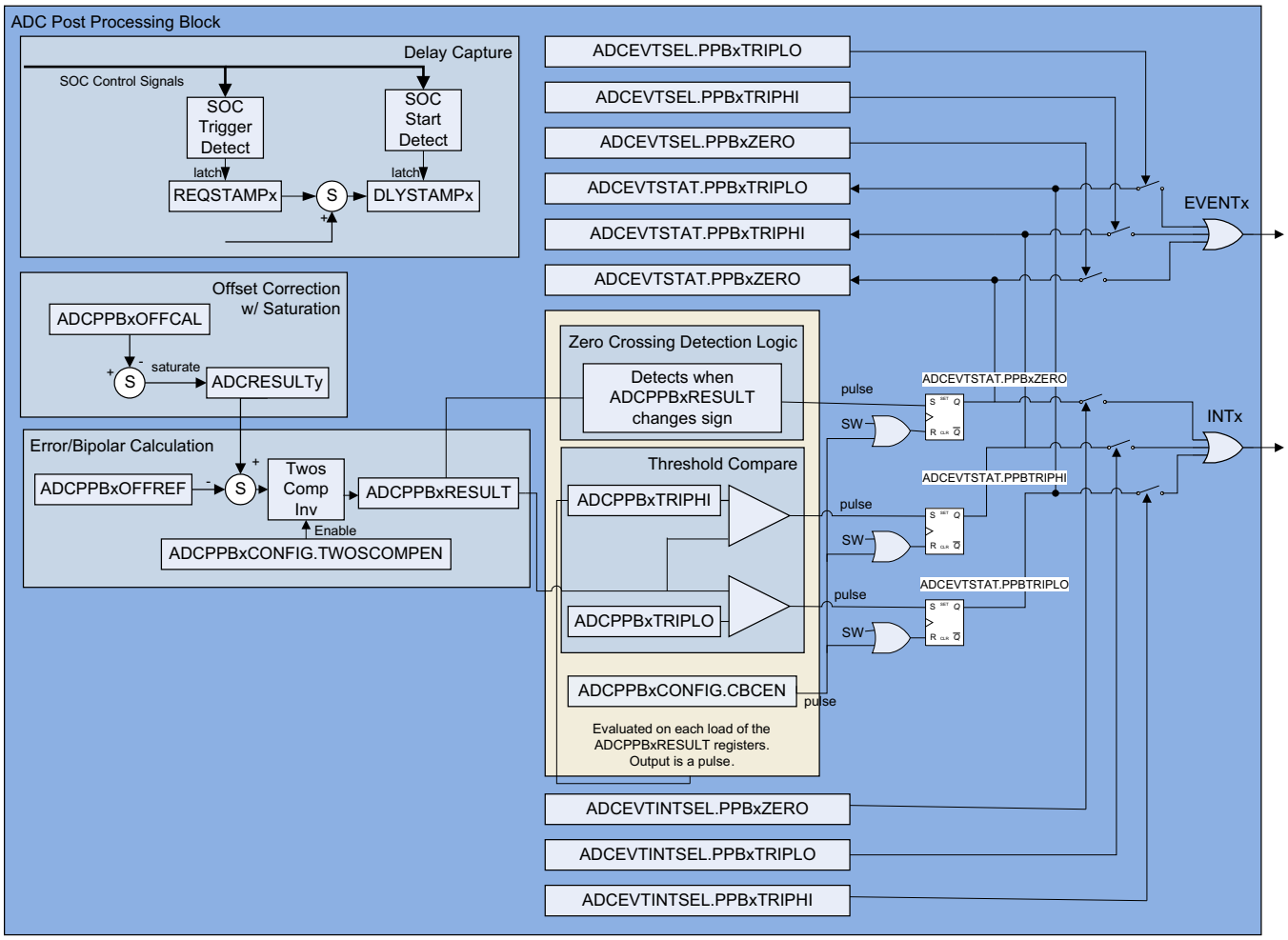
- Enable early interrupt generation mode by clearing the bit INTPULSEPOS to 0 in ADCCTL1
- The number of cycles in the conversion phase after the end of SOC where the interrupt becomes active is determined by the value written to the OFFSET field in ADCINTCYCLE. This will be in terms of SYSCLK cycles and starts at the falling edge of SOC.
- If the value of OFFSET goes beyond EOC, the ADC interrupt will be generated along with EOC.
- Writing values to OFFSET when INTPULSEPOS is set to 1 will not have any effect on the interrupt generation.

Refer to Figure 13-12 and Table 13-8 for more information on early interrupt, $t_{INT}$.

### 13.1.9 Post-Processing Blocks

Each ADC module contains four post-processing blocks (PPB). These blocks can be associated with any of the 16 RESULT registers using the ADCPPBxCONFIG.CONFIG bit field. Each PPB can simultaneously remove an offset associated with the ADCIN channel, subtract out a reference value, flag a zero-crossing point, and flag a high or low compare limit. Furthermore, the zero-crossing and compare flags can trip a PWM and/or generate an interrupt. A PPB is also capable of recording the delay between when the SOC associated with the PPB is triggered and when it actually begins to be sampled. Figure 13-8 presents the structure of each PPB. Subsequent sections explain the use of each submodule.

**Figure 13-8. ADC PPB Block Diagram**




#### 13.1.9.1 PPB Offset Correction

In many applications, external sensors and signal sources produce an offset. A global trimming of the ADC offset is not enough to compensate for these offsets, which vary from channel to channel. The post-processing block can remove these offsets with zero overhead, saving numerous cycles in tight control loops.

Offset correction is accomplished by first pointing the ADCPPBxCONFIG.CONFIG to the desired SOC, then writing an offset correction value to the ADCPPBxOFFCAL.OFFCAL register. The post-processing block will automatically add or subtract the value in the OFFCAL register from the raw conversion result and store it in the ADCRESULT register. This addition/subtraction will saturate at 0 on the low end and 4095 on the high end.

**NOTES:**

- Writing a 0 to the OFFCAL register effectively disables the offset correction feature, passing the raw

result unchanged to the ADCRESULT register.

- It is possible to point multiple PPBs to the same SOC. In this case, the OFFCAL value that will actually be applied will be that of the PPB with the highest number.
- In particular, care needs to be taken when using the PPB on SOC0, as all the PPB point to this SOC by default. This may cause unintentional overwriting of offset correction of a lower numbered PPB by a higher numbered PPB.

#### 13.1.9.2 PPB Error Calculation

In many applications, an error from a setpoint or expected value must be computed from the digital output of an ADC conversion. In other cases, a bipolar signal is necessary or convenient for control calculations. The PPB can perform these function automatically, reducing the sample to output latency and reducing software overhead.

Error calculation is accomplished by first pointing the ADCPPBxCONFIG.CONFIG to the desired SOC, then writing a value to the ADCPPBxOFFCAL.OFFREF register. The post-processing block will automatically subtract the value in the OFFREF register from the ADCRESULT value and store it in the ADCPPBxRESULT register. This subtraction will produce a sign-extended 32-bit result. It is also possible to selectively invert the calculated value before storing in the ADCPPBxRESULT register by setting the TWOSCOMPEN bit in the ADCPPBxCONFIG register.

**NOTES:**

- Do not write a value larger than 12 bits to the OFFREF register.
- Since the PPBxRESULT register is unique for each PPB, it is possible to point multiple PPBs to the same SOC and get different results for each PPB.
- Writing a 0 to the OFFREF register effectively disables the error calculation feature, passing the ADCRESULT value unchanged to the ADCPPBxRESULT register.

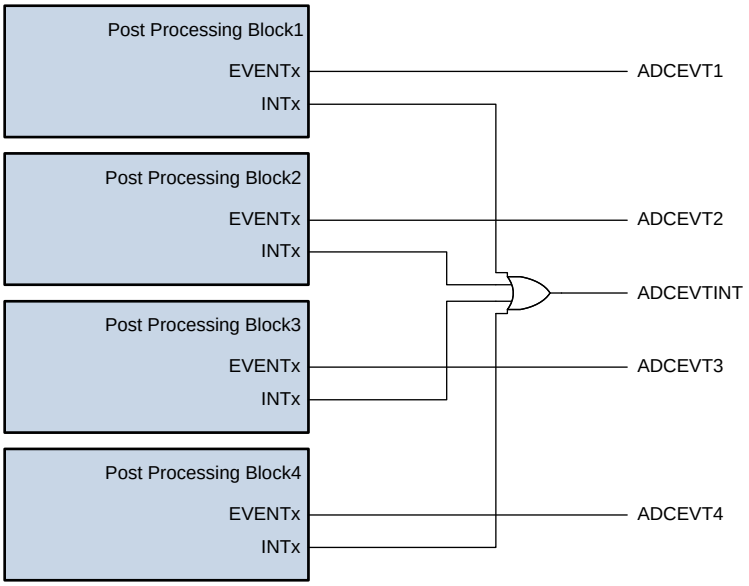
#### 13.1.9.3 PPB Limit Detection and Zero-Crossing Detection

Many applications perform a limit check against the ADC conversion results. The PPB can automatically perform a check against a high and low limit or whenever ADCPPBxRESULT changes sign. Based on these comparisons, it can generate a trip to the PWM and/or an interrupt automatically, lowering the sample to ePWM latency and reducing software overhead. This functionality also enables safety conscious applications to trip the ePWM based on an out-of-range ADC conversion without any CPU intervention.

To enable this functionality, first point the ADCPPBxCONFIG.CONFIG to the desired SOC, then write a value to one or both of the registers ADCPPBxTRIPHI.LIMITHI and ADCPPBxTRIPLO.LIMITLO (zero-crossing detection does not require further configuration). Whenever these limits are exceeded, the PPBxTRIPHI bit or PPBxTRIPLO bit will be set in the ADCEVTSTAT register. Whenever the ADCPPBxRESULT changes signs, the PPBxZERO bit will only be set in the ADCEVTSTAT register whenever the corresponding EOC signal occurs.. The ADCEVTCLR register has corresponding bits to clear these event flags. The ADCEVTSEL register has corresponding bits which allow the events to propagate through to the PWM. The ADCINTSEL register has corresponding bits which allow the events to propagate through to the PIE.

One PIE interrupt is shared between all the PPBs for a given ADC module as shown in Figure 13-9.

**Figure 13-9. ADC PPB Interrupt Event**




**NOTES:**

- Zero-crossing and limit compare reference the ADCPPBxRESULT register. This will include any correction applied by the OFFCAL and OFFREF registers. TRIPHI and TRIPLO do NOT perform a signed comparison. It is recommended to leave OFFREF as 0 when using limit compare functionality.
- If different actions need to be taken for different PPB events from the same ADC module, then the ADCEVTINT ISR will have to read the PPB event flags in the ADCEVTSTAT register to determine which event caused the interrupt.
- If different ePWM trips need to be generated separately for high compare, low compare, and/or zero-crossing, this can be achieved by pointing multiple PPBs to the same SOC.

### 13.1.9.4 PPB Sample Delay Capture

When multiple control loops are running asynchronously on the same ADC, there is a chance that an ADC request from two or more loops will collide, causing one of the samples to be delayed. This shows up as a measurement error in the system. By knowing when this delay occurs and the amount of delay that has occurred software can employ extrapolation techniques to reduce the error.

To this effect, each PPB has the field DLYSTAMP in the ADCPPBxSTAMP register. This field will contain the number of SYSCLK cycles between when the associate SOC was triggered and when it began converting.

This is achieved by having a global 12-bit free running counter based off of SYSCLK, which is in the field FREECOUNT in the ADCCOUNTER register. When the trigger for the associated SOC arrives, the value of this counter is loaded into the bit field ADCPPBxTRIPLO.REQSTAMP. When the actual sample window for that SOC begins, the value in REQSTAMP is subtracted from the current FREECOUNT value and stored in DLYSTAMP.

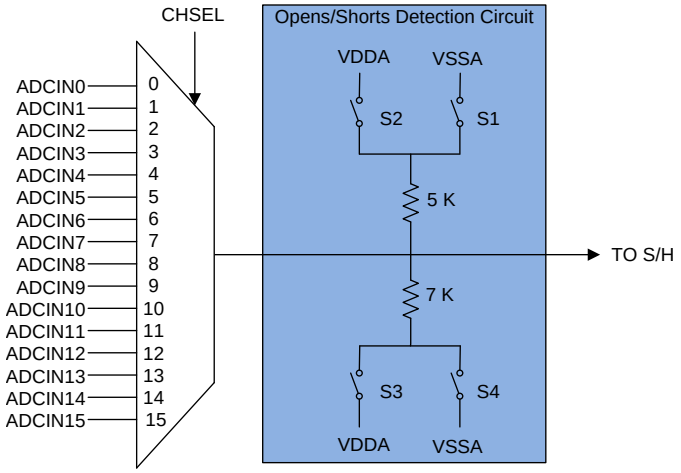
**NOTE:** If more than 4096 SYSCLK cycles elapse between the SOC trigger and the actual start of the SOC acquisition, the FREECOUNT register may overflow more than once, leading to incorrect DLYSTAMP value. Be cautious when using very slow conversions to prevent this from happening.

**NOTE:** The sample delay capture will not function if the associated SOC is triggered via software. It will, however, correctly record the delay if the software triggering of a different SOC causes the SOC associated with the PPB to be delayed

### 13.1.10 Opens/Shorts Detection Circuit (OSDETECT)

The opens/shorts detection circuit (OSDETECT) can be used to diagnose pin faults in the system. The circuit connects to the ADC input after the channel select multiplexer but before the S+H as shown in Figure 13-10. Note that the divider resistance tolerances can vary widely, hence this feature should not be used to check for conversion accuracy. Consult the device specification for implementation and availability of analog input channels.

**Figure 13-10. Opens/Shorts Detection Circuit**




The circuit can be operated by wiring a value to the DETECTCFG field in the ADCOSDETECT register. This will cause the circuit to source a voltage onto the input during the S+H phase of any conversion. The voltage and drive strength of the OSDETECT circuit for different DETECTCFG settings is given in Table 13-7.

**Table 13-7. DETECTCFG Settings**

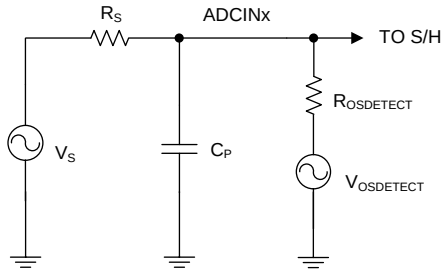
| ADCOSDETECT.DETECTCFG([1]) | Source Voltage | S4 | S3 | S2 | S1 | Drive Impedance |
|---|---|---|---|---|---|---|
| 0 | Off | Open | Open | Open | Open | Open |
| 1 | Zero Scale | Closed | Open | Open | Closed | 5K \|\| 7K |
| 2 | Full Scale | Open | Closed | Closed | Open | 5K \|\| 7K |
| 3 | 5/12 VDDA | Open | Closed | Open | Closed | 5K \|\| 7K |
| 4 | 7/12 VDDA | Closed | Open | Closed | Open | 5K \|\| 7K |
| 5 | Zero Scale with 5K pull down | Open | Open | Open | Closed | 5K |
| 6 | Full Scale with 5K pull up | Open | Open | Closed | Open | 5K |
| 7 | Zero scale with 7K pull down | Closed | Open | Open | Open | 7K |

(1) All combinations not listed are illegal.

#### 13.1.10.1 Implementation

A representative circuit with the OSDETECT implementation consists of the signal source with series resistance $R_S$, shunt capacitor $C_P$, the equivalent OSDETECT resistance $R_{OSDETECT}$ and voltage $V_{OSDETECT}$ is shown in Figure 13-11 and can be used as a basis to calculate the signal level going in to the sampling capacitor. $R_{OSDETECT}$ and $V_{OSDETECT}$ are the equivalent input resistance and voltage source contributed by the OSDETECT circuit with values shown in Table 13-7 for the different configuration settings. Refer to Figure 13-11 when deriving the input signal to S/H if signal source $V_S$ is driving while the OSDETECT feature is enabled.

**Figure 13-11. Input Circuit Equivalent with OSDETECT Enabled**




The input impedance $R_S$ and $C_P$ may be integral parts of the signal source or these could have been implemented in the design to precondition the signal or to control signal settling time to meet S/H requirements. The input path has to be considered when using the OSDETECT feature as this would affect the conversion results. For instance, driving an input signal when this feature is enabled would connect signal $V_S$ to the OSDETECT circuit through $R_S$ and affecting the ADC results. Larger $C_P$ values (in the order greater than hundreds of pF) would require using higher ACQPS to ensure that signal at the input has settled prior to conversion.

To enable the circuit:

1. Configure the ADC for conversion (for example, channel, SOC, ACQPS, prescaler, trigger, and so on).
2. Set up the ADCOSDETECT register for the desired voltage divider connection as shown in Table 13-7.
3. Initiate a conversion and inspect the conversion result.

   ***Note:*** You must interpret the results based on what is driving on the input side and what are the values of $R_S$ and $C_P$. If the $V_S$ signal can be disconnected from the input pin, the circuit can be used to detect open and shorted input pins as described in the following sections.

first then

#### 13.1.10.2 Detecting an Open Input Pin

By cycling through the various OSDETECT settings, the input signal will be pulled towards the sourced voltages. An input with good drive strength (pin not open) will be minimally affected. However, if the pin is open, the sampled voltages will be close to the source voltages specified in Table 13-7.

#### 13.1.10.3 Detecting a Shorted Input Pin

By cycling through the various OSDETECT settings, the input signal will be pulled towards the sourced voltages. An input with finite drive strength (pin not shorted) will be pulled toward each sourced voltage. However, if the pin is shorted, the signal will remain at the same voltage.

### *13.1.11 Power-Up Sequence*

Upon device power-up or system level reset, the ADC will be powered down and disabled. When powering up the ADC, use the following sequence:

1. Set the bit to enable the desired ADC clock in the PCLKCR13 register.
2. Set the desired ADC clock divider in the PRESCALE field of ADCCTL2.
3. Power up the ADC by setting the ADCPWDNZ bit in ADCCTL1.
4. If using internal reference mode, set the REFSEL bit to 1
5. Allow at least 1ms delay before sampling.

If multiple ADCs are powered up simultaneously, steps 1 and step 3 can each be done for all ADCs in one write instruction. Also, only one delay is necessary as long as it occurs after all the ADCs have begun powering up.

### 13.1.12 ADC Calibration

During the fabrication and test process, Texas Instruments calibrates the gain, offset, and linearity of the ADCs, the offset of the buffered DACs and the gain and offset of the PGAs. These trim settings are embedded into TI reserved OTP memory as part of C-callable functions.

- The Device_cal() function copies the trim values for ADC linearity, ADC offset, PGA gain and offset, and DAC offset from OTP memory to their respective trim registers.
- The getTempSlope() function returns slope and getTempOffset() function returns offset values from OTP memory which the user can reference to in calculating temperature measured by the internal temperature sensor. getTemperatureC(int16 sensorSample) function translates an ADC sample of the temperature sensor to a temperature. The sensor reading needs to be scaled to a VREFHI value of 2.5V.

Until the appropriate factory trim is loaded, the ADC (and other modules) will not be guaranteed to operate within datasheet specifications. Similarly, if trim values other than the factory settings are placed into the trim registers, the ADC (and other modules) will not be guaranteed to operate within datasheet specifications.

The boot ROM will call the calibration functions, so trim values should be initially populated without user intervention. However, if the trims are cleared due to a module reset or modified for some other reason, then the user can call the calibration functions (defined in the headerfiles).

#### 13.1.12.1 ADC Zero Offset Calibration

Zero offset error is defined as the difference from 0 that occurs when converting a voltage at VREFLO. The zero offset error can be positive or negative. To correct this error, an adjustment of equal magnitude and opposite polarity is written into the ADCOFFTRIM register. The value contained in this register will be applied before the results are available in the ADC result registers. This operation is fully contained within the ADC core, so the timing of the results will not be affected and the full dynamic range of the ADC will be maintained for any trim value.

Using the GetAdcOffsetTrimOTP(Uint16) function, the ADCOFFTRIM register can be loaded with the factory calibrated offset error correction. The user can modify the ADCOFFTRIM register to compensate for additional offset error induced by the application environment if desired, but this is not typically necessary to achieve datasheet specified performance.

**NOTE:** Regardless of the converter resolution, the size of each ADCOFFTRIM step is (VREFHI-VREFLO)/65536.

Use the following procedure to re-calibrate the ADC offset in 12-bit, single-ended mode:

1. Set ADCOFFTRIM to +112 steps (0x70). This adds an artificial offset to account for negative offset that may reside in the ADC core.
2. Perform some multiple of 16 conversions on VREFLO (internal connection), accumulating the results (for example, 32*16 conversions = 512 conversions).
3. Divide the accumulated result by the multiple of 16 (for example, for 512 conversions, divide by 32).
4. Set ADCOFFTRIM to 112 – result from step 3.

## 13.2 ADC Timings

The process of converting an analog voltage to a digital value is broken down into an S+H phase and a conversion phase. The ADC sample and hold circuits (S+H) are clocked by SYSCLK while the ADC conversion process is clocked by ADCCLK. ADCCLK is generated by dividing down SYSCLK based on the PRESCALE field in the ADCCTL2 register.

The S+H duration is the value of the ACQPS field of the SOC being converted, plus one, times the SYSCLK period. The user must ensure that this duration exceeds both 1 ADCCLK period and the minimum S+H duration specified in the datasheet. The conversion time is approximately 10.5 ADCCLK cycles. The exact conversion time is always a whole number of SYSCLK cycles. See the timing diagrams and tables in Section 13.2.1 for exact timings.

### 13.2.1 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).

**Figure 13-12. ADC Timings for 12-bit Mode**

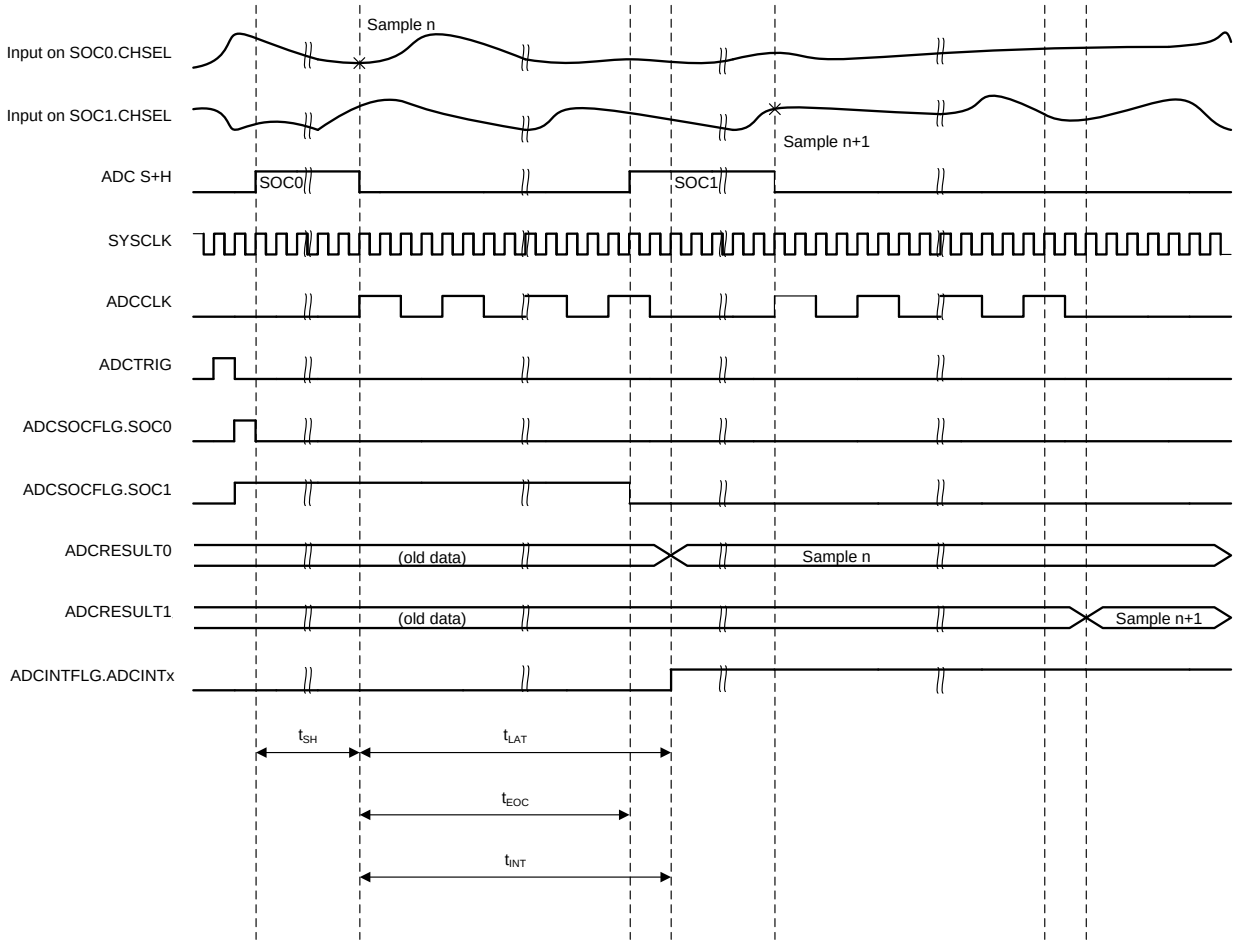



**Table 13-8. ADC Timings in 12-bit Mode**

| ADCCLK PRESCALE | | SYSCLK CYCLES | | | | ADCCLK CYCLES |
|---|---|---|---|---|---|---|
| ADCCTL2 [PRESCALE] | RATIO ADCCLK:SYSCLK | $t_{EOC}$ | $t_{LAT}$ | $t_{INT(EARLY)}$ (1) | $t_{INT(LATE)}$ | $t_{EOC}$ |
| 0 | 1 | 11 | 13 | 1 | 11 | 11 |
| 2 | 2 | 21 | 23 | 1 | 21 | 10.5 |
| 4 | 3 | 31 | 34 | 1 | 31 | 10.3 |
| 6 | 4 | 41 | 44 | 1 | 41 | 10.3 |
| 8 | 5 | 51 | 55 | 1 | 51 | 10.2 |
| 10 | 6 | 61 | 65 | 1 | 61 | 10.2 |
| 12 | 7 | 71 | 76 | 1 | 71 | 10.1 |
| 14 | 8 | 81 | 86 | 1 | 81 | 10.1 |

(1) By default, $t_{INT}$ will occur 1 SYSCLK cycle after S+H window if INTPULSEPOS is 0. This can be changed by writing to OFFSET field in the ADCINTCYCLE register.

## 13.3 Additional Information

The following sections contain additional practical information.

### *13.3.1 Ensuring Synchronous Operation*

For best performance, all ADCs on the device should be operated synchronously. The device datasheet specifies the performance in both synchronous and asynchronous mode for those parameters which differ between the modes of operation.

To ensure synchronous operation, all ADCs on the device should operate in lockstep. This is accomplished by writing configurations to all ADCs that cause the sampling and conversion phases of all ADCs to be exactly aligned. The easiest way to accomplish this is to write identical values to the SOC configurations for each ADC for trigger select and ACQPS (S+H duration).

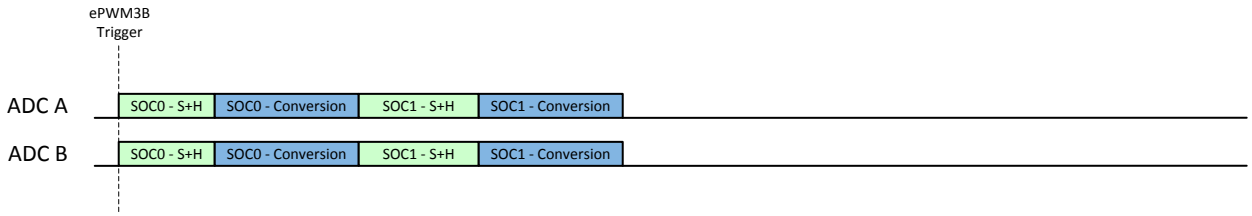
#### 13.3.1.1 Basic Synchronous Operation

The below example configures two SOCs each on ADCA and ADCB with identical trigger select and ACQPS values. This will result in synchronous operation between ADCA and ADCB. For devices with more than two ADCs, the same principles can be used to synchronize all the ADCs.

**Example 1: Basic Synchronous Operation**

```
AdcaRegs.ADCSOC0CTL.bit.CHSEL = 4;    //SOC0 will convert ADCINA4
AdcaRegs.ADCSOC0CTL.bit.ACQPS = 19;   //SOC0 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 10; //SOC0 will begin conversion on ePWM3 SOCB
AdcbRegs.ADCSOC0CTL.bit.CHSEL = 0;    //SOC0 will convert ADCINB0
AdcbRegs.ADCSOC0CTL.bit.ACQPS = 19;   //SOC0 will use sample duration of 20 SYSCLK cycles
AdcbRegs.ADCSOC0CTL.bit.TRIGSEL = 10; //SOC0 will begin conversion on ePWM3 SOCB

AdcaRegs.ADCSOC1CTL.bit.CHSEL = 4;    //SOC1 will convert ADCINA4
AdcaRegs.ADCSOC1CTL.bit.ACQPS = 30;   //SOC1 will use sample duration of 31 SYSCLK cycles
AdcaRegs.ADCSOC1CTL.bit.TRIGSEL = 10; //SOC1 will begin conversion on ePWM3 SOCB
AdcbRegs.ADCSOC1CTL.bit.CHSEL = 1;    //SOC1 will convert ADCINB1
AdcbRegs.ADCSOC1CTL.bit.ACQPS = 30;   //SOC1 will use sample duration of 31 SYSCLK cycles
AdcbRegs.ADCSOC1CTL.bit.TRIGSEL = 10; //SOC1 will begin conversion on ePWM3 SOCB
```

**Figure 13-13. Example 1: Basic Synchronous Operation**




Several things should be noted from Figure 13-13. 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.

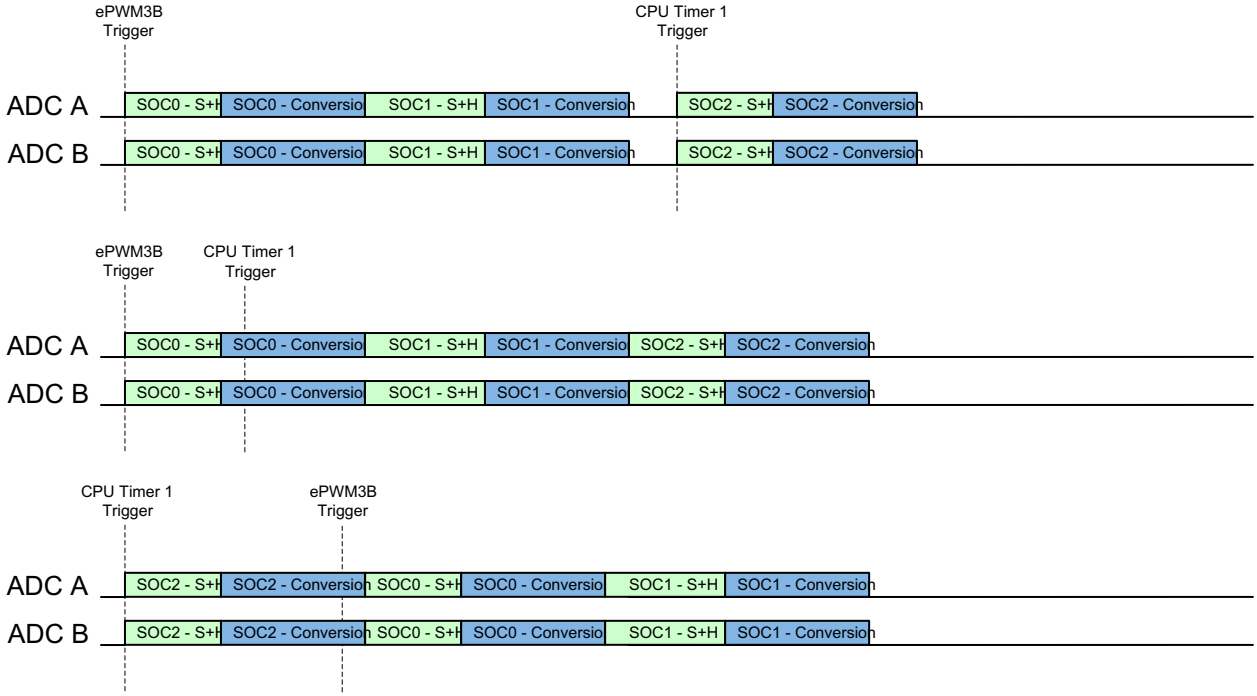
#### 13.3.1.2 Synchronous Operation with Multiple Trigger Sources

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 below example demonstrates synchronous operation between ADCA and ADCB while using three SOCs and two trigger sources. Figure 13-14 demonstrates that any combination of relative trigger timings still results in synchronous operation.

```
Example 2: Synchronous Operation With Multiple Trigger Sources

AdcaRegs.ADCSOC0CTL.bit.CHSEL = 4;    //SOC0 will convert ADCINA4
AdcaRegs.ADCSOC0CTL.bit.ACQPS = 19;   //SOC0 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 10; //SOC0 will begin conversion on ePWM3 SOCB
AdcbRegs.ADCSOC0CTL.bit.CHSEL = 0;    //SOC0 will convert ADCINB0
AdcbRegs.ADCSOC0CTL.bit.ACQPS = 19;   //SOC0 will use sample duration of 20 SYSCLK cycles
AdcbRegs.ADCSOC0CTL.bit.TRIGSEL = 10; //SOC0 will begin conversion on ePWM3 SOCB

AdcaRegs.ADCSOC1CTL.bit.CHSEL = 4;    //SOC1 will convert ADCINA4
AdcaRegs.ADCSOC1CTL.bit.ACQPS = 30;   //SOC1 will use sample duration of 31 SYSCLK cycles
AdcaRegs.ADCSOC1CTL.bit.TRIGSEL = 10; //SOC1 will begin conversion on ePWM3 SOCB
AdcbRegs.ADCSOC1CTL.bit.CHSEL = 1;    //SOC1 will convert ADCINB1
AdcbRegs.ADCSOC1CTL.bit.ACQPS = 30;   //SOC1 will use sample duration of 31 SYSCLK cycles
AdcbRegs.ADCSOC1CTL.bit.TRIGSEL = 10; //SOC1 will begin conversion on ePWM3 SOCB

AdcaRegs.ADCSOC2CTL.bit.CHSEL = 0;    //SOC2 will convert ADCINA0
AdcaRegs.ADCSOC2CTL.bit.ACQPS = 19;   //SOC2 will use sample duration of 31 SYSCLK cycles
AdcaRegs.ADCSOC2CTL.bit.TRIGSEL = 2;  //SOC2 will begin conversion on CPU Timer1
AdcbRegs.ADCSOC2CTL.bit.CHSEL = 2;    //SOC2 will convert ADCINB2
AdcbRegs.ADCSOC2CTL.bit.ACQPS = 19;   //SOC2 will use sample duration of 31 SYSCLK cycles
AdcbRegs.ADCSOC2CTL.bit.TRIGSEL = 2;  //SOC2 will begin conversion on CPU Timer1
```

**Figure 13-14. Example 2: Synchronous Operation with Multiple Trigger Sources**




Note that any trigger source that can be selected in the TRIGSEL field can be used except for software triggering. There is no way to issue the software triggers for all ADCs simultaneously, so it will likely result in asynchronous operation. ADCINT1 or ADCINT2 can also be used as a trigger as long as the ADCINTSOCSEL1 and ADCINTSOCSEL2 registers are configured identically for all ADCs and software triggering is not used to start the chain of conversions.

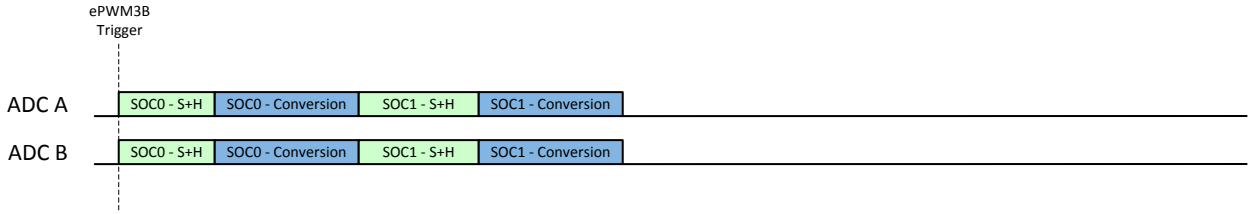
### 13.3.1.3 Synchronous Operation with Uneven SOC Numbers

If only one trigger source is used, one ADC can use more SOCs than the other ADCs while still operating synchronously.

**Example 3: Synchronous Operation With Uneven SOC Numbers**

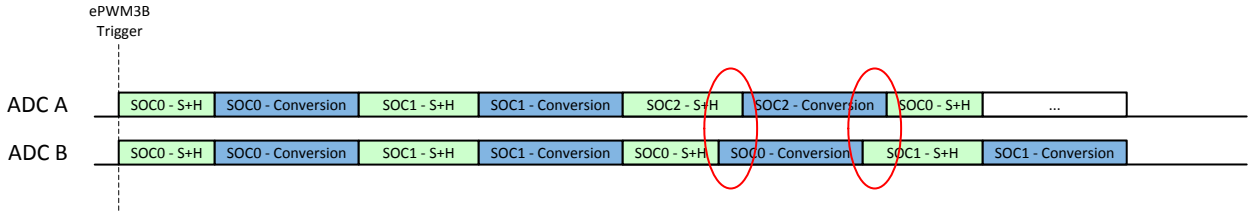
```
AdcaRegs.ADCSOC0CTL.bit.CHSEL = 4;    //SOC0 will convert ADCINA4
AdcaRegs.ADCSOC0CTL.bit.ACQPS = 19;   //SOC0 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 10; //SOC0 will begin conversion on ePWM3 SOCB
AdcbRegs.ADCSOC0CTL.bit.CHSEL = 0;    //SOC0 will convert ADCINB0
AdcbRegs.ADCSOC0CTL.bit.ACQPS = 19;   //SOC0 will use sample duration of 20 SYSCLK cycles
AdcbRegs.ADCSOC0CTL.bit.TRIGSEL = 10; //SOC0 will begin conversion on ePWM3 SOCB

AdcaRegs.ADCSOC1CTL.bit.CHSEL = 4;    //SOC1 will convert ADCINA4
AdcaRegs.ADCSOC1CTL.bit.ACQPS = 30;   //SOC1 will use sample duration of 31 SYSCLK cycles
AdcaRegs.ADCSOC1CTL.bit.TRIGSEL = 10; //SOC1 will begin conversion on ePWM3 SOCB
AdcbRegs.ADCSOC1CTL.bit.CHSEL = 1;    //SOC1 will convert ADCINB1
AdcbRegs.ADCSOC1CTL.bit.ACQPS = 30;   //SOC1 will use sample duration of 31 SYSCLK cycles
AdcbRegs.ADCSOC1CTL.bit.TRIGSEL = 10; //SOC1 will begin conversion on ePWM3 SOCB

AdcaRegs.ADCSOC2CTL.bit.CHSEL = 0; /   /SOC2 will convert ADCINA0
AdcaRegs.ADCSOC2CTL.bit.ACQPS = 19;   //SOC2 will use sample duration of 31 SYSCLK cycles
AdcaRegs.ADCSOC2CTL.bit.TRIGSEL = 10; //SOC2 will begin conversion on ePWM3 SOCB
```

**Figure 13-15. Example 3A: Synchronous Operation with Uneven SOC Numbers**




Note that if the trigger comes again before all SOCs have completed their conversions, ADCB will begin converting immediately on SOC0 while ADCA will not start converting SOC0 again until SOC2 is complete. This will result in asynchronous operation, so care must be taken to not overflow the trigger.

**Figure 13-16. Example 3B: Asynchronous Operation with Uneven SOC Numbers – Trigger Overflow**



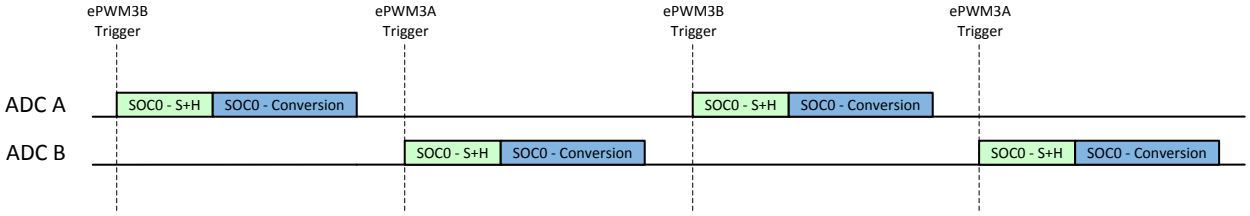

### 13.3.1.4 Non-overlapping Conversions

If conversion timings can be guaranteed to not overlap by the user, then it is not necessary to configure all SOCs identically on all ADCs to achieve performance equivalent to synchronous operation. For example, if the two ADC triggers in a system come from two ePWM sources which are always 180 degrees out-of-phase, then SOC0 could be used for both ADCA and ADCB with different trigger sources and different ACQPS values.

**Example 5: Operation with Non-overlapping Conversions**

```
//ePWM3 SOCA and SOCB are 180 degrees out of phase
AdcaRegs.ADCSOC0CTL.bit.CHSEL = 4;    //SOC0 will convert ADCINA4
AdcaRegs.ADCSOC0CTL.bit.ACQPS = 19;   //SOC0 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 10; //SOC0 will begin conversion on ePWM3 SOCB
AdcbRegs.ADCSOC0CTL.bit.CHSEL = 0;    //SOC0 will convert ADCINB0
AdcbRegs.ADCSOC0CTL.bit.ACQPS = 19;   //SOC0 will use sample duration of 20 SYSCLK cycles
AdcbRegs.ADCSOC0CTL.bit.TRIGSEL = 9;  //SOC0 will begin conversion on ePWM3 SOCA
```

**Figure 13-17. Example 5: Synchronous Equivalent Operation with Non-Overlapping Conversions**




### 13.3.2 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 will be charged to within ½ LSB or ¼ LSB of the final value, depending on the tolerable settling error.

An approximation of the required settling time can be determined using an RC settling model. The time constant for the model is given by the equation:

$$\tau = (Rs + Ron) \cdot Ch + Rs \cdot (Cs + Cp)$$

And the number of time constants needed is given by the equation:

$$k = In\left(\frac{2n}{settling\ error}\right) - In\left(\frac{C_S + C_P}{CH}\right)$$

So the total S+H time should be set to at least:

$t = k \cdot \tau$

Where the following parameters are provided by the ADC input model in the device data manual:

- n = ADC resolution (in bits)
- $R_{ON}$ = ADC sampling switch resistance (in Ohms)
- $C_H$ = ADC sampling capacitor (in pF)
- $C_p$ = ADC channel parasitic input capacitance (in pF)

And the following parameters are dependent on the application design:

- settling error = tolerable settling error (in LSBs)
- $R_s$ = ADC driving circuit source impedance (in Ohms)
- $C_S$ = capacitance on ADC input pin (in pF)

For example, assuming the following parameters:

- n = 12-bits
- $R_{ON}$ = 500Ω
- $C_H$ = 12.5pF
- $C_p$ = 12.7pF
- settling error = ¼ LSB
- $R_s$ = 180Ω
- $C_s$ = 150pF

The time constant would be calculated as:

$$\tau = (180\Omega + 500\Omega) \cdot 12.5pF + 180\Omega \cdot (150pF + 12.7pF) = 8.5ns + 29.3ns = 37.8ns$$

And the number of required time constants would be:

$$k = ln\left(\frac{2^{12}}{0.25}\right) - ln\left(\frac{150pF + 12.7pF}{12.5pF}\right) = 9.70 - 2.57 = 7.13$$

So the S+H time should be set to at least:

37.8ns · 7.13 = 270ns

If SYSCLK = 100 MHz, then each SYSCLK is 10ns. S+H duration should be 270ns/10ns = 27.0 SYSCLKs so ACQPS for this input should be set to at least CEILING(27.0) – 1 = 26.

While this gives a rough estimate of the required acquisition window, a better method would be to setup a circuit with the ADC input model, a model of the source impedance/capacitance, and any board parasitics in SPICE (or similar software) and simulate to verify that the sampling capacitor settles to the desired accuracy.

**NOTE:** The device data manual will specify a minimum ADC S+H window duration. Do not use an ACQPS value that gives a duration less than this specification.

### 13.3.3 Achieving Simultaneous Sampling

While the Type 5 ADC does not have dual S+H circuits, it is easy to achieve simultaneous sampling. This is accomplished by setting the SOC triggers on two or more ADC modules to use the same trigger source. The example below demonstrates x3 simultaneous sampling based on an ePWM3 event. ADCINA3, ADCINB5, and ADCINC2 are sampled. An acquisition window of 20 SYSCLK cycles is used, but different durations are possible.

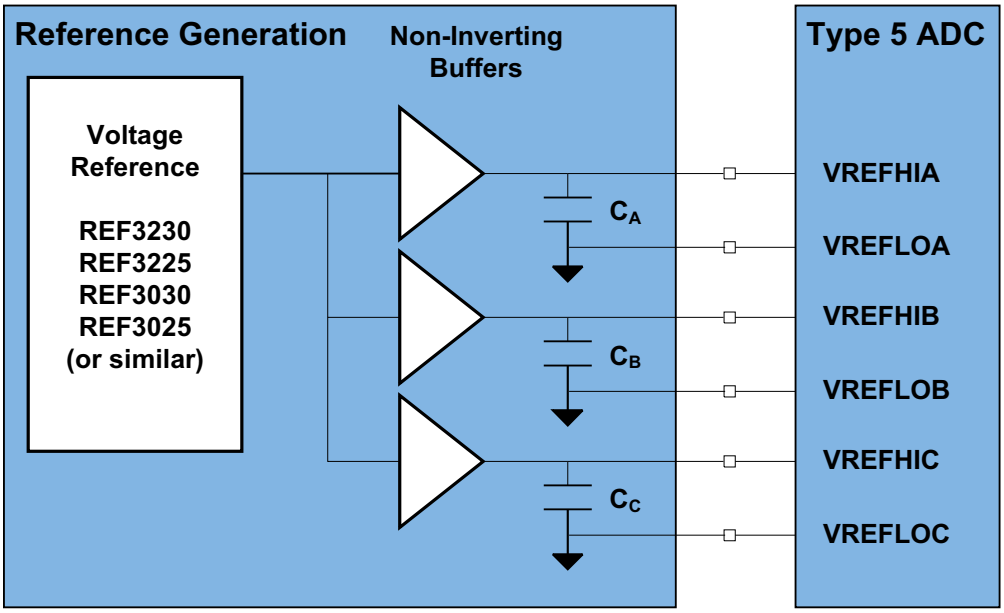
```
AdcaRegs.ADCSOC0CTL.bit.CHSEL = 3;        //SOC0 will convert ADCINA3
AdcaRegs.ADCSOC0CTL.bit.ACQPS = 19;       //SOC0 will use sample duration of 20 SYSCLK cycles
AdcaRegs.ADCSOC0CTL.bit.TRIGSEL = 10;     //SOC0 will begin conversion on ePWM3 SOCB
AdcbRegs.ADCSOC0CTL.bit.CHSEL = 5;        //SOC0 will convert ADCINB5
AdcbRegs.ADCSOC0CTL.bit.ACQPS = 19;       //SOC0 will use sample duration of 20 SYSCLK cycles
AdcbRegs.ADCSOC0CTL.bit.TRIGSEL = 10;     //SOC0 will begin conversion on ePWM3 SOCB
AdccRegs.ADCSOC0CTL.bit.CHSEL = 2;        //SOC0 will convert ADCINC2
AdccRegs.ADCSOC0CTL.bit.ACQPS = 19;       //SOC0 will use sample duration of 20 SYSCLK cycles
AdccRegs.ADCSOC0CTL.bit.TRIGSEL = 10;     //SOC0 will begin conversion on ePWM3 SOCB
```

When the ePWM3 trigger is received, all three ADCs will begin converting in parallel immediately. All results will be in the ADCRESULT0 register for each ADC. Note that this assumes that all ADCs are idle when the trigger is received. If one or more ADCs is busy, the samples will not happen at exactly the same time.

### 13.3.4 Designing an External Reference Circuit

One external reference circuit can be shared by one or more ADC module. This is desirable to keep ADC reference mismatch minimal, and to save on the number of components necessary in the system. A block diagram for a shared reference system is presented below in Figure 13-18.

**Figure 13-18. Shared Reference System**




### *13.3.5 Internal Temperature Sensor*

The internal temperature sensor measures the junction temperature of the device. The output of the sensor can be sampled with the ADC through an internal connection. This can be enabled on channel ADCIN14 on ADCB by setting the ENABLE bit in the TSNSCTL registers.

## 13.4 Registers

### 13.4.1 Analog to Digital Base Addresses

**Table 13-9. ADC Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| AdcaResultRegs | ADC_RESULT_REGS | 0x0000_0B00 | 0x0000_0B1F |
| AdcbResultRegs | ADC_RESULT_REGS | 0x0000_0B20 | 0x0000_0B3F |
| AdccResultRegs | ADC_RESULT_REGS | 0x0000_0B40 | 0x0000_0B5F |
| AdcaRegs | ADC_REGS | 0x0000_7400 | 0x0000_747F |
| AdcbRegs | ADC_REGS | 0x0000_7480 | 0x0000_74FF |
| AdccRegs | ADC_REGS | 0x0000_7500 | 0x0000_757F |

### 13.4.1.1 ADC_REGS Registers

Table 13-10 lists the memory-mapped registers for the ADC_REGS. All register offset addresses not listed in Table 13-10 should be considered as reserved locations and the register contents should not be modified.

**Table 13-10. 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 |
| 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 |

**Table 13-10. ADC_REGS Registers (continued)**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 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 |
| 6Fh | ADCINTCYCLE | ADC Early Interrupt Generation Cycle | 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 |

Complex bit access types are encoded to fit into small table cells. Table 13-11 shows the codes that are used for access types in this section.

**Table 13-11. ADC_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| W=1 | W | Write |
| **Reset or Default Value** | | |
| *-n* | | Value after reset or the default value |
| **Register Array Variables** | | |

**Table 13-11. ADC_REGS Access Type Codes (continued)**

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

#### 13.4.1.1.1 ADCCTL1 Register (Offset = 0h) [reset = 0h]

ADCCTL1 is shown in Figure 13-19 and described in Table 13-12.

Return to Summary Table.

ADC Control 1 Register

**Figure 13-19. ADCCTL1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | ADCBSY | RESERVED | ADCBSYCHN | | | |
| R-0h | | R-0h | R-0h | R-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ADCPWDNZ | RESERVED | | | | INTPULSEPOS | RESERVED | |
| R/W-0h | R-0h | | | | R/W-0h | R-0h | |

**Table 13-12. 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 |

**Table 13-12. ADCCTL1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | INTPULSEPOS | R/W | 0h | ADC Interrupt Pulse Position.<br>0 Interrupt pulse generation occurs when ADC begins conversion (at the end of the acquisition window) plus a number of SYSCLK cycles as specified in the ADCINTCYCLE.OFFSET register.<br>1 Interrupt pulse generation occurs at the end of the conversion, 1 cycle prior to the ADC result latching into its result register<br>Reset type: SYSRSn |
| 1-0 | RESERVED | R | 0h | Reserved |

#### 13.4.1.1.2 ADCCTL2 Register (Offset = 1h) [reset = 0h]

ADCCTL2 is shown in Figure 13-20 and described in Table 13-13.

Return to Summary Table.

ADC Control 2 Register

**Figure 13-20. ADCCTL2 Register**

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

**Table 13-13. ADCCTL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12-8 | RESERVED | R | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | RESERVED | R | 0h | Reserved |
| 5-4 | RESERVED | R | 0h | Reserved |
| 3-0 | PRESCALE | R/W | 0h | ADC Clock Prescaler.<br>0000 ADCCLK = Input Clock / 1.0<br>0001 Reserved<br>0010 ADCCLK = Input Clock / 2.0<br>0011 Reserved<br>0100 ADCCLK = Input Clock / 3.0<br>0101 Reserved<br>0110 ADCCLK = Input Clock / 4.0<br>0111 Reserved<br>1000 ADCCLK = Input Clock / 5.0<br>1001 Reserved<br>1010 ADCCLK = Input Clock / 6.0<br>1011 Reserved<br>1100 ADCCLK = Input Clock / 7.0<br>1101 Reserved<br>1110 ADCCLK = Input Clock / 8.0<br>1111 Reserved<br>Reset type: SYSRSn |

#### 13.4.1.1.3 ADCBURSTCTL Register (Offset = 2h) [reset = 0h]

ADCBURSTCTL is shown in Figure 13-21 and described in Table 13-14.

Return to Summary Table.

ADC Burst Control Register

**Figure 13-21. ADCBURSTCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| BURSTEN | RESERVED | | | BURSTSIZE | | | |
| R/W-0h | R-0h | | | R/W-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | BURSTTRIGSEL | | | | | |
| R-0h | | R/W-0h | | | | | |

**Table 13-14. 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 |

**Table 13-14. 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>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 - 3Fh - Reserved<br>Reset type: SYSRSn |

#### 13.4.1.1.4 ADCINTFLG Register (Offset = 3h) [reset = 0h]

ADCINTFLG is shown in Figure 13-22 and described in Table 13-15.

Return to Summary Table.

ADC Interrupt Flag Register

**Figure 13-22. ADCINTFLG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | ADCINT4 | ADCINT3 | ADCINT2 | ADCINT1 |
| R-0h | | | | R-0h | R-0h | R-0h | R-0h |

**Table 13-15. 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 continuous mode (INTSELxNy register) then further interrupt pulses are generated whenever a selected EOC event occurs even if the flag bit is set. If the continuous mode is not enabled, then no further interrupt pulses are generated until the user clears this flag bit using the ADCINFLGCLR register. Rather, an ADC interrupt overflow event occurs in the ADCINTOVF register.<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 continuous mode (INTSELxNy register) then further interrupt pulses are generated whenever a selected EOC event occurs even if the flag bit is set. If the continuous mode is not enabled, then no further interrupt pulses are generated until the user clears this flag bit using the ADCINFLGCLR register. Rather, an ADC interrupt overflow event occurs in the ADCINTOVF register.<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 continuous mode (INTSELxNy register) then further interrupt pulses are generated whenever a selected EOC event occurs even if the flag bit is set. If the continuous mode is not enabled, then no further interrupt pulses are generated until the user clears this flag bit using the ADCINFLGCLR register. Rather, an ADC interrupt overflow event occurs in the ADCINTOVF register.<br>Reset type: SYSRSn |

**Table 13-15. 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 continuous mode (INTSELxNy register) then further interrupt pulses are generated whenever a selected EOC event occurs even if the flag bit is set. If the continuous mode is not enabled, then no further interrupt pulses are generated until the user clears this flag bit using the ADCINFLGCLR register. Rather, an ADC interrupt overflow event occurs in the ADCINTOVF register.<br>Reset type: SYSRSn |

#### 13.4.1.1.5 ADCINTFLGCLR Register (Offset = 4h) [reset = 0h]

ADCINTFLGCLR is shown in Figure 13-23 and described in Table 13-16.

Return to Summary Table.

ADC Interrupt Flag Clear Register

**Figure 13-23. ADCINTFLGCLR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | ADCINT4 | ADCINT3 | ADCINT2 | ADCINT1 |
| R-0h | | | | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 13-16. ADCINTFLGCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | ADCINT4 | R=0/W=1 | 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 but the overflow bit will not be affected (retains current state)<br>Boundary condition: If hardware is trying to set the bit flag while software tries to clear the bit in the same cycle, the following will take place:<br>1. SW has prioirity and will clear the flag<br>2. HW set will be discarded<br>no signal will propagate to the PIE from the latch<br>3. Overflow flag/condition will be generated<br>Reset type: SYSRSn |
| 2 | ADCINT3 | R=0/W=1 | 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 but the overflow bit will not be affected (retains current state)<br>Boundary condition: If hardware is trying to set the bit flag while software tries to clear the bit in the same cycle, the following will take place:<br>1. SW has prioirity and will clear the flag<br>2. HW set will be discarded<br>no signal will propagate to the PIE from the latch<br>3. Overflow flag/condition will be generated<br>Reset type: SYSRSn |
| 1 | ADCINT2 | R=0/W=1 | 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 but the overflow bit will not be affected (retains current state)<br>Boundary condition: If hardware is trying to set the bit flag while software tries to clear the bit in the same cycle, the following will take place:<br>1. SW has prioirity and will clear the flag<br>2. HW set will be discarded<br>no signal will propagate to the PIE from the latch<br>3. Overflow flag/condition will be generated<br>Reset type: SYSRSn |

**Table 13-16. ADCINTFLGCLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | ADCINT1 | R=0/W=1 | 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 but the overflow bit will not be affected (retains current state)<br>Boundary condition: If hardware is trying to set the bit flag while software tries to clear the bit in the same cycle, the following will take place:<br>1. SW has prioirity and will clear the flag<br>2. HW set will be discarded<br>no signal will propagate to the PIE from the latch<br>3. Overflow flag/condition will be generated<br>Reset type: SYSRSn |

#### 13.4.1.1.6 ADCINTOVF Register (Offset = 5h) [reset = 0h]

ADCINTOVF is shown in Figure 13-24 and described in Table 13-17.

Return to Summary Table.

ADC Interrupt Overflow Register

**Figure 13-24. ADCINTOVF Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | ADCINT4 | ADCINT3 | ADCINT2 | ADCINT1 |
| R-0h | | | | R-0h | R-0h | R-0h | R-0h |

**Table 13-17. 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 |

**Table 13-17. ADCINTOVF Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | ADCINT1 | R | 0h | ADC Interrupt 1 Overflow Flags<br><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><br>0 No ADC interrupt overflow event detected.<br>1 ADC Interrupt overflow event detected.<br><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 |

#### 13.4.1.1.7 ADCINTOVFCLR Register (Offset = 6h) [reset = 0h]

ADCINTOVFCLR is shown in Figure 13-25 and described in Table 13-18.

Return to Summary Table.

ADC Interrupt Overflow Clear Register

**Figure 13-25. ADCINTOVFCLR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | ADCINT4 | ADCINT3 | ADCINT2 | ADCINT1 |
| R-0h | | | | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 13-18. ADCINTOVFCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | ADCINT4 | R=0/W=1 | 0h | ADC Interrupt 4 Overflow Clear Bits<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/W=1 | 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/W=1 | 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/W=1 | 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 |

#### 13.4.1.1.8 ADCINTSEL1N2 Register (Offset = 7h) [reset = 0h]

ADCINTSEL1N2 is shown in Figure 13-26 and described in Table 13-19.

Return to Summary Table.

ADC Interrupt 1 and 2 Selection Register

**Figure 13-26. ADCINTSEL1N2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | INT2CONT | INT2E | RESERVED | INT2SEL | | | |
| R-0h | R/W-0h | R/W-0h | R-0h | R/W-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | INT1CONT | INT1E | RESERVED | INT1SEL | | | |
| R-0h | R/W-0h | R/W-0h | R-0h | R/W-0h | | | |

**Table 13-19. ADCINTSEL1N2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14 | INT2CONT | R/W | 0h | ADCINT2 Continuous Mode Enable<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 |

**Table 13-19. ADCINTSEL1N2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | INT1CONT | R/W | 0h | ADCINT1 Continuous Mode Enable<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 |
| 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 |

#### 13.4.1.1.9 ADCINTSEL3N4 Register (Offset = 8h) [reset = 0h]

ADCINTSEL3N4 is shown in Figure 13-27 and described in Table 13-20.

Return to Summary Table.

ADC Interrupt 3 and 4 Selection Register

**Figure 13-27. ADCINTSEL3N4 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | INT4CONT | INT4E | RESERVED | INT4SEL | | | |
| R-0h | R/W-0h | R/W-0h | R-0h | R/W-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | INT3CONT | INT3E | RESERVED | INT3SEL | | | |
| R-0h | R/W-0h | R/W-0h | R-0h | R/W-0h | | | |

**Table 13-20. ADCINTSEL3N4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14 | INT4CONT | R/W | 0h | ADCINT4 Continuous Mode Enable<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 |

**Table 13-20. ADCINTSEL3N4 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | INT3CONT | R/W | 0h | ADCINT3 Continuous Mode Enable<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 |
| 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 |

#### 13.4.1.1.10 ADCSOCPRICTL Register (Offset = 9h) [reset = 200h]

ADCSOCPRICTL is shown in Figure 13-28 and described in Table 13-21.

Return to Summary Table.

ADC SOC Priority Control Register

**Figure 13-28. 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 13-21. ADCSOCPRICTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-10 | RESERVED | R | 0h | Reserved |

**Table 13-21. ADCSOCPRICTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 device is reset, when the ADCCTL1.RESET bit is set, 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 |

**Table 13-21. 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 |

#### 13.4.1.1.11 ADCINTSOCSEL1 Register (Offset = Ah) [reset = 0h]

ADCINTSOCSEL1 is shown in Figure 13-29 and described in Table 13-22.

Return to Summary Table.

ADC Interrupt SOC Selection 1 Register

**Figure 13-29. ADCINTSOCSEL1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SOC7 | | SOC6 | | SOC5 | | SOC4 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SOC3 | | SOC2 | | SOC1 | | SOC0 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 13-22. 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 |

**Table 13-22. 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 |

#### 13.4.1.1.12 ADCINTSOCSEL2 Register (Offset = Bh) [reset = 0h]

ADCINTSOCSEL2 is shown in Figure 13-30 and described in Table 13-23.

Return to Summary Table.

ADC Interrupt SOC Selection 2 Register

**Figure 13-30. ADCINTSOCSEL2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SOC15 | | SOC14 | | SOC13 | | SOC12 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SOC11 | | SOC10 | | SOC9 | | SOC8 | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 13-23. 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 |

**Table 13-23. 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 |

#### 13.4.1.1.13 ADCSOCFLG1 Register (Offset = Ch) [reset = 0h]

ADCSOCFLG1 is shown in Figure 13-31 and described in Table 13-24.

Return to Summary Table.

ADC SOC Flag 1 Register

**Figure 13-31. ADCSOCFLG1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SOC15 | SOC14 | SOC13 | SOC12 | SOC11 | SOC10 | SOC9 | SOC8 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SOC7 | SOC6 | SOC5 | SOC4 | SOC3 | SOC2 | SOC1 | SOC0 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 13-24. 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.<br>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.<br>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.<br>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 |

**Table 13-24. 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.<br>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.<br>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.<br>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.<br>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 |

**Table 13-24. ADCSOCFLG1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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.<br>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 |
| 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 |

**Table 13-24. ADCSOCFLG1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 13-24. ADCSOCFLG1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |

#### 13.4.1.1.14 ADCSOCFRC1 Register (Offset = Dh) [reset = 0h]

ADCSOCFRC1 is shown in Figure 13-32 and described in Table 13-25.

Return to Summary Table.

ADC SOC Force 1 Register

**Figure 13-32. ADCSOCFRC1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SOC15 | SOC14 | SOC13 | SOC12 | SOC11 | SOC10 | SOC9 | SOC8 |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SOC7 | SOC6 | SOC5 | SOC4 | SOC3 | SOC2 | SOC1 | SOC0 |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 13-25. ADCSOCFRC1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | SOC15 | R=0/W=1 | 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/W=1 | 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 |

**Table 13-25. ADCSOCFRC1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 13 | SOC13 | R=0/W=1 | 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 |
| 12 | SOC12 | R=0/W=1 | 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/W=1 | 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/W=1 | 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 |

**Table 13-25. ADCSOCFRC1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 9 | SOC9 | R=0/W=1 | 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 |
| 8 | SOC8 | R=0/W=1 | 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/W=1 | 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/W=1 | 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 |

**Table 13-25. ADCSOCFRC1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | SOC5 | R=0/W=1 | 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 |
| 4 | SOC4 | R=0/W=1 | 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/W=1 | 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/W=1 | 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 |

**Table 13-25. ADCSOCFRC1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | SOC1 | R=0/W=1 | 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 |
| 0 | SOC0 | R=0/W=1 | 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 |

#### 13.4.1.1.15 ADCSOCOVF1 Register (Offset = Eh) [reset = 0h]

ADCSOCOVF1 is shown in Figure 13-33 and described in Table 13-26.

Return to Summary Table.

ADC SOC Overflow 1 Register

**Figure 13-33. ADCSOCOVF1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SOC15 | SOC14 | SOC13 | SOC12 | SOC11 | SOC10 | SOC9 | SOC8 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SOC7 | SOC6 | SOC5 | SOC4 | SOC3 | SOC2 | SOC1 | SOC0 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 13-26. 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 |

**Table 13-26. ADCSOCOVF1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 13-26. ADCSOCOVF1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 13-26. ADCSOCOVF1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |

#### 13.4.1.1.16 ADCSOCOVFCLR1 Register (Offset = Fh) [reset = 0h]

ADCSOCOVFCLR1 is shown in Figure 13-34 and described in Table 13-27.

Return to Summary Table.

ADC SOC Overflow Clear 1 Register

**Figure 13-34. ADCSOCOVFCLR1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SOC15 | SOC14 | SOC13 | SOC12 | SOC11 | SOC10 | SOC9 | SOC8 |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SOC7 | SOC6 | SOC5 | SOC4 | SOC3 | SOC2 | SOC1 | SOC0 |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 13-27. ADCSOCOVFCLR1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | SOC15 | R=0/W=1 | 0h | SOC15 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC15 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.<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/W=1 | 0h | SOC14 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC14 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.<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/W=1 | 0h | SOC13 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC13 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.<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/W=1 | 0h | SOC12 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC12 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.<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 |

**Table 13-27. ADCSOCOVFCLR1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | SOC11 | R=0/W=1 | 0h | SOC11 Clear Start of Conversion Overflow Bit. Writing a 1 will clear the SOC11 overflow flag in the ADCSOCOVF1 register. Writes of 0 are ignored. Reads will always return a 0.<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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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 |

**Table 13-27. ADCSOCOVFCLR1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | SOC6 | R=0/W=1 | 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 |
| 5 | SOC5 | R=0/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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 |

**Table 13-27. ADCSOCOVFCLR1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | SOC1 | R=0/W=1 | 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/W=1 | 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 |

#### 13.4.1.1.17 ADCSOC0CTL Register (Offset = 10h) [reset = 0h]

ADCSOC0CTL is shown in Figure 13-35 and described in Table 13-28.

Return to Summary Table.

ADC SOC0 Control Register

**Figure 13-35. ADCSOC0CTL Register**

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

**Table 13-28. ADCSOC0CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |
| 24-20 | TRIGSEL | R/W | 0h | SOC0 Trigger Source Select. Along with the SOC0 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC0 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<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 - 1Fh - Reserved<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |

**Table 13-28. 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>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>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 |

#### 13.4.1.1.18 ADCSOC1CTL Register (Offset = 12h) [reset = 0h]

ADCSOC1CTL is shown in Figure 13-36 and described in Table 13-29.

Return to Summary Table.

ADC SOC1 Control Register

**Figure 13-36. ADCSOC1CTL Register**

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

**Table 13-29. ADCSOC1CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |
| 24-20 | TRIGSEL | R/W | 0h | SOC1 Trigger Source Select. Along with the SOC1 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC1 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<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 - 1Fh - Reserved<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |

**Table 13-29. 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>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>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 |

#### 13.4.1.1.19 ADCSOC2CTL Register (Offset = 14h) [reset = 0h]

ADCSOC2CTL is shown in Figure 13-37 and described in Table 13-30.

Return to Summary Table.

ADC SOC2 Control Register

**Figure 13-37. ADCSOC2CTL Register**

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

**Table 13-30. ADCSOC2CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |
| 24-20 | TRIGSEL | R/W | 0h | SOC2 Trigger Source Select. Along with the SOC2 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC2 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<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 - 1Fh - Reserved<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |

**Table 13-30. 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>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>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 |

#### 13.4.1.1.20 ADCSOC3CTL Register (Offset = 16h) [reset = 0h]

ADCSOC3CTL is shown in Figure 13-38 and described in Table 13-31.

Return to Summary Table.

ADC SOC3 Control Register

**Figure 13-38. ADCSOC3CTL Register**

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

**Table 13-31. ADCSOC3CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |
| 24-20 | TRIGSEL | R/W | 0h | SOC3 Trigger Source Select. Along with the SOC3 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC3 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<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 - 1Fh - Reserved<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |

**Table 13-31. 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>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>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 |

#### 13.4.1.1.21 ADCSOC4CTL Register (Offset = 18h) [reset = 0h]

ADCSOC4CTL is shown in Figure 13-39 and described in Table 13-32.

Return to Summary Table.

ADC SOC4 Control Register

**Figure 13-39. ADCSOC4CTL Register**

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

**Table 13-32. ADCSOC4CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |
| 24-20 | TRIGSEL | R/W | 0h | SOC4 Trigger Source Select. Along with the SOC4 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC4 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<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 - 1Fh - Reserved<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |

**Table 13-32. 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>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>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 |

#### 13.4.1.1.22 ADCSOC5CTL Register (Offset = 1Ah) [reset = 0h]

ADCSOC5CTL is shown in Figure 13-40 and described in Table 13-33.

Return to Summary Table.

ADC SOC5 Control Register

**Figure 13-40. ADCSOC5CTL Register**

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

**Table 13-33. ADCSOC5CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |
| 24-20 | TRIGSEL | R/W | 0h | SOC5 Trigger Source Select. Along with the SOC5 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC5 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<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 - 1Fh - Reserved<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |

**Table 13-33. 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>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>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 |

#### 13.4.1.1.23 ADCSOC6CTL Register (Offset = 1Ch) [reset = 0h]

ADCSOC6CTL is shown in Figure 13-41 and described in Table 13-34.

Return to Summary Table.

ADC SOC6 Control Register

**Figure 13-41. ADCSOC6CTL Register**

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

**Table 13-34. ADCSOC6CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |
| 24-20 | TRIGSEL | R/W | 0h | SOC6 Trigger Source Select. Along with the SOC6 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC6 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<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 - 1Fh - Reserved<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |

**Table 13-34. 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>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>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 |

#### 13.4.1.1.24 ADCSOC7CTL Register (Offset = 1Eh) [reset = 0h]

ADCSOC7CTL is shown in Figure 13-42 and described in Table 13-35.

Return to Summary Table.

ADC SOC7 Control Register

**Figure 13-42. ADCSOC7CTL Register**

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

**Table 13-35. ADCSOC7CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |
| 24-20 | TRIGSEL | R/W | 0h | SOC7 Trigger Source Select. Along with the SOC7 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC7 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<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 - 1Fh - Reserved<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |

**Table 13-35. 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>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>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 |

#### 13.4.1.1.25 ADCSOC8CTL Register (Offset = 20h) [reset = 0h]

ADCSOC8CTL is shown in Figure 13-43 and described in Table 13-36.

Return to Summary Table.

ADC SOC8 Control Register

**Figure 13-43. ADCSOC8CTL Register**

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

**Table 13-36. ADCSOC8CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |
| 24-20 | TRIGSEL | R/W | 0h | SOC8 Trigger Source Select. Along with the SOC8 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC8 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<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 - 1Fh - Reserved<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |

**Table 13-36. 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>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>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 |

#### 13.4.1.1.26 ADCSOC9CTL Register (Offset = 22h) [reset = 0h]

ADCSOC9CTL is shown in Figure 13-44 and described in Table 13-37.

Return to Summary Table.

ADC SOC9 Control Register

**Figure 13-44. ADCSOC9CTL Register**

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

**Table 13-37. ADCSOC9CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |
| 24-20 | TRIGSEL | R/W | 0h | SOC9 Trigger Source Select. Along with the SOC9 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC9 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<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 - 1Fh - Reserved<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |

**Table 13-37. 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>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>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 |

#### 13.4.1.1.27 ADCSOC10CTL Register (Offset = 24h) [reset = 0h]

ADCSOC10CTL is shown in Figure 13-45 and described in Table 13-38.

Return to Summary Table.

ADC SOC10 Control Register

**Figure 13-45. ADCSOC10CTL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | TRIGSEL |
| R-0h | | | | | | | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TRIGSEL | | | | RESERVED | CHSEL | | |
| R/W-0h | | | | R-0h | R/W-0h | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CHSEL | RESERVED | | | | | | ACQPS |
| R/W-0h | R-0h | | | | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ACQPS | | | | | | | |
| R/W-0h | | | | | | | |

**Table 13-38. ADCSOC10CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |
| 24-20 | TRIGSEL | R/W | 0h | SOC10 Trigger Source Select. Along with the SOC10 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC10 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<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 - 1Fh - Reserved<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |

**Table 13-38. 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>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>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 |

#### 13.4.1.1.28 ADCSOC11CTL Register (Offset = 26h) [reset = 0h]

ADCSOC11CTL is shown in Figure 13-46 and described in Table 13-39.

Return to Summary Table.

ADC SOC11 Control Register

**Figure 13-46. ADCSOC11CTL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | TRIGSEL |
| R-0h | | | | | | | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TRIGSEL | | | | RESERVED | CHSEL | | |
| R/W-0h | | | | R-0h | R/W-0h | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CHSEL | RESERVED | | | | | | ACQPS |
| R/W-0h | R-0h | | | | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ACQPS | | | | | | | |
| R/W-0h | | | | | | | |

**Table 13-39. ADCSOC11CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |
| 24-20 | TRIGSEL | R/W | 0h | SOC11 Trigger Source Select. Along with the SOC11 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC11 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<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 - 1Fh - Reserved<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |

**Table 13-39. 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>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>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 |

#### 13.4.1.1.29 ADCSOC12CTL Register (Offset = 28h) [reset = 0h]

ADCSOC12CTL is shown in Figure 13-47 and described in Table 13-40.

Return to Summary Table.

ADC SOC12 Control Register

**Figure 13-47. ADCSOC12CTL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | TRIGSEL |
| R-0h | | | | | | | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TRIGSEL | | | | RESERVED | CHSEL | | |
| R/W-0h | | | | R-0h | R/W-0h | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CHSEL | RESERVED | | | | | | ACQPS |
| R/W-0h | R-0h | | | | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ACQPS | | | | | | | |
| R/W-0h | | | | | | | |

**Table 13-40. ADCSOC12CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |
| 24-20 | TRIGSEL | R/W | 0h | SOC12 Trigger Source Select. Along with the SOC12 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC12 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<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 - 1Fh - Reserved<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |

**Table 13-40. 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>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>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 |

#### 13.4.1.1.30 ADCSOC13CTL Register (Offset = 2Ah) [reset = 0h]

ADCSOC13CTL is shown in Figure 13-48 and described in Table 13-41.

Return to Summary Table.

ADC SOC13 Control Register

**Figure 13-48. ADCSOC13CTL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | TRIGSEL |
| R-0h | | | | | | | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TRIGSEL | | | | RESERVED | CHSEL | | |
| R/W-0h | | | | R-0h | R/W-0h | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CHSEL | RESERVED | | | | | | ACQPS |
| R/W-0h | R-0h | | | | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ACQPS | | | | | | | |
| R/W-0h | | | | | | | |

**Table 13-41. ADCSOC13CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |
| 24-20 | TRIGSEL | R/W | 0h | SOC13 Trigger Source Select. Along with the SOC13 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC13 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<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 - 1Fh - Reserved<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |

**Table 13-41. 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>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>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 |

#### 13.4.1.1.31 ADCSOC14CTL Register (Offset = 2Ch) [reset = 0h]

ADCSOC14CTL is shown in Figure 13-49 and described in Table 13-42.

Return to Summary Table.

ADC SOC14 Control Register

**Figure 13-49. ADCSOC14CTL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | TRIGSEL |
| R-0h | | | | | | | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TRIGSEL | | | | RESERVED | CHSEL | | |
| R/W-0h | | | | R-0h | R/W-0h | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CHSEL | RESERVED | | | | | | ACQPS |
| R/W-0h | R-0h | | | | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ACQPS | | | | | | | |
| R/W-0h | | | | | | | |

**Table 13-42. ADCSOC14CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |
| 24-20 | TRIGSEL | R/W | 0h | SOC14 Trigger Source Select. Along with the SOC14 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC14 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<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 - 1Fh - Reserved<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |

**Table 13-42. 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>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>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 |

#### 13.4.1.1.32 ADCSOC15CTL Register (Offset = 2Eh) [reset = 0h]

ADCSOC15CTL is shown in Figure 13-50 and described in Table 13-43.

Return to Summary Table.

ADC SOC15 Control Register

**Figure 13-50. ADCSOC15CTL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | TRIGSEL |
| R-0h | | | | | | | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TRIGSEL | | | | RESERVED | CHSEL | | |
| R/W-0h | | | | R-0h | R/W-0h | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CHSEL | RESERVED | | | | | | ACQPS |
| R/W-0h | R-0h | | | | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ACQPS | | | | | | | |
| R/W-0h | | | | | | | |

**Table 13-43. ADCSOC15CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | RESERVED | R | 0h | Reserved |
| 24-20 | TRIGSEL | R/W | 0h | SOC15 Trigger Source Select. Along with the SOC15 field in the ADCINTSOCSEL1 register, this bit field configures which trigger will set the SOC15 flag in the ADCSOCFLG1 register to initiate a conversion to start once priority is given to it.<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 - 1Fh - Reserved<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |

**Table 13-43. 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>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>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 |

#### 13.4.1.1.33 ADCEVTSTAT Register (Offset = 30h) [reset = 0h]

ADCEVTSTAT is shown in Figure 13-51 and described in Table 13-44.

Return to Summary Table.

ADC Event Status Register

**Figure 13-51. 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 13-44. 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>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>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>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>Reset type: SYSRSn |
| 9 | PPB3TRIPLO | R | 0h | Post Processing Block 3 Trip Low Flag. When set indicates a digital compare trip low event has occurred.<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>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>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>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>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>Reset type: SYSRSn |

**Table 13-44. 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>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>Reset type: SYSRSn |

#### 13.4.1.1.34 ADCEVTCLR Register (Offset = 32h) [reset = 0h]

ADCEVTCLR is shown in Figure 13-52 and described in Table 13-45.

Return to Summary Table.

ADC Event Clear Register

**Figure 13-52. ADCEVTCLR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | PPB4ZERO | PPB4TRIPLO | PPB4TRIPHI | RESERVED | PPB3ZERO | PPB3TRIPLO | PPB3TRIPHI |
| R-0h | R/W-0h | R/W-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | PPB2ZERO | PPB2TRIPLO | PPB2TRIPHI | RESERVED | PPB1ZERO | PPB1TRIPLO | PPB1TRIPHI |
| R-0h | R/W-0h | R/W-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 13-45. ADCEVTCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14 | PPB4ZERO | R/W | 0h | Post Processing Block 4 Zero Crossing Clear. Clears the corresponding zero crossing flag in the ADCEVTSTAT register.<br>Reset type: SYSRSn |
| 13 | PPB4TRIPLO | R/W | 0h | Post Processing Block 4 Trip Low Clear. Clears the corresponding trip low flag in the ADCEVTSTAT register.<br>Reset type: SYSRSn |
| 12 | PPB4TRIPHI | R/W | 0h | Post Processing Block 4 Trip High Clear. Clears the corresponding trip high flag in the ADCEVTSTAT register.<br>Reset type: SYSRSn |
| 11 | RESERVED | R | 0h | Reserved |
| 10 | PPB3ZERO | R/W | 0h | Post Processing Block 3 Zero Crossing Clear. Clears the corresponding zero crossing flag in the ADCEVTSTAT register.<br>Reset type: SYSRSn |
| 9 | PPB3TRIPLO | R/W | 0h | Post Processing Block 3 Trip Low Clear. Clears the corresponding trip low flag in the ADCEVTSTAT register.<br>Reset type: SYSRSn |
| 8 | PPB3TRIPHI | R/W | 0h | Post Processing Block 3 Trip High Clear. Clears the corresponding trip high flag in the ADCEVTSTAT register.<br>Reset type: SYSRSn |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | PPB2ZERO | R/W | 0h | Post Processing Block 2 Zero Crossing Clear. Clears the corresponding zero crossing flag in the ADCEVTSTAT register.<br>Reset type: SYSRSn |
| 5 | PPB2TRIPLO | R/W | 0h | Post Processing Block 2 Trip Low Clear. Clears the corresponding trip low flag in the ADCEVTSTAT register.<br>Reset type: SYSRSn |
| 4 | PPB2TRIPHI | R/W | 0h | Post Processing Block 2 Trip High Clear. Clears the corresponding trip high flag in the ADCEVTSTAT register.<br>Reset type: SYSRSn |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | PPB1ZERO | R/W | 0h | Post Processing Block 1 Zero Crossing Clear. Clears the corresponding zero crossing flag in the ADCEVTSTAT register.<br>Reset type: SYSRSn |
| 1 | PPB1TRIPLO | R/W | 0h | Post Processing Block 1 Trip Low Clear. Clears the corresponding trip low flag in the ADCEVTSTAT register.<br>Reset type: SYSRSn |

**Table 13-45. ADCEVTCLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | PPB1TRIPHI | R/W | 0h | Post Processing Block 1 Trip High Clear. Clears the corresponding trip high flag in the ADCEVTSTAT register.<br>Reset type: SYSRSn |

#### 13.4.1.1.35 ADCEVTSEL Register (Offset = 34h) [reset = 0h]

ADCEVTSEL is shown in Figure 13-53 and described in Table 13-46.

Return to Summary Table.

ADC Event Selection Register

**Figure 13-53. ADCEVTSEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | PPB4ZERO | PPB4TRIPLO | PPB4TRIPHI | RESERVED | PPB3ZERO | PPB3TRIPLO | PPB3TRIPHI |
| R-0h | R/W-0h | R/W-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | PPB2ZERO | PPB2TRIPLO | PPB2TRIPHI | RESERVED | PPB1ZERO | PPB1TRIPLO | PPB1TRIPHI |
| R-0h | R/W-0h | R/W-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 13-46. 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 |

**Table 13-46. 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 |

#### 13.4.1.1.36 ADCEVTINTSEL Register (Offset = 36h) [reset = 0h]

ADCEVTINTSEL is shown in Figure 13-54 and described in Table 13-47.

Return to Summary Table.

ADC Event Interrupt Selection Register

**Figure 13-54. 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 13-47. 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 |

**Table 13-47. 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 |

#### 13.4.1.1.37 ADCOSDETECT Register (Offset = 38h) [reset = 0h]

ADCOSDETECT is shown in Figure 13-55 and described in Table 13-48.

Return to Summary Table.

ADC Open and Shorts Detect Register

**Figure 13-55. ADCOSDETECT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | DETECTCFG | | |
| R-0h | | | | | R/W-0h | | |

**Table 13-48. 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 |

#### 13.4.1.1.38 ADCCOUNTER Register (Offset = 39h) [reset = 0h]

ADCCOUNTER is shown in Figure 13-56 and described in Table 13-49.

Return to Summary Table.

ADC Counter Register

**Figure 13-56. ADCCOUNTER Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | FREECOUNT | | | |
| R-0h | | | | R-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FREECOUNT | | | | | | | |
| R-0h | | | | | | | |

**Table 13-49. ADCCOUNTER Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-0 | FREECOUNT | R | 0h | ADC Free Running Counter Value. This bit field reflects the status of the free running ADC counter.<br>Reset type: SYSRSn |

#### 13.4.1.1.39 ADCREV Register (Offset = 3Ah) [reset = 5h]

ADCREV is shown in Figure 13-57 and described in Table 13-50.

Return to Summary Table.

ADC Revision Register

**Figure 13-57. ADCREV Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| REV | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TYPE | | | | | | | |
| R-5h | | | | | | | |

**Table 13-50. 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 | 5h | ADC Type. Always set to 5 for this ADC.<br>Reset type: SYSRSn |

#### 13.4.1.1.40 ADCOFFTRIM Register (Offset = 3Bh) [reset = 0h]

ADCOFFTRIM is shown in Figure 13-58 and described in Table 13-51.

Return to Summary Table.

ADC Offset Trim Register

**Figure 13-58. ADCOFFTRIM Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OFFTRIM | | | | | | | |
| R/W-0h | | | | | | | |

**Table 13-51. ADCOFFTRIM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-0 | OFFTRIM | R/W | 0h | ADC Offset Trim<br>Adjusts the conversion results of the converter up<br>or down to account for offset error in the ADC. A factory trim setting will be loaded during device boot.<br>Offset can be corrected in the range of +7 to -8 LSBs. Value is 16*Offset in 8-bit 2's complement:<br>7 LSB (16*7) = 112<br>6 LSB (16*6) = 96<br>5 LSB (16*5) = 80<br>4 LSB (16*4) = 64<br>3 LSB (16*3) = 48<br>2 LSB (16*2) = 32<br>1 LSB (16*1) = 16<br>0 LSB (16*0) = 0<br>-1 LSB (16*(-1)) = 240<br>:<br>:<br>-7LSB(16*(-7)) = 144<br>Reset type: SYSRSn |

#### 13.4.1.1.41 ADCPPB1CONFIG Register (Offset = 40h) [reset = 0h]

ADCPPB1CONFIG is shown in Figure 13-59 and described in Table 13-52.

Return to Summary Table.

ADC PPB1 Config Register

**Figure 13-59. ADCPPB1CONFIG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | CBCEN | TWOSCOMPEN | CONFIG | | | |
| R-0h | | R/W-0h | R/W-0h | R/W-0h | | | |

**Table 13-52. ADCPPB1CONFIG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-6 | RESERVED | R | 0h | Reserved |
| 5 | CBCEN | R/W | 0h | ADC Post Processing Block Cycle By Cycle Enable. When set, this bit enables the post conversion hardware processing circuit to automatically clear the ADCEVTSTAT on a conversion if the event condition is no longer present.<br>Reset type: SYSRSn |
| 4 | TWOSCOMPEN | R/W | 0h | ADC Post Processing Block 1 Two's Complement Enable. When set this bit enables the post conversion hardware processing circuit that performs a two's complement on the output of the offset/reference subtraction unit before storing the result in the ADCPPB1RESULT register.<br>0 ADCPPB1RESULT = ADCRESULTx - ADCPPB1OFFREF<br>1 ADCPPB1RESULT = ADCPPB1OFFREF - ADCRESULTx<br>Reset type: SYSRSn |

**Table 13-52. 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 |

#### 13.4.1.1.42 ADCPPB1STAMP Register (Offset = 41h) [reset = 0h]

ADCPPB1STAMP is shown in Figure 13-60 and described in Table 13-53.

Return to Summary Table.

ADC PPB1 Sample Delay Time Stamp Register

**Figure 13-60. ADCPPB1STAMP Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | DLYSTAMP | | | |
| R-0h | | | | R-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DLYSTAMP | | | | | | | |
| R-0h | | | | | | | |

**Table 13-53. ADCPPB1STAMP Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-0 | DLYSTAMP | R | 0h | ADC Post Processing Block 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 |

#### 13.4.1.1.43 ADCPPB1OFFCAL Register (Offset = 42h) [reset = 0h]

ADCPPB1OFFCAL is shown in Figure 13-61 and described in Table 13-54.

Return to Summary Table.

ADC PPB1 Offset Calibration Register

**Figure 13-61. ADCPPB1OFFCAL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | OFFCAL | |
| R-0h | | | | | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OFFCAL | | | | | | | |
| R/W-0h | | | | | | | |

**Table 13-54. 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>Reset type: SYSRSn |

#### 13.4.1.1.44 ADCPPB1OFFREF Register (Offset = 43h) [reset = 0h]

ADCPPB1OFFREF is shown in Figure 13-62 and described in Table 13-55.

Return to Summary Table.

ADC PPB1 Offset Reference Register

**Figure 13-62. ADCPPB1OFFREF Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OFFREF | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 13-55. 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 |

#### 13.4.1.1.45 ADCPPB1TRIPHI Register (Offset = 44h) [reset = 0h]

ADCPPB1TRIPHI is shown in Figure 13-63 and described in Table 13-56.

Return to Summary Table.

ADC PPB1 Trip High Register

**Figure 13-63. ADCPPB1TRIPHI Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | HSIGN |
| R-0h | | | | | | | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LIMITHI | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LIMITHI | | | | | | | |
| R/W-0h | | | | | | | |

**Table 13-56. ADCPPB1TRIPHI Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-17 | RESERVED | R | 0h | Reserved |
| 16 | HSIGN | R/W | 0h | High Limit Sign Bit. This is the sign bit (17th bit) to the LIMITHI bit field when in 16-bit ADC mode.<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 |

#### 13.4.1.1.46 ADCPPB1TRIPLO Register (Offset = 46h) [reset = 0h]

ADCPPB1TRIPLO is shown in Figure 13-64 and described in Table 13-57.

Return to Summary Table.

ADC PPB1 Trip Low/Trigger Time Stamp Register

**Figure 13-64. ADCPPB1TRIPLO Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| REQSTAMP | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| REQSTAMP | | | | RESERVED | | | LSIGN |
| R-0h | | | | R-0h | | | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LIMITLO | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LIMITLO | | | | | | | |
| R/W-0h | | | | | | | |

**Table 13-57. 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 |

#### 13.4.1.1.47 ADCPPB2CONFIG Register (Offset = 48h) [reset = 0h]

ADCPPB2CONFIG is shown in Figure 13-65 and described in Table 13-58.

Return to Summary Table.

ADC PPB2 Config Register

**Figure 13-65. ADCPPB2CONFIG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | CBCEN | TWOSCOMPEN | CONFIG | | | |
| R-0h | | R/W-0h | R/W-0h | R/W-0h | | | |

**Table 13-58. ADCPPB2CONFIG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-6 | RESERVED | R | 0h | Reserved |
| 5 | CBCEN | R/W | 0h | ADC Post Processing Block Cycle By Cycle Enable. When set, this bit enables the post conversion hardware processing circuit to automatically clear the ADCEVTSTAT on a conversion if the event condition is no longer present.<br>Reset type: SYSRSn |
| 4 | TWOSCOMPEN | R/W | 0h | ADC Post Processing Block 2 Two's Complement Enable. When set this bit enables the post conversion hardware processing circuit that performs a two's complement on the output of the offset/reference subtraction unit before storing the result in the ADCPPB2RESULT register.<br>0 ADCPPB2RESULT = ADCRESULTx - ADCPPB2OFFREF<br>1 ADCPPB2RESULT = ADCPPB2OFFREF - ADCRESULTx<br>Reset type: SYSRSn |

**Table 13-58. 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 |

#### 13.4.1.1.48 ADCPPB2STAMP Register (Offset = 49h) [reset = 0h]

ADCPPB2STAMP is shown in Figure 13-66 and described in Table 13-59.

Return to Summary Table.

ADC PPB2 Sample Delay Time Stamp Register

**Figure 13-66. ADCPPB2STAMP Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | DLYSTAMP | | | |
| R-0h | | | | R-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DLYSTAMP | | | | | | | |
| R-0h | | | | | | | |

**Table 13-59. ADCPPB2STAMP Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-0 | DLYSTAMP | R | 0h | ADC Post Processing Block 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 |

#### 13.4.1.1.49 ADCPPB2OFFCAL Register (Offset = 4Ah) [reset = 0h]

ADCPPB2OFFCAL is shown in Figure 13-67 and described in Table 13-60.

Return to Summary Table.

ADC PPB2 Offset Calibration Register

**Figure 13-67. ADCPPB2OFFCAL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | OFFCAL | |
| R-0h | | | | | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OFFCAL | | | | | | | |
| R/W-0h | | | | | | | |

**Table 13-60. 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>Reset type: SYSRSn |

#### 13.4.1.1.50 ADCPPB2OFFREF Register (Offset = 4Bh) [reset = 0h]

ADCPPB2OFFREF is shown in Figure 13-68 and described in Table 13-61.

Return to Summary Table.

ADC PPB2 Offset Reference Register

**Figure 13-68. ADCPPB2OFFREF Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OFFREF | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 13-61. 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 |

#### 13.4.1.1.51 ADCPPB2TRIPHI Register (Offset = 4Ch) [reset = 0h]

ADCPPB2TRIPHI is shown in Figure 13-69 and described in Table 13-62.

Return to Summary Table.

ADC PPB2 Trip High Register

**Figure 13-69. ADCPPB2TRIPHI Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | HSIGN |
| R-0h | | | | | | | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LIMITHI | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LIMITHI | | | | | | | |
| R/W-0h | | | | | | | |

**Table 13-62. ADCPPB2TRIPHI Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-17 | RESERVED | R | 0h | Reserved |
| 16 | HSIGN | R/W | 0h | High Limit Sign Bit. This is the sign bit (17th bit) to the LIMITHI bit field when in 16-bit ADC mode.<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 |

#### 13.4.1.1.52 ADCPPB2TRIPLO Register (Offset = 4Eh) [reset = 0h]

ADCPPB2TRIPLO is shown in Figure 13-70 and described in Table 13-63.

Return to Summary Table.

ADC PPB2 Trip Low/Trigger Time Stamp Register

**Figure 13-70. ADCPPB2TRIPLO Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| REQSTAMP | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| REQSTAMP | | | | RESERVED | | | LSIGN |
| R-0h | | | | R-0h | | | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LIMITLO | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LIMITLO | | | | | | | |
| R/W-0h | | | | | | | |

**Table 13-63. 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 |

#### 13.4.1.1.53 ADCPPB3CONFIG Register (Offset = 50h) [reset = 0h]

ADCPPB3CONFIG is shown in Figure 13-71 and described in Table 13-64.

Return to Summary Table.

ADC PPB3 Config Register

**Figure 13-71. ADCPPB3CONFIG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | CBCEN | TWOSCOMPEN | CONFIG | | | |
| R-0h | | R/W-0h | R/W-0h | R/W-0h | | | |

**Table 13-64. ADCPPB3CONFIG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-6 | RESERVED | R | 0h | Reserved |
| 5 | CBCEN | R/W | 0h | ADC Post Processing Block Cycle By Cycle Enable. When set, this bit enables the post conversion hardware processing circuit to automatically clear the ADCEVTSTAT on a conversion if the event condition is no longer present.<br>Reset type: SYSRSn |
| 4 | TWOSCOMPEN | R/W | 0h | ADC Post Processing Block 3 Two's Complement Enable. When set this bit enables the post conversion hardware processing circuit that performs a two's complement on the output of the offset/reference subtraction unit before storing the result in the ADCPPB3RESULT register.<br>0 ADCPPB3RESULT = ADCRESULTx - ADCPPB3OFFREF<br>1 ADCPPB3RESULT = ADCPPB3OFFREF - ADCRESULTx<br>Reset type: SYSRSn |

**Table 13-64. 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 |

#### 13.4.1.1.54 ADCPPB3STAMP Register (Offset = 51h) [reset = 0h]

ADCPPB3STAMP is shown in Figure 13-72 and described in Table 13-65.

Return to Summary Table.

ADC PPB3 Sample Delay Time Stamp Register

**Figure 13-72. ADCPPB3STAMP Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | DLYSTAMP | | | |
| R-0h | | | | R-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DLYSTAMP | | | | | | | |
| R-0h | | | | | | | |

**Table 13-65. ADCPPB3STAMP Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-0 | DLYSTAMP | R | 0h | ADC Post Processing Block 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 |

#### 13.4.1.1.55 ADCPPB3OFFCAL Register (Offset = 52h) [reset = 0h]

ADCPPB3OFFCAL is shown in Figure 13-73 and described in Table 13-66.

Return to Summary Table.

ADC PPB3 Offset Calibration Register

**Figure 13-73. ADCPPB3OFFCAL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | OFFCAL | |
| R-0h | | | | | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OFFCAL | | | | | | | |
| R/W-0h | | | | | | | |

**Table 13-66. 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>Reset type: SYSRSn |

#### 13.4.1.1.56 ADCPPB3OFFREF Register (Offset = 53h) [reset = 0h]

ADCPPB3OFFREF is shown in Figure 13-74 and described in Table 13-67.

Return to Summary Table.

ADC PPB3 Offset Reference Register

**Figure 13-74. ADCPPB3OFFREF Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OFFREF | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 13-67. 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 |

#### 13.4.1.1.57 ADCPPB3TRIPHI Register (Offset = 54h) [reset = 0h]

ADCPPB3TRIPHI is shown in Figure 13-75 and described in Table 13-68.

Return to Summary Table.

ADC PPB3 Trip High Register

**Figure 13-75. ADCPPB3TRIPHI Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | HSIGN |
| R-0h | | | | | | | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LIMITHI | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LIMITHI | | | | | | | |
| R/W-0h | | | | | | | |

**Table 13-68. ADCPPB3TRIPHI Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-17 | RESERVED | R | 0h | Reserved |
| 16 | HSIGN | R/W | 0h | High Limit Sign Bit. This is the sign bit (17th bit) to the LIMITHI bit field when in 16-bit ADC mode.<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 |

#### 13.4.1.1.58 ADCPPB3TRIPLO Register (Offset = 56h) [reset = 0h]

ADCPPB3TRIPLO is shown in Figure 13-76 and described in Table 13-69.

Return to Summary Table.

ADC PPB3 Trip Low/Trigger Time Stamp Register

**Figure 13-76. ADCPPB3TRIPLO Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| REQSTAMP | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| REQSTAMP | | | | RESERVED | | | LSIGN |
| R-0h | | | | R-0h | | | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LIMITLO | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LIMITLO | | | | | | | |
| R/W-0h | | | | | | | |

**Table 13-69. 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 |

#### 13.4.1.1.59 ADCPPB4CONFIG Register (Offset = 58h) [reset = 0h]

ADCPPB4CONFIG is shown in Figure 13-77 and described in Table 13-70.

Return to Summary Table.

ADC PPB4 Config Register

**Figure 13-77. ADCPPB4CONFIG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | CBCEN | TWOSCOMPEN | CONFIG | | | |
| R-0h | | R/W-0h | R/W-0h | R/W-0h | | | |

**Table 13-70. ADCPPB4CONFIG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-6 | RESERVED | R | 0h | Reserved |
| 5 | CBCEN | R/W | 0h | ADC Post Processing Block Cycle By Cycle Enable. When set, this bit enables the post conversion hardware processing circuit to automatically clear the ADCEVTSTAT on a conversion if the event condition is no longer present.<br>Reset type: SYSRSn |
| 4 | TWOSCOMPEN | R/W | 0h | ADC Post Processing Block 4 Two's Complement Enable. When set this bit enables the post conversion hardware processing circuit that performs a two's complement on the output of the offset/reference subtraction unit before storing the result in the ADCPPB4RESULT register.<br>0 ADCPPB4RESULT = ADCRESULTx - ADCPPB4OFFREF<br>1 ADCPPB4RESULT = ADCPPB4OFFREF - ADCRESULTx<br>Reset type: SYSRSn |

**Table 13-70. 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 |

#### 13.4.1.1.60 ADCPPB4STAMP Register (Offset = 59h) [reset = 0h]

ADCPPB4STAMP is shown in Figure 13-78 and described in Table 13-71.

Return to Summary Table.

ADC PPB4 Sample Delay Time Stamp Register

**Figure 13-78. ADCPPB4STAMP Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | DLYSTAMP | | | |
| R-0h | | | | R-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DLYSTAMP | | | | | | | |
| R-0h | | | | | | | |

**Table 13-71. ADCPPB4STAMP Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-0 | DLYSTAMP | R | 0h | ADC Post Processing Block 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 |

#### 13.4.1.1.61 ADCPPB4OFFCAL Register (Offset = 5Ah) [reset = 0h]

ADCPPB4OFFCAL is shown in Figure 13-79 and described in Table 13-72.

Return to Summary Table.

ADC PPB4 Offset Calibration Register

**Figure 13-79. ADCPPB4OFFCAL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | OFFCAL | |
| R-0h | | | | | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OFFCAL | | | | | | | |
| R/W-0h | | | | | | | |

**Table 13-72. 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>Reset type: SYSRSn |

#### 13.4.1.1.62 ADCPPB4OFFREF Register (Offset = 5Bh) [reset = 0h]

ADCPPB4OFFREF is shown in Figure 13-80 and described in Table 13-73.

Return to Summary Table.

ADC PPB4 Offset Reference Register

**Figure 13-80. ADCPPB4OFFREF Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OFFREF | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 13-73. 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 |

#### 13.4.1.1.63 ADCPPB4TRIPHI Register (Offset = 5Ch) [reset = 0h]

ADCPPB4TRIPHI is shown in Figure 13-81 and described in Table 13-74.

Return to Summary Table.

ADC PPB4 Trip High Register

**Figure 13-81. ADCPPB4TRIPHI Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | HSIGN |
| R-0h | | | | | | | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LIMITHI | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LIMITHI | | | | | | | |
| R/W-0h | | | | | | | |

**Table 13-74. ADCPPB4TRIPHI Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-17 | RESERVED | R | 0h | Reserved |
| 16 | HSIGN | R/W | 0h | High Limit Sign Bit. This is the sign bit (17th bit) to the LIMITHI bit field when in 16-bit ADC mode.<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 |

#### 13.4.1.1.64 ADCPPB4TRIPLO Register (Offset = 5Eh) [reset = 0h]

ADCPPB4TRIPLO is shown in Figure 13-82 and described in Table 13-75.

Return to Summary Table.

ADC PPB4 Trip Low/Trigger Time Stamp Register

**Figure 13-82. ADCPPB4TRIPLO Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| REQSTAMP | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| REQSTAMP | | | | RESERVED | | | LSIGN |
| R-0h | | | | R-0h | | | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| LIMITLO | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LIMITLO | | | | | | | |
| R/W-0h | | | | | | | |

**Table 13-75. 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 |

#### 13.4.1.1.65 ADCINTCYCLE Register (Offset = 6Fh) [reset = 0h]

ADCINTCYCLE is shown in Figure 13-83 and described in Table 13-76.

Return to Summary Table.

ADC Early Interrupt Generation Cycle

**Figure 13-83. ADCINTCYCLE Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DELAY | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 13-76. ADCINTCYCLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | DELAY | R/W | 0h | ADC Early Interrupt Generation Cycle Delay: Defines the delay from the fall edge of ADCSOC in terms of system clock cycles, for the interrupt to be generated.<br>Reset type: SYSRSn |

#### 13.4.1.1.66 ADCINLTRIM1 Register (Offset = 70h) [reset = X]

ADCINLTRIM1 is shown in Figure 13-84 and described in Table 13-77.

Return to Summary Table.

ADC Linearity Trim 1 Register

**Figure 13-84. ADCINLTRIM1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INLTRIM31TO0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 13-77. ADCINLTRIM1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | INLTRIM31TO0 | R/W | X | ADC Linearity Trim Bits 31-0.<br><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 |

#### 13.4.1.1.67 ADCINLTRIM2 Register (Offset = 72h) [reset = X]

ADCINLTRIM2 is shown in Figure 13-85 and described in Table 13-78.

Return to Summary Table.

ADC Linearity Trim 2 Register

**Figure 13-85. ADCINLTRIM2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INLTRIM63TO32 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 13-78. ADCINLTRIM2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | INLTRIM63TO32 | R/W | X | 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 |

#### 13.4.1.1.68 ADCINLTRIM3 Register (Offset = 74h) [reset = X]

ADCINLTRIM3 is shown in Figure 13-86 and described in Table 13-79.

Return to Summary Table.

ADC Linearity Trim 3 Register

**Figure 13-86. ADCINLTRIM3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INLTRIM95TO64 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-X | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 13-79. ADCINLTRIM3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | INLTRIM95TO64 | R/W | X | ADC Linearity Trim Bits 95-64.<br><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 |

### 13.4.1.2 ADC_RESULT_REGS Registers

Table 13-80 lists the memory-mapped registers for the ADC_RESULT_REGS. All register offset addresses not listed in Table 13-80 should be considered as reserved locations and the register contents should not be modified.

**Table 13-80. 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 13-81 shows the codes that are used for access types in this section.

**Table 13-81. 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. |

#### 13.4.1.2.1 ADCRESULT0 Register (Offset = 0h) [reset = 0h]

ADCRESULT0 is shown in Figure 13-87 and described in Table 13-82.

Return to Summary Table.

ADC Result 0 Register

**Figure 13-87. ADCRESULT0 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESULT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 13-82. ADCRESULT0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 0<br><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 |

#### 13.4.1.2.2 ADCRESULT1 Register (Offset = 1h) [reset = 0h]

ADCRESULT1 is shown in Figure 13-88 and described in Table 13-83.

Return to Summary Table.

ADC Result 1 Register

**Figure 13-88. ADCRESULT1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESULT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 13-83. ADCRESULT1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 1<br><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 |

#### 13.4.1.2.3 ADCRESULT2 Register (Offset = 2h) [reset = 0h]

ADCRESULT2 is shown in Figure 13-89 and described in Table 13-84.

Return to Summary Table.

ADC Result 2 Register

**Figure 13-89. ADCRESULT2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESULT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 13-84. ADCRESULT2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 2<br><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 |

#### 13.4.1.2.4 ADCRESULT3 Register (Offset = 3h) [reset = 0h]

ADCRESULT3 is shown in Figure 13-90 and described in Table 13-85.

Return to Summary Table.

ADC Result 3 Register

**Figure 13-90. ADCRESULT3 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESULT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 13-85. ADCRESULT3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 3<br><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 |

#### 13.4.1.2.5 ADCRESULT4 Register (Offset = 4h) [reset = 0h]

ADCRESULT4 is shown in Figure 13-91 and described in Table 13-86.

Return to Summary Table.

ADC Result 4 Register

**Figure 13-91. ADCRESULT4 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESULT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 13-86. ADCRESULT4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 4<br><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 |

#### 13.4.1.2.6 ADCRESULT5 Register (Offset = 5h) [reset = 0h]

ADCRESULT5 is shown in Figure 13-92 and described in Table 13-87.

Return to Summary Table.

ADC Result 5 Register

**Figure 13-92. ADCRESULT5 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESULT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 13-87. ADCRESULT5 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 5<br><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 |

#### 13.4.1.2.7 ADCRESULT6 Register (Offset = 6h) [reset = 0h]

ADCRESULT6 is shown in Figure 13-93 and described in Table 13-88.

Return to Summary Table.

ADC Result 6 Register

**Figure 13-93. ADCRESULT6 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESULT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 13-88. ADCRESULT6 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 6<br><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 |

#### 13.4.1.2.8 ADCRESULT7 Register (Offset = 7h) [reset = 0h]

ADCRESULT7 is shown in Figure 13-94 and described in Table 13-89.

Return to Summary Table.

ADC Result 7 Register

**Figure 13-94. ADCRESULT7 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESULT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 13-89. ADCRESULT7 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 7<br><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 |

#### 13.4.1.2.9 ADCRESULT8 Register (Offset = 8h) [reset = 0h]

ADCRESULT8 is shown in Figure 13-95 and described in Table 13-90.

Return to Summary Table.

ADC Result 8 Register

**Figure 13-95. ADCRESULT8 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESULT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 13-90. ADCRESULT8 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 8<br><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 |

#### 13.4.1.2.10 ADCRESULT9 Register (Offset = 9h) [reset = 0h]

ADCRESULT9 is shown in Figure 13-96 and described in Table 13-91.

Return to Summary Table.

ADC Result 9 Register

**Figure 13-96. ADCRESULT9 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESULT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 13-91. ADCRESULT9 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 9<br><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 |

#### 13.4.1.2.11 ADCRESULT10 Register (Offset = Ah) [reset = 0h]

ADCRESULT10 is shown in Figure 13-97 and described in Table 13-92.

Return to Summary Table.

ADC Result 10 Register

**Figure 13-97. ADCRESULT10 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESULT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 13-92. ADCRESULT10 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 10<br><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 |

#### 13.4.1.2.12 ADCRESULT11 Register (Offset = Bh) [reset = 0h]

ADCRESULT11 is shown in Figure 13-98 and described in Table 13-93.

Return to Summary Table.

ADC Result 11 Register

**Figure 13-98. ADCRESULT11 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESULT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 13-93. ADCRESULT11 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 11<br><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 |

#### 13.4.1.2.13 ADCRESULT12 Register (Offset = Ch) [reset = 0h]

ADCRESULT12 is shown in Figure 13-99 and described in Table 13-94.

Return to Summary Table.

ADC Result 12 Register

**Figure 13-99. ADCRESULT12 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESULT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 13-94. ADCRESULT12 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 12<br><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 |

#### 13.4.1.2.14 ADCRESULT13 Register (Offset = Dh) [reset = 0h]

ADCRESULT13 is shown in Figure 13-100 and described in Table 13-95.

Return to Summary Table.

ADC Result 13 Register

**Figure 13-100. ADCRESULT13 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESULT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 13-95. ADCRESULT13 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 13<br><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 |

#### 13.4.1.2.15 ADCRESULT14 Register (Offset = Eh) [reset = 0h]

ADCRESULT14 is shown in Figure 13-101 and described in Table 13-96.

Return to Summary Table.

ADC Result 14 Register

**Figure 13-101. ADCRESULT14 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESULT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 13-96. ADCRESULT14 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 14<br><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 |

#### 13.4.1.2.16 ADCRESULT15 Register (Offset = Fh) [reset = 0h]

ADCRESULT15 is shown in Figure 13-102 and described in Table 13-97.

Return to Summary Table.

ADC Result 15 Register

**Figure 13-102. ADCRESULT15 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESULT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 13-97. ADCRESULT15 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RESULT | R | 0h | ADC Result 15<br><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 |

#### 13.4.1.2.17 ADCPPB1RESULT Register (Offset = 10h) [reset = 0h]

ADCPPB1RESULT is shown in Figure 13-103 and described in Table 13-98.

Return to Summary Table.

ADC Post Processing Block 1 Result Register

**Figure 13-103. ADCPPB1RESULT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SIGN | | | | | | | | | | | | | | | | PPBRESULT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | |

**Table 13-98. 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. If ADCINTFLG is polled in reading PPBRESULT, user needs to add a NOP instruction to ensure that post conversion processing is populated in this register.<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 |

#### 13.4.1.2.18 ADCPPB2RESULT Register (Offset = 12h) [reset = 0h]

ADCPPB2RESULT is shown in Figure 13-104 and described in Table 13-99.

Return to Summary Table.

ADC Post Processing Block 2 Result Register

**Figure 13-104. ADCPPB2RESULT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SIGN | | | | | | | | | | | | | | | | PPBRESULT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | |

**Table 13-99. 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><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><br>The result of the offset/reference subtraction post conversion processing is stored in this register. If ADCINTFLG is polled in reading PPBRESULT, user needs to add a NOP instruction to ensure that post conversion processing is populated in this register.<br><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 |

#### 13.4.1.2.19 ADCPPB3RESULT Register (Offset = 14h) [reset = 0h]

ADCPPB3RESULT is shown in Figure 13-105 and described in Table 13-100.

Return to Summary Table.

ADC Post Processing Block 3 Result Register

**Figure 13-105. ADCPPB3RESULT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SIGN | | | | | | | | | | | | | | | | PPBRESULT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | |

**Table 13-100. 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. If ADCINTFLG is polled in reading PPBRESULT, user needs to add a NOP instruction to ensure that post conversion processing is populated in this register.<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 |

#### 13.4.1.2.20 ADCPPB4RESULT Register (Offset = 16h) [reset = 0h]

ADCPPB4RESULT is shown in Figure 13-106 and described in Table 13-101.

Return to Summary Table.

ADC Post Processing Block 4 Result Register

**Figure 13-106. ADCPPB4RESULT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SIGN | | | | | | | | | | | | | | | | PPBRESULT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | |

**Table 13-101. 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><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><br>The result of the offset/reference subtraction post conversion processing is stored in this register. If ADCINTFLG is polled in reading PPBRESULT, user needs to add a NOP instruction to ensure that post conversion processing is populated in this register.<br><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 |

# Programmable Gain Amplifier (PGA)

The Programmable Gain Amplifier (PGA) is used to amplify an input voltage for the purpose of increasing the dynamic range of the downstream ADC and CMPSS modules.

**Topic** **Page**

## 14.1 Programmable Gain Amplifier (PGA) Overview

The integrated PGA helps to reduce cost and design effort for many control applications that traditionally require external, standalone amplifiers. On-chip integration ensures that the PGA is compatible with the downstream ADC and CMPSS modules. Software selectable gain and filter settings make the PGA adaptable to various performance needs.

### 14.1.1 Features

Features available to PGA modules are:

- Four programmable gain modes: 3x, 6x, 12x, 24x
- Internally powered by VDDA and VSSA
- Hardware-based trims to reduce offset and gain errors
- Support for Kelvin ground connections using PGA_GND pin
- Embedded series resistors for RC filtering

### 14.1.2 Block Diagram

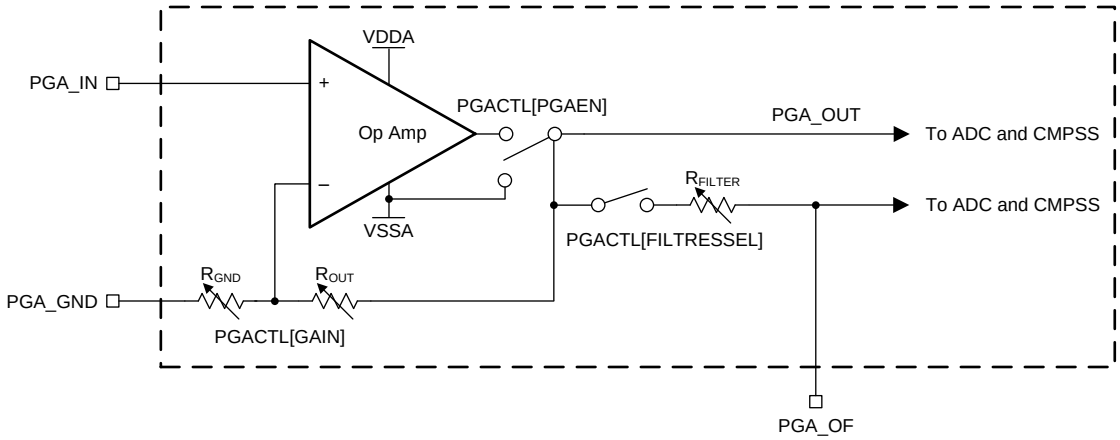
The active component in the PGA is an embedded operational amplifier (op-amp) that is configured as a non-inverting amplifier with internal feedback resistors. These internal feedback resistor values are paired to produce software selectable voltage gains.

Three PGA signals are available at the device pins:

- PGA_IN is the positive input to the PGA op-amp. The signal applied to this pin will be amplified by the PGA.
- PGA_GND is the Kelvin ground reference for the PGA_IN signal. Ideally, the PGA_GND reference is equal to VSSA, however the PGA can tolerate small voltage offsets from VSSA. See the device data manual for more information.
- PGA_OF supports op-amp output filtering with RC components. The filtered signal is available for sampling and monitoring by internal ADC and CMPSS modules.

PGA_OUT is an internal signal at the op-amp output. It is available for sampling and monitoring by the internal ADC and CMPSS modules.

**Figure 14-1. PGA Block Diagram**



## 14.2 Linear Output Range

The absolute output range of the PGA is bounded by the analog VDDA and VSSA supplies – the PGA cannot produce output voltages greater than VDDA or less than VSSA.

Although the PGA can produce full-scale output across the absolute voltage range of VSSA to VDDA, the amplifier output is only linear within a subset of the absolute range. This reduced range is referred to as the linear output range.

The PGA performance specifications in the device data manual only apply to the linear output range. For best performance, the input signal should be conditioned in such a way that the PGA stays within the linear output range during normal system operation.

**NOTE:** The voltage input range required to operate the PGA in the linear output range is unique for each gain mode. See the device data manual for the linear output range.

## 14.3 Gain Modes

Gain modes of 3x, 6x, 12x, and 24x are software-selectable via the PGACTL[GAIN] register field. The gain of the PGA is determined by a preset ratio between resistors $R_{OUT}$ and $R_{GND}$:

$$\text{Gain} = 1 + \frac{R_{OUT}}{R_{GND}}$$

The ideal target values for the gain resistors by mode are shown in Table 14-1.

**Table 14-1. Ideal Gain Resistor Values**

| Gain Mode | Ideal $R_{OUT}$ | Ideal $R_{GND}$ |
|---|---|---|
| 3x | 20kΩ | 10kΩ |
| 6x | 25kΩ | 5kΩ |
| 12x | 27.5kΩ | 2.5kΩ |
| 24x | 28.75kΩ | 1.25kΩ |

Changing the gain mode during normal operation is allowed, but a minimum configuration settling time should be observed when doing so. See the device data manual for the gain switch settling time.

## 14.4 External Filtering

The PGA output may be routed to a pin through an embedded series resistor for the purpose of low-pass filtering the amplified signal. The filter resistance is software selectable via the PGACTL[FILTRESSEL] register field. The default selection of PGACTL[FILTRESSEL]=0 will disable the filter path.

The cutoff frequency can be estimated using the standard low-pass RC equation of:

$$f_C = \frac{1}{2\pi RC}$$

Each gain mode requires a minimum amount of series resistance when filtering is enabled. The values are shown in Table 14-2.

**Table 14-2. Minimum Filter Resistance**

| Gain Mode | Minimum $R_{FILTER}$ |
|---|---|
| 3x | 50Ω |
| 6x | 50Ω |
| 12x | 80Ω |
| 24x | 100Ω |

The choice of $C_{FILTER}$ value may also influence the ADC sampling performance. See Section 14.10.1.2 for more information.

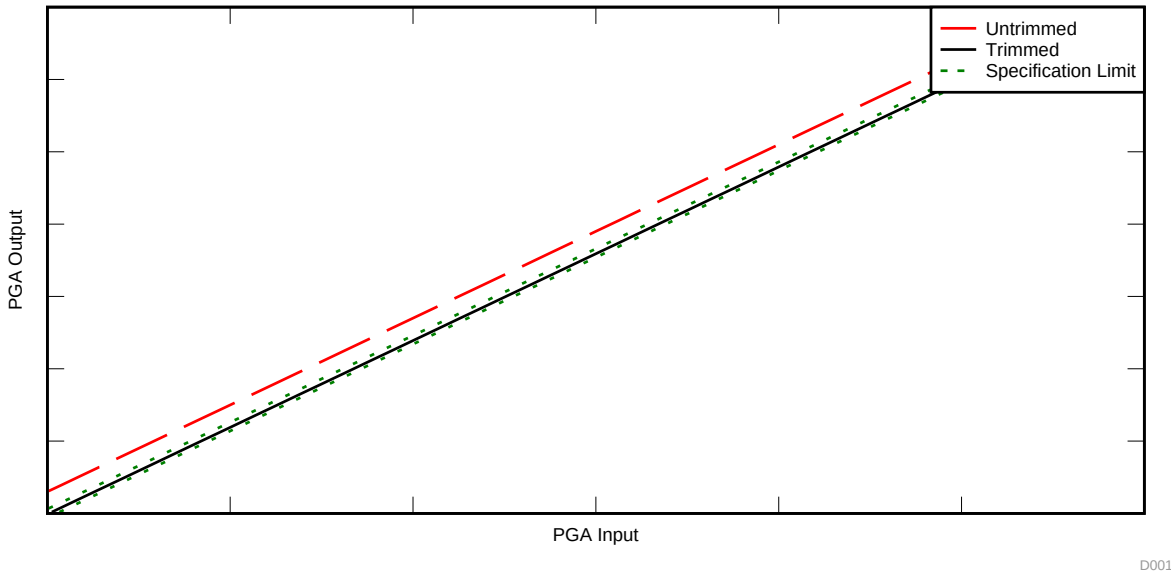
## 14.5 Error Calibration

Inherent offset and gain errors can be reduced through the use of built-in hardware trim circuitry. Factory-generated values are written to the trim registers by calling the Device_cal() function that is located in TI reserved OTP.

### *14.5.1 Offset Error*

The offset error appears as a constant DC offset across the PGA output range. The hardware trim will reduce the offset error so that it falls within data manual specifications.

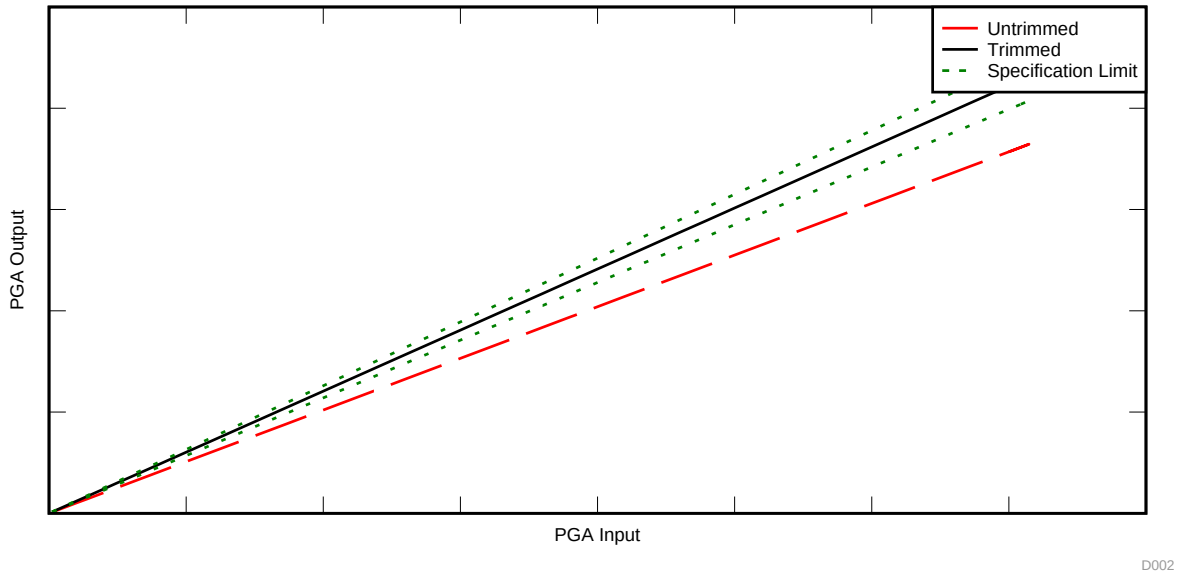
**Figure 14-2. PGA Offset Trim**




### *14.5.2 Gain Error*

After the offset error has been removed, the remaining error, gain error appears as a scaled error that increases in magnitude with increasing PGA output voltage. The hardware trim will target the gain performance so that it falls within data manual specifications.

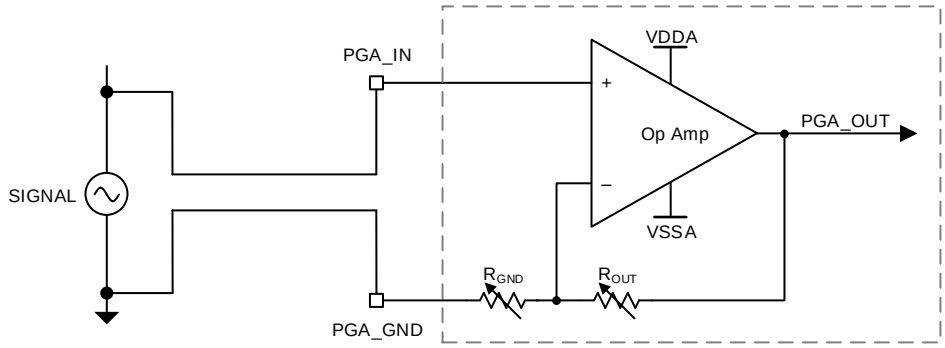
**Figure 14-3. PGA Gain Trim**




## 14.6 Ground Routing

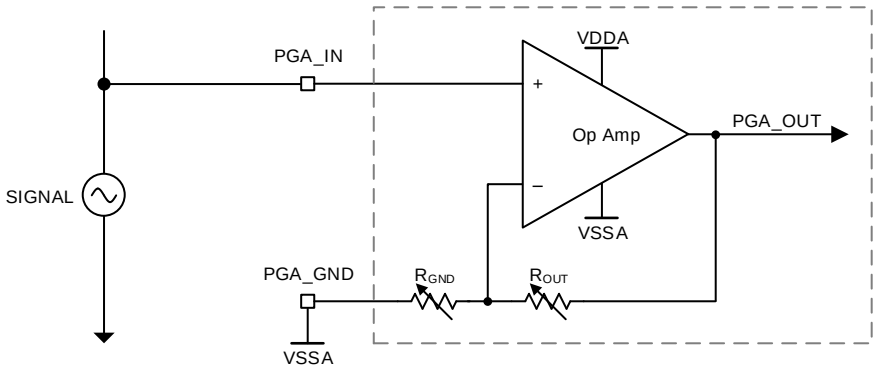
PGA_GND should be connected to the source signal ground reference when possible. Routing the PGA_IN and PGA_GND signals in close proximity (especially in parallel) may also help to reduce sensitivity to external noise.

**Figure 14-4. PGA_GND to Remote Ground**




PGA_GND may also be connected to VSSA if the source signal ground reference is not available.

**Figure 14-5. PGA_GND to VSSA**




PGA_GND should not be biased with an intentional DC offset with respect to VSSA.

## 14.7 Enabling and Disabling the PGA Clock

If the clock to the PGA is disabled while the PGA is outputting a voltage, the output voltage remains unaffected but the PGA registers will no longer be updated with register writes. Enabling the clock resumes register writes.

## 14.8 Lock Register

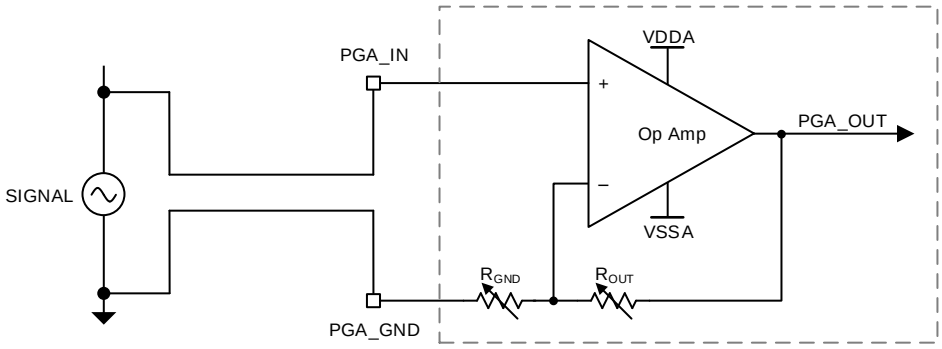
The PGALOCK register provides the ability to block writes to certain PGA configuration registers. Locking register writes can prevent spurious or malicious code from modifying critical register settings. Once a register has been locked via the PGALOCK register, only a module- or device-level reset can restore write functionality.

## 14.9 Examples

### 14.9.1 Direct Amplifier

Given a small positive signal, the PGA can be used to amplify the signal to increase the dynamic range of ADC sampling and comparator trip monitoring. For example, an input signal with a valid range between 0.25V and 0.75V can be amplified in 3x mode to produce an output signal between 0.75V and 2.25V.

**Figure 14-6. PGA Amplifier Example**



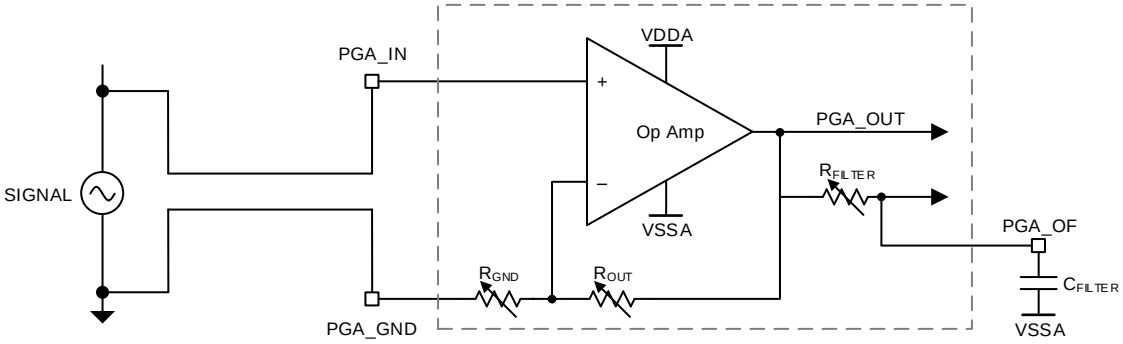

The amplified output voltage is calculated as:

$$V_{PGA\_OUT} = \left(1 + \frac{R_{OUT}}{R_{GND}}\right) \times V_{PGA\_IN}$$

### 14.9.2 RC Filter

If low-pass filtering is desired, an external capacitor can be added to the PGA_OF pin in the following topology.

**Figure 14-7. PGA Filter Example**




The amplified voltage is calculated as:

$$V_{PGA\_OUT} = \left(1 + \frac{R_{OUT}}{R_{GND}}\right) \times V_{PGA\_IN}$$

The filter cutoff frequency is estimated as:

$$f_{CUTOFF} = \frac{1}{2\pi R_{FILTER} C_{FILTER}}$$

## 14.10 Analog Front End Integration

The PGAs operate in concert with the other embedded analog modules (ADC, CMPSS, Buffered DAC) as an analog front end system.

### 14.10.1 ADC

In its simplest application, the PGA amplifies small input signals in order to increase the ADC dynamic range. The PGA also provides the additional benefit of buffering input signals from the ADC sample and hold capacitor, which further reduces sampling error.

Both the filtered and unfiltered PGA output paths are available for ADC sampling. Minimum ADC acquisition windows are recommended when sampling the PGA paths.

#### 14.10.1.1 Unfiltered Acquisition Window

The device data manual provides a minimum estimated ADC acquisition window for sampling the PGA_OUT signal with one ADC. This estimated value should provide sampling accuracy close to the specified performance parameters of the ADC. A longer acquisition window may be used for better performance.

#### 14.10.1.2 Filtered Acquisition Window

The minimum ADC acquisition window for sampling the PGA_OF filtered signal with one ADC will vary based on the values of $R_{FILTER}$ and $C_{FILTER}$. To ensure good performance, choose a $C_{FILTER}$ capacitor that is large enough to satisfy most of the ADC sample and hold capacitor (Ch) charge requirements by itself. The $C_{FILTER}$ value can be sized based on the acceptable amount of ADC sampling error ($LSB_{Err}$):

$C_{FILTER} = C_h \times 4096 / LSB_{Err}$

See the *Analog-to-Digital Converter (ADC)* chapter to determine the optimal ADC acquisition window, where the ADC source resistance ($R_s$) is equal to $R_{FILTER}$.

### 14.10.2 CMPSS

The PGA output can be monitored by a CMPSS module for trips above or below a reference voltage. Up to two independent reference thresholds per CMPSS can be used for trip detection.

Both the filtered and unfiltered PGA output paths are available for CMPSS trip monitoring.

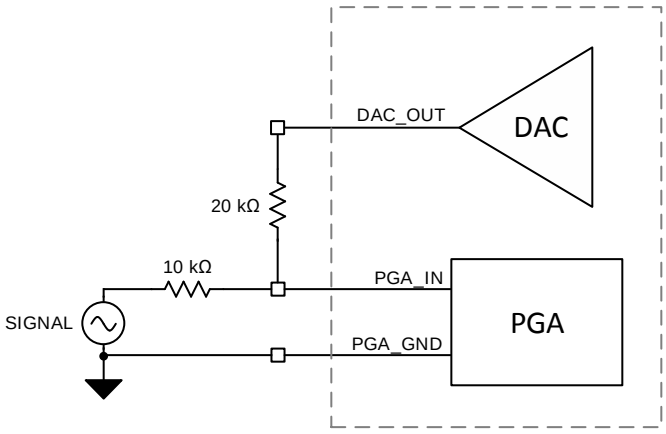
See the *Comparator Subsystem (CMPSS*) chapter for CMPSS usage information.

### 14.10.3 Buffered DAC

As a best practice, the PGA input signal should be conditioned so that the PGA output is centered within its linear range. The input signal may require some combination of offset and attenuation in order to achieve this goal.

For example, an external resistor divider can attenuate the input signal while the embedded buffered DAC can provide a positive voltage offset.

**Figure 14-8. Buffered DAC Offset**




With the topology shown above, the voltage seen at the PGA_IN pin can be calculated as follows:

$$V_{PGA\_IN} = \frac{20\ k\Omega \times (V_{SIGNAL} - V_{DAC\_OUT})}{10\ k\Omega + 20\ k\Omega} + V_{DAC\_OUT}$$

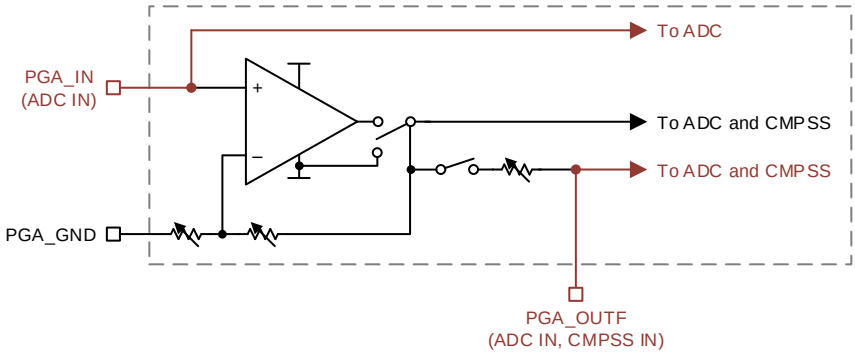
Supplying a $V_{DAC\_OUT}$ of 1.65V would transform a bipolar $V_{SIGNAL}$ range of -0.5V to +0.5V into a $V_{PGA\_IN}$ range of 0.22V to 0.88V, which is ideal for the 3x gain mode.

See the *Buffered Digital to Analog Converter (DAC)* chapter for buffered DAC usage information.

### 14.10.4 Alternate Functions

Each PGA has up to three device terminals for operation: ground, input, and output filter. Alternate paths at the device level allow the input and output filter terminals to take on alternate functions. This can be especially valuable if the respective PGA resource is not used for an application. The ground terminal should be tied to VSSA if the PGA is not used.

**Figure 14-9. PGA Pin Alternate Functions**




See the *Analog Subsystem* chapter for more information.

## 14.11 Registers

### 14.11.1 Programmable Gain Amplifier Base Addresses

**Table 14-3. I2C Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| Pga1Regs | PGA_REGS | 0000 5B00 | 0000 5B0F |
| Pga2Regs | PGA_REGS | 0000 5B10 | 0000 5B1F |
| Pga3Regs | PGA_REGS | 0000 5B20 | 0000 5B2F |
| Pga4Regs | PGA_REGS | 0000 5B30 | 0000 5B3F |
| Pga5Regs | PGA_REGS | 0000 5B40 | 0000 5B4F |
| Pga6Regs | PGA_REGS | 0000 5B50 | 0000 5B5F |
| Pga7Regs | PGA_REGS | 0000 5B60 | 0000 5B6F |

### 14.11.1.1 PGA_REGS Registers

Table 14-4 lists the memory-mapped registers for the PGA_REGS. All register offset addresses not listed in Table 14-4 should be considered as reserved locations and the register contents should not be modified.

**Table 14-4. PGA_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | PGACTL | PGA Control Register | EALLOW | Go |
| 2h | PGALOCK | PGA Lock Register | EALLOW | Go |
| 4h | PGAGAIN3TRIM | PGA Gain Trim Register for a gain setting of 3 | EALLOW | Go |
| 5h | PGAGAIN6TRIM | PGA Gain Trim Register for a gain setting of 6 | EALLOW | Go |
| 6h | PGAGAIN12TRIM | PGA Gain Trim Register for a gain setting of 12 | EALLOW | Go |
| 7h | PGAGAIN24TRIM | PGA Gain Trim Register for a gain setting of 24 | EALLOW | Go |
| 8h | PGATYPE | PGA Type Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 14-5 shows the codes that are used for access types in this section.

**Table 14-5. PGA_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| **Write Type** | | |
| W | W | Write |
| WSOnce | SOnce<br>W | Set once<br>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. |

#### 14.11.1.1.1 PGACTL Register (Offset = 0h) [reset = 0h]

PGACTL is shown in Figure 14-10 and described in Table 14-6.

Return to Summary Table.

PGA Control Register

**Figure 14-10. PGACTL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GAIN | | | FILTRESSEL | | | | PGAEN |
| R/W-0h | | | R/W-0h | | | | R/W-0h |

**Table 14-6. PGACTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-5 | GAIN | R/W | 0h | PGA Gain.<br>000 x 3<br>001 x 6<br>010 x 12<br>011 x 24<br>Reset type: SYSRSn |
| 4-1 | FILTRESSEL | R/W | 0h | 0000 Filter disabled (default)<br>0001 Filter Resistance 200 Ohm<br>0010 Filter Resistance 160 Ohm<br>0011 Filter Resistance 130 Ohm<br>0100 Filter Resistance 100 Ohm<br>0101 Filter Resistance 80 Ohm<br>0110 Filter Resistance 50 Ohm<br>Reset type: SYSRSn |
| 0 | PGAEN | R/W | 0h | PGA Enable.<br>0 PGA is disabled and powered down.<br>1 PGA is enabled.<br>Reset type: SYSRSn |

#### 14.11.1.1.2 PGALOCK Register (Offset = 2h) [reset = 0h]

PGALOCK is shown in Figure 14-11 and described in Table 14-7.

Return to Summary Table.

PGA Lock Register

**Figure 14-11. PGALOCK Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | RESERVED | PGAGAIN24TRIM | PGAGAIN12TRIM | PGAGAIN6TRIM | PGAGAIN3TRIM | RESERVED | PGACTL |
| R-0h | R-0h | WSOnce-0h | WSOnce-0h | WSOnce-0h | WSOnce-0h | R-0h | WSOnce-0h |

**Table 14-7. PGALOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | RESERVED | R | 0h | Reserved |
| 6 | RESERVED | R | 0h | Reserved |
| 5 | PGAGAIN24TRIM | WSOnce | 0h | 0 Writes to PGAGAIN24TRIM are enabled.<br>1 Writes to PGAGAIN24TRIM are disabled.<br>Reset type: SYSRSn |
| 4 | PGAGAIN12TRIM | WSOnce | 0h | 0 Writes to PGAGAIN12TRIM are enabled.<br>1 Writes to PGAGAIN12TRIM are disabled.<br>Reset type: SYSRSn |
| 3 | PGAGAIN6TRIM | WSOnce | 0h | 0 Writes to PGAGAIN6TRIM are enabled.<br>1 Writes to PGAGAIN6TRIM are disabled.<br>Reset type: SYSRSn |
| 2 | PGAGAIN3TRIM | WSOnce | 0h | 0 Writes to PGAGAIN3TRIM are enabled.<br>1 Writes to PGAGAIN3TRIM are disabled.<br>Reset type: SYSRSn |
| 1 | RESERVED | R | 0h | Reserved |
| 0 | PGACTL | WSOnce | 0h | 0 Writes to PGACTL are enabled.<br>1 Writes to PGACTL are disabled.<br>Reset type: SYSRSn |

#### 14.11.1.1.3 PGAGAIN3TRIM Register (Offset = 4h) [reset = 0h]

PGAGAIN3TRIM is shown in Figure 14-12 and described in Table 14-8.

Return to Summary Table.

PGA Gain Trim Register for a gain setting of 3

**Figure 14-12. PGAGAIN3TRIM Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| OFFSETTRIM | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GAINTRIM | | | | | | | |
| R/W-0h | | | | | | | |

**Table 14-8. PGAGAIN3TRIM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | OFFSETTRIM | R/W | 0h | Offset TRIM value, when Gain setting is 3<br>Reset type: SYSRSn |
| 7-0 | GAINTRIM | R/W | 0h | Gain TRIM value, when gain setting is 3<br>Reset type: SYSRSn |

#### 14.11.1.1.4 PGAGAIN6TRIM Register (Offset = 5h) [reset = 0h]

PGAGAIN6TRIM is shown in Figure 14-13 and described in Table 14-9.

Return to Summary Table.

PGA Gain Trim Register for a gain setting of 6

**Figure 14-13. PGAGAIN6TRIM Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| OFFSETTRIM | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GAINTRIM | | | | | | | |
| R/W-0h | | | | | | | |

**Table 14-9. PGAGAIN6TRIM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | OFFSETTRIM | R/W | 0h | Offset TRIM value, when Gain setting is 6<br>Reset type: SYSRSn |
| 7-0 | GAINTRIM | R/W | 0h | Gain TRIM value, when gain setting is 6<br>Reset type: SYSRSn |

#### 14.11.1.1.5 PGAGAIN12TRIM Register (Offset = 6h) [reset = 0h]

PGAGAIN12TRIM is shown in Figure 14-14 and described in Table 14-10.

Return to Summary Table.

PGA Gain Trim Register for a gain setting of 12

**Figure 14-14. PGAGAIN12TRIM Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| OFFSETTRIM | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GAINTRIM | | | | | | | |
| R/W-0h | | | | | | | |

**Table 14-10. PGAGAIN12TRIM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | OFFSETTRIM | R/W | 0h | Offset TRIM value, when Gain setting is 12<br>Reset type: SYSRSn |
| 7-0 | GAINTRIM | R/W | 0h | Gain TRIM value, when gain setting is 12<br>Reset type: SYSRSn |

#### 14.11.1.1.6 PGAGAIN24TRIM Register (Offset = 7h) [reset = 0h]

PGAGAIN24TRIM is shown in Figure 14-15 and described in Table 14-11.

Return to Summary Table.

PGA Gain Trim Register for a gain setting of 24

**Figure 14-15. PGAGAIN24TRIM Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| OFFSETTRIM | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GAINTRIM | | | | | | | |
| R/W-0h | | | | | | | |

**Table 14-11. PGAGAIN24TRIM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | OFFSETTRIM | R/W | 0h | Offset TRIM value, when Gain setting is 24<br>Reset type: SYSRSn |
| 7-0 | GAINTRIM | R/W | 0h | Gain TRIM value, when gain setting is 24<br>Reset type: SYSRSn |

#### 14.11.1.1.7 PGATYPE Register (Offset = 8h) [reset = 0h]

PGATYPE is shown in Figure 14-16 and described in Table 14-12.

Return to Summary Table.

PGA Type Register

**Figure 14-16. PGATYPE Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| TYPE | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| REV | | | | | | | |
| R-0h | | | | | | | |

**Table 14-12. PGATYPE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | TYPE | R | 0h | PGA Type.<br>Reset type: SYSRSn |
| 7-0 | REV | R | 0h | PGA Revision.<br>Reset type: SYSRSn |

# Buffered Digital-to-Analog Converter (DAC)

The buffered digital-to-analog converter (DAC) is an analog module that can output a programmable, arbitrary reference voltage.

**Topic** **Page**

## 15.1 Buffered Digital-to-Analog Converter (DAC) Overview

The buffered DAC module consists of an internal 12-bit DAC and an analog output buffer that is capable of driving an external load. For driving even higher loads than typical, a trade-off can be made between load size and output voltage swing. For the load conditions of the buffered DAC, see the device-specific data manual. The buffered DAC is a general-purpose DAC that can be used to generate a DC voltage in addition to AC waveforms such as sine waves, square waves, triangle waves and so forth. Software writes to the DAC value register can take effect immediately or can be synchronized with EPWMSYNCO events.

### 15.1.1 Features

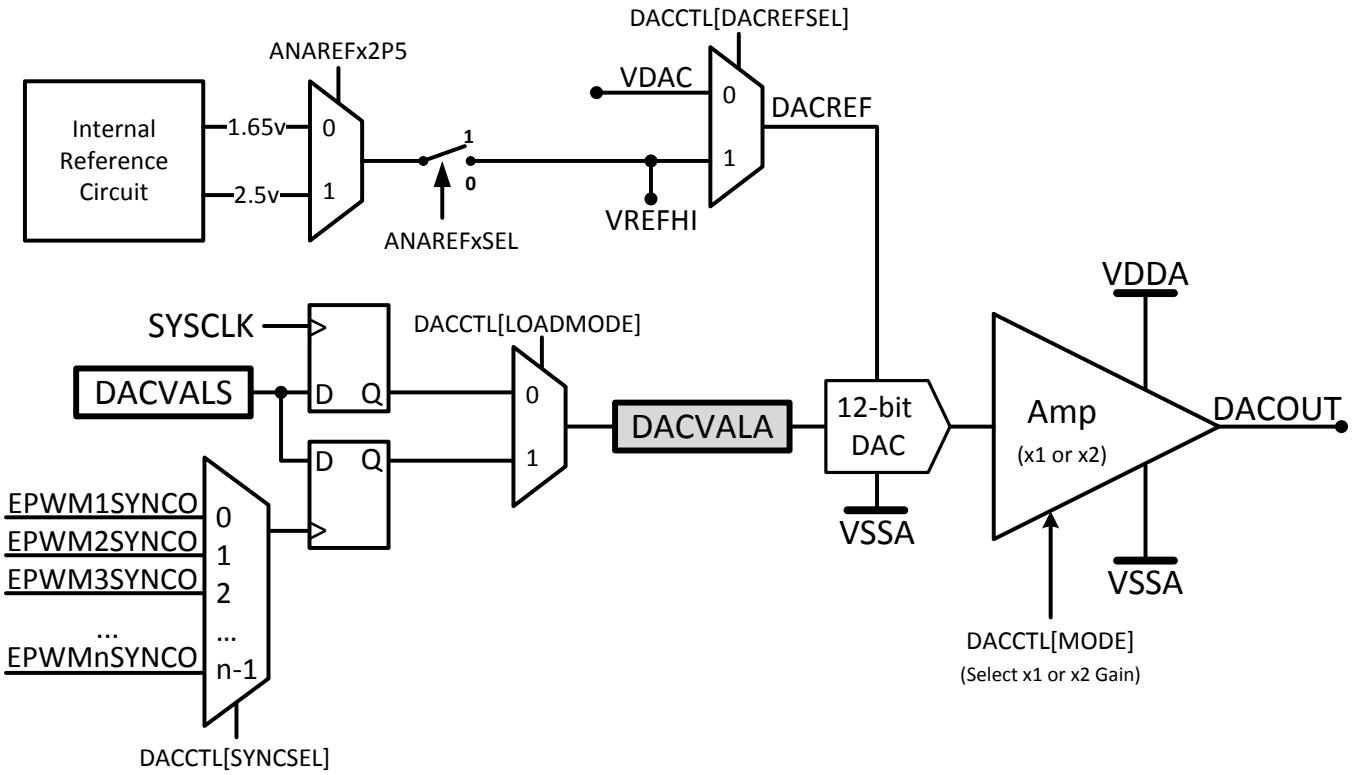
Each buffered DAC has the following features:

- 12-bit programmable internal DAC
- Selectable reference voltage source
- x1 and x2 gain modes when using internal VREFHI
- Ability to synchronize with EPWMSYNCO

### 15.1.2 Block Diagram

The block diagram for the buffered DAC is shown in Figure 15-1.

**Figure 15-1. DAC Module Block Diagram**




## 15.2 Using the DAC

The internal DAC's reference voltage source, DACREF, is selectable between VDAC and VREFHI. The x2 gain mode is only available when VREFHI is set as DACREF and internal reference mode is used (see the *Analog Subsystem* chapter on how to switch to internal reference mode). Even though the buffered DAC has an x2 gain mode, the maximum output voltage from the buffered DAC will not be greater than VDDA. Table 15-1 lists the gain mode combinations supported by the buffered DAC. In this table, x = A or B, X = Don't Care, VDAC/VREFHI = 2.5v, VDDA = 3.3v, and DACVAL = 4095.

**Table 15-1. DAC Supported Gain Mode Combinations**

| DACREFSEL | ANAREFxSEL | ANAREFx2P5 | REF SOURCE | REF VOLTAGE (V) | MODE | MAXIMUM DAC OUTPUT (V) | SUPPORT STATUS |
|---|---|---|---|---|---|---|---|
| 0 | X | X | EXTERNAL | VDAC | 0 | 2.5 | SUPPORTED |
| 0 | X | X | EXTERNAL | VDAC | 1 | 2.5 | NOT SUPPORTED |
| 1 | 0 | 0 | INTERNAL | 1.65 | 0 | 1.65 | NOT SUPPORTED |
| 1 | 0 | 0 | INTERNAL | 1.65 | 1 | 3.3 | SUPPORTED |
| 1 | 0 | 1 | INTERNAL | 2.5 | 0 | 2.5 | SUPPORTED |
| 1 | 0 | 1 | INTERNAL | 2.5 | 1 | 3.3 | NOT SUPPORTED |
| 1 | 1 | X | EXTERNAL | VREFHI | 0 | 2.5 | SUPPORTED |
| 1 | 1 | X | EXTERNAL | VREFHI | 1 | 2.5 | NOT SUPPORTED |

Two sets of DACVAL registers are present in the buffered DAC module: DACVALA and DACVALS. 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 EPWMSYNCO event. If the clock to the buffered DAC is disabled while the buffered DAC is outputting a voltage, the output voltage remains unaffected but DACVALA and DACVALS will no longer be updated with register writes. Enabling the clock to the buffered DAC restores it to the state before the clock was disabled.

The ideal output of the internal DAC can be calculated as shown in Equation 1.

(1)

$$DACOUT = \frac{DACVALA * DACREF}{4096}$$

The buffered DAC can be used to level-shift the PGA's input signal (see the PGA section for details). The output buffer of the buffered DAC may exhibit non-linear behavior near the supply rails (VDDA/VSSA). To determine the linear range of the buffered DAC, see the device-specific data manual.

### 15.2.1 Initialization Sequence

1. Enable the buffered DAC clock.
2. Set DACREF with DACREFSEL.
3. Power up the buffered DAC with DACOUTEN.
4. Wait for the power up time to elapse before outputting a voltage. To determine the power up time of the buffered DAC, see the device-specific data manual.
5. For predictable behavior of the buffered DAC, consecutive writes to DACVALS should be spaced apart according to the settling time of the buffered DAC. To determine the settling time of the buffered DAC, see the device-specific data manual.

### 15.2.2 DAC Offset Adjustment

Zero offset error is defined as the difference between the voltage at midcode (2048) and 1.25v (for 2.5v reference voltage). DAC offset error is calibrated at 2.5v reference voltage and loaded into the DAC offset trim register as part of the Device_cal() function. If the DAC is used at any reference voltage other than 2.5v, the offset trim must be adjusted to ensure the offset error performance stays within the device-specific data manual limits. The DAC offset register is a 16-bit register that contains the 8-bit signed offset trim in the lower half of the register. Use the function call DAC_tuneOffsetTrim() to adjust the offset.

## 15.3 Lock Registers

A DACLOCK register is provided to prevent spurious writes from modifying the DACCTL, DACVALS, and DACOUTEN registers. Once a register is protected through DACLOCK, write access will be locked out until the device is reset.

## 15.4 Registers

### 15.4.1 Buffered DAC Base Addresses

**Table 15-2. DAC Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| DacaRegs | DAC_REGS | 0x0000_5C00 | 0x0000_5C0F |
| DacbRegs | DAC_REGS | 0x0000_5C10 | 0x0000_5C1F |

### 15.4.1.1 DAC_REGS Registers

Table 15-3 lists the memory-mapped registers for the DAC_REGS. All register offset addresses not listed in Table 15-3 should be considered as reserved locations and the register contents should not be modified.

**Table 15-3. DAC_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | DACREV | DAC Revision Register | | 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 15-4 shows the codes that are used for access types in this section.

**Table 15-4. DAC_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| WSOnce | SOnce<br>W | Set once<br>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. |

#### 15.4.1.1.1 DACREV Register (Offset = 0h) [reset = 0h]

DACREV is shown in Figure 15-2 and described in Table 15-5.

Return to Summary Table.

DAC Revision Register

**Figure 15-2. DACREV Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| REV | | | | | | | |
| R-0h | | | | | | | |

**Table 15-5. DACREV Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-0 | REV | R | 0h | DAC Revision<br>Reset type: SYSRSn |

#### 15.4.1.1.2 DACCTL Register (Offset = 1h) [reset = 0h]

DACCTL is shown in Figure 15-3 and described in Table 15-6.

Return to Summary Table.

DAC Control Register

**Figure 15-3. DACCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SYNCSEL | | | | RESERVED | LOADMODE | MODE | DACREFSEL |
| R/W-0h | | | | R-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 15-6. DACCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-4 | SYNCSEL | R/W | 0h | DAC EPWMSYNCO select. Determines which EPWMSYNCO signal will update the DACVALA register.<br>Where n represents the maximum number of EPWMSYNCO signals available on the device:<br>0 EPWM1SYNCO<br>1 EPWM2SYNCO<br>2 EPWM3SYNCO<br>...<br>n-1 EPWMnSYNCO<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 EPWMSYNCO specified by SYNCSEL<br>Reset type: SYSRSn |
| 1 | MODE | R/W | 0h | DAC gain mode select. Selects the gain mode for the buffered output. The MODE value is only used when DACREFSEL=1 and internal ADC reference mode is selected.<br>0 Gain is 1<br>1 Gain is 2<br>Reset type: SYSRSn |
| 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/VREFLO are the reference voltages<br>Reset type: SYSRSn |

#### 15.4.1.1.3 DACVALA Register (Offset = 2h) [reset = 0h]

DACVALA is shown in Figure 15-4 and described in Table 15-7.

Return to Summary Table.

DAC Value Register - Active

**Figure 15-4. DACVALA Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | DACVALA | | | |
| R-0h | | | | R-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DACVALA | | | | | | | |
| R-0h | | | | | | | |

**Table 15-7. DACVALA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-0 | DACVALA | R | 0h | Active output code currently driven by the DAC<br>Reset type: SYSRSn |

#### 15.4.1.1.4 DACVALS Register (Offset = 3h) [reset = 0h]

DACVALS is shown in Figure 15-5 and described in Table 15-8.

Return to Summary Table.

DAC Value Register - Shadow

**Figure 15-5. DACVALS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | DACVALS | | | |
| R-0h | | | | R/W-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DACVALS | | | | | | | |
| R/W-0h | | | | | | | |

**Table 15-8. DACVALS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-0 | DACVALS | R/W | 0h | Shadow output code to be loaded into DACVALA<br>Reset type: SYSRSn |

#### 15.4.1.1.5 DACOUTEN Register (Offset = 4h) [reset = 0h]

DACOUTEN is shown in Figure 15-6 and described in Table 15-9.

Return to Summary Table.

DAC Output Enable Register

**Figure 15-6. DACOUTEN Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | DACOUTEN |
| R-0h | | | | | | | R/W-0h |

**Table 15-9. DACOUTEN Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-1 | RESERVED | R | 0h | Reserved |
| 0 | DACOUTEN | R/W | 0h | DAC output enable<br>0 DAC output is disabled<br>1 DAC output is enabled<br>Reset type: SYSRSn |

#### 15.4.1.1.6 DACLOCK Register (Offset = 5h) [reset = 0h]

DACLOCK is shown in Figure 15-7 and described in Table 15-10.

Return to Summary Table.

DAC Lock Register

**Figure 15-7. DACLOCK Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| KEY | | | | RESERVED | | | |
| R=0/W-0h | | | | R-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | DACOUTEN | DACVAL | DACCTL |
| R-0h | | | | | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h |

**Table 15-10. DACLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | KEY | R=0/W | 0h | Writes to this register succeed only if this field is written with a value of 0xA.<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 |

#### 15.4.1.1.7 DACTRIM Register (Offset = 6h) [reset = 0h]

DACTRIM is shown in Figure 15-8 and described in Table 15-11.

Return to Summary Table.

DAC Trim Register

**Figure 15-8. DACTRIM Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | RESERVED | | | |
| R-0h | | | | R-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OFFSET_TRIM | | | | | | | |
| R/W-0h | | | | | | | |

**Table 15-11. DACTRIM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-8 | RESERVED | R | 0h | Reserved |
| 7-0 | OFFSET_TRIM | R/W | 0h | DAC Offset Trim.<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 |

# Comparator Subsystem (CMPSS)

The Comparator Subsystem (CMPSS) consists of analog comparators and supporting circuits that are useful for power applications such as peak current mode control, switched-mode power, power factor correction, voltage trip monitoring, and so forth.

**Topic** **Page**

## 16.1 CMPSS Overview

The comparator subsystem is built around a number of modules. Each subsystem contains two comparators, two reference 12-bit DACs, two digital filters and one ramp generator. Comparators are denoted "H" or "L" within each module where “H” and “L” represent high and low, respectively. Each comparator generates a digital output which indicates whether the voltage on the positive input is greater than the voltage on the negative input. The positive input of the comparator can be driven from an external pin or by the PGA (see the *Analog Subsystem* chapter for mux options available to the CMPSS). The negative input can be driven by an external pin or by the programmable reference 12-bit DAC. Each comparator output passes through a programmable digital filter that can remove spurious trip signals. An unfiltered output is also available if filtering is not required. A ramp generator circuit is optionally available to control the reference 12-bit DAC value for the high comparator in the subsystem.

### *16.1.1 Features*

Each CMPSS includes:

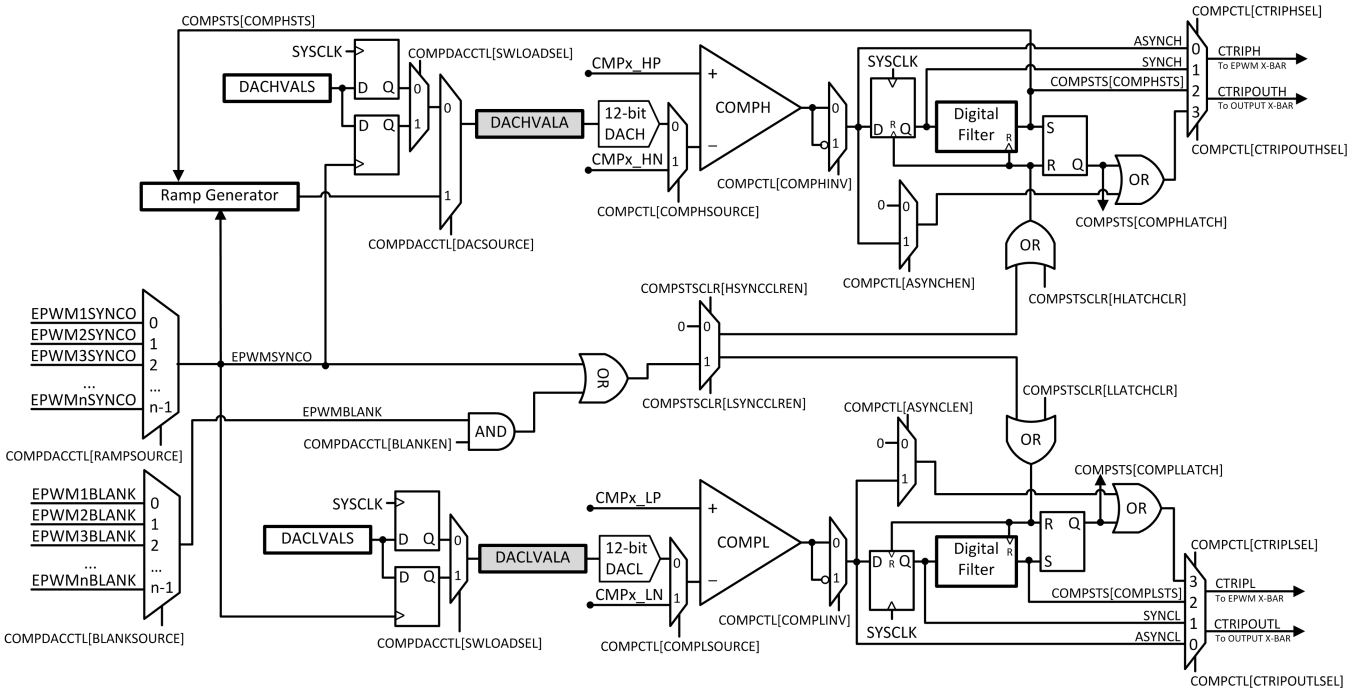
- Two analog comparators
- Two programmable reference 12-bit DACs
- One ramp generator
- Two digital filters
- Ability to synchronize submodules with EPWMSYNCO
- Ability to extend clear signal with EPWMBLANK
- Ability to synchronize output with SYSCLK
- Ability to latch output
- Ability to invert output
- Option to use hysteresis on the input
- Option for positive input of comparator to be driven by an external signal or by the PGA
- Option for negative input of comparator to be driven by an external signal or by the reference DAC
- Option to choose between VDDA or VDAC to be the DAC reference voltage

### *16.1.2 Block Diagram*

The block diagram for the CMPSS is shown in Figure 16-1.

- CTRIPx(x= "H" or "L") signals are connected to the ePWM X-BAR for ePWM trip response. See the *ePWM* chapter for more details on the ePWM X-BAR mux configuration.
- CTRIPxOUTx(x= "H" or "L") signals are connected to the Output X-BAR for external signaling. See the *GPIO* chapter for more details on the Output X-BAR mux configuration.

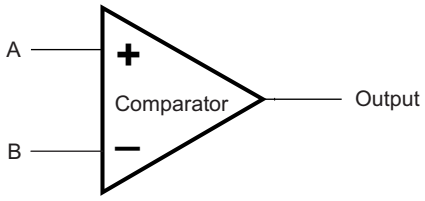
**Figure 16-1. CMPSS Module Block Diagram**




## 16.2 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 16-2.

**Figure 16-2. Comparator Block Diagram**




| Voltages | Output |
|---|---|
| Voltage A > Voltage B | 1 |
| Voltage A < Voltage B | 0 |

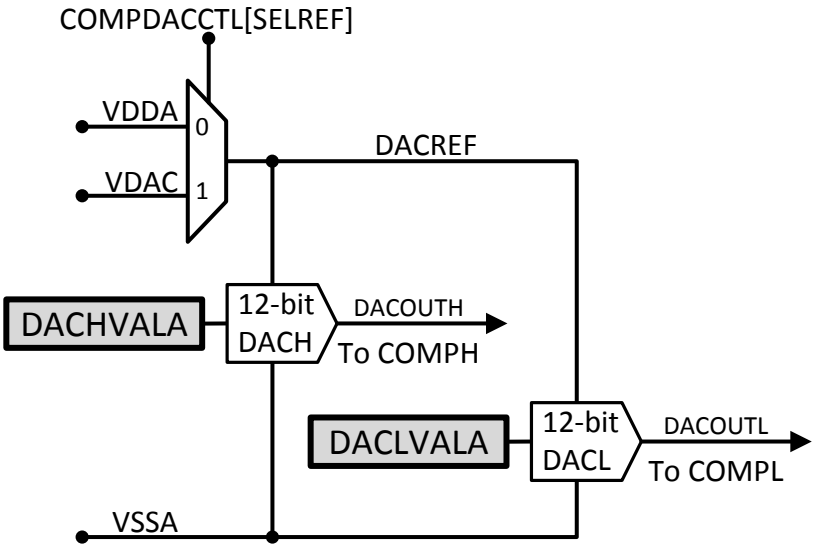
## 16.3 Reference DAC

Each reference 12-bit DAC can be configured to drive a reference voltage into the negative input of its respective comparator. The reference 12-bit DAC output is internal only and cannot be observed externally.

Two sets of DACxVAL registers are present for each reference 12-bit DAC: DACxVALA and DACxVALS. 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 EPWMSYNCO event. The high reference 12-bit DAC (DACH) can optionally source its DACHVALA value from the ramp generator instead of DACHVALS.

The operating range of the reference 12-bit DAC is bounded by DACREF and VSSA. The high voltage reference is VDDA by default, but it can be configured to be VDAC. The reference 12-bit DAC is illustrated in Figure 16-3.

**Figure 16-3. Reference DAC Block Diagram**




The ideal output of the reference 12-bit DAC can be calculated as follows:

**Figure 16-4. Output Voltage Calculation**

$$DACOUT = \frac{DACVALA * DACREF}{4096}$$

## 16.4 Ramp Generator

### 16.4.1 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 its input. The low 4 bits of the RAMPSTS countdown register effectively act as a prescale for the falling-ramp rate configurable with RAMPDECVALA.

The ramp generator is enabled by setting DACSOURCE = 1. On setting DACSOURCE = 1, the value of RAMPSTS is loaded from RAMPMAXREFS and the register remains static until the selected EPWMSYNCO signal is received. After receiving the selected EPWMSYNCO 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 EPWMSYNCO event, the RAMPDLYA register which serves as a delay counter can be used to hold off the RAMPSTS subtraction. On receiving a EPWMSYNCO event, the value of RAMPDLYA is decremented by one on every SYSCLK cycle until the register reaches zero. The RAMPSTS subtraction will only begin when RAMPDLYA is zero.

### 16.4.2 Ramp Generator Behavior

The ramp generator makes state changes on every rising edge of DACSOURCE, EPWMSYNCO and COMPHSTS.

On the rising edge of DACSOURCE; RAMPMAXREFA, RAMPDECVALA and RAMPDLYA are loaded with their shadow registers. RAMPSTS is loaded with RAMPMAXREFS.

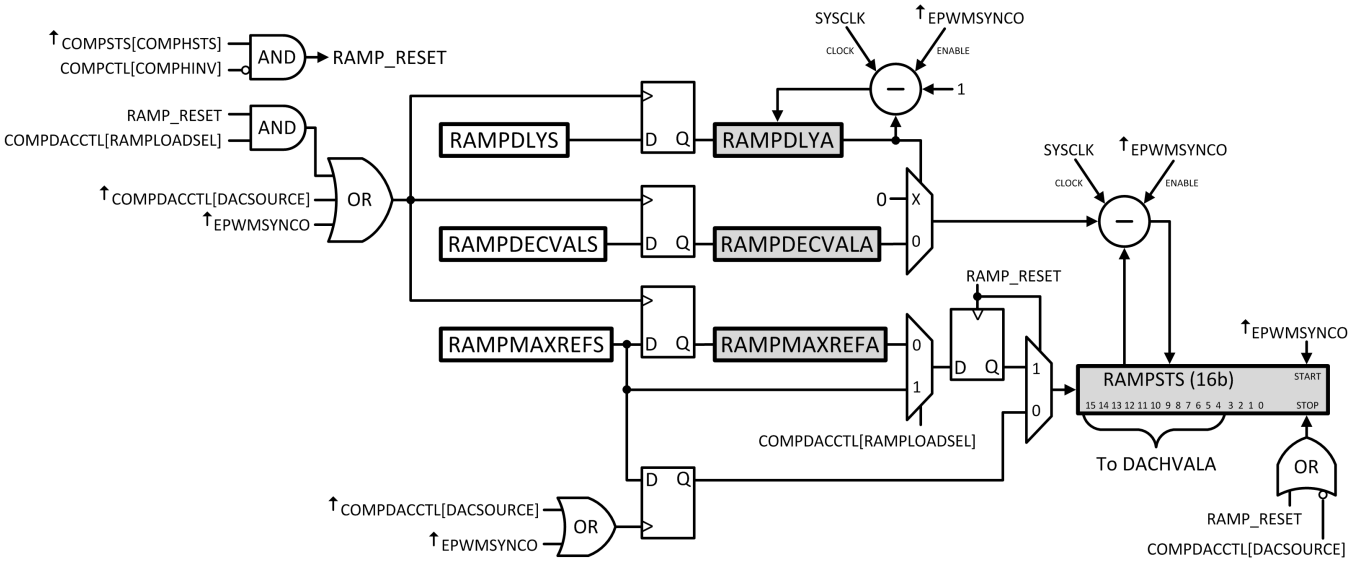
On the rising edge of the selected EPWMSYNCO; RAMPMAXREFA, RAMPDECVALA and RAMPDLYA are loaded with their shadow registers. RAMPSTS is loaded with RAMPMAXREFS and starts decrementing when RAMPDLYA counter reaches zero.

On the rising edge of COMPHSTS with RAMPLOADSEL = 1; RAMPMAXREFA, RAMPDECVALA and RAMPDLYA are loaded with their shadow registers. RAMPSTS is loaded with RAMPMAXREFS and stops decrementing.

On the rising edge of COMPHSTS with RAMPLOADSEL = 0; RAMPSTS is loaded with RAMPMAXREFA and stops decrementing.

Additionally, if the value of RAMPSTS reaches zero, the RAMPSTS register will remain static at zero until the next EPWMSYNCO is received. These state changes are illustrated in the ramp generator block diagram in Figure 16-5.

**Figure 16-5. Ramp Generator Block Diagram**



### 16.4.3 Ramp Generator Behavior at Corner Cases

Since the ramp generator makes state changes on every rising edge of EPWMSYNCO and COMPHSTS, the following behavior can be expected on instances when these two events occur simultaneously or very close together.

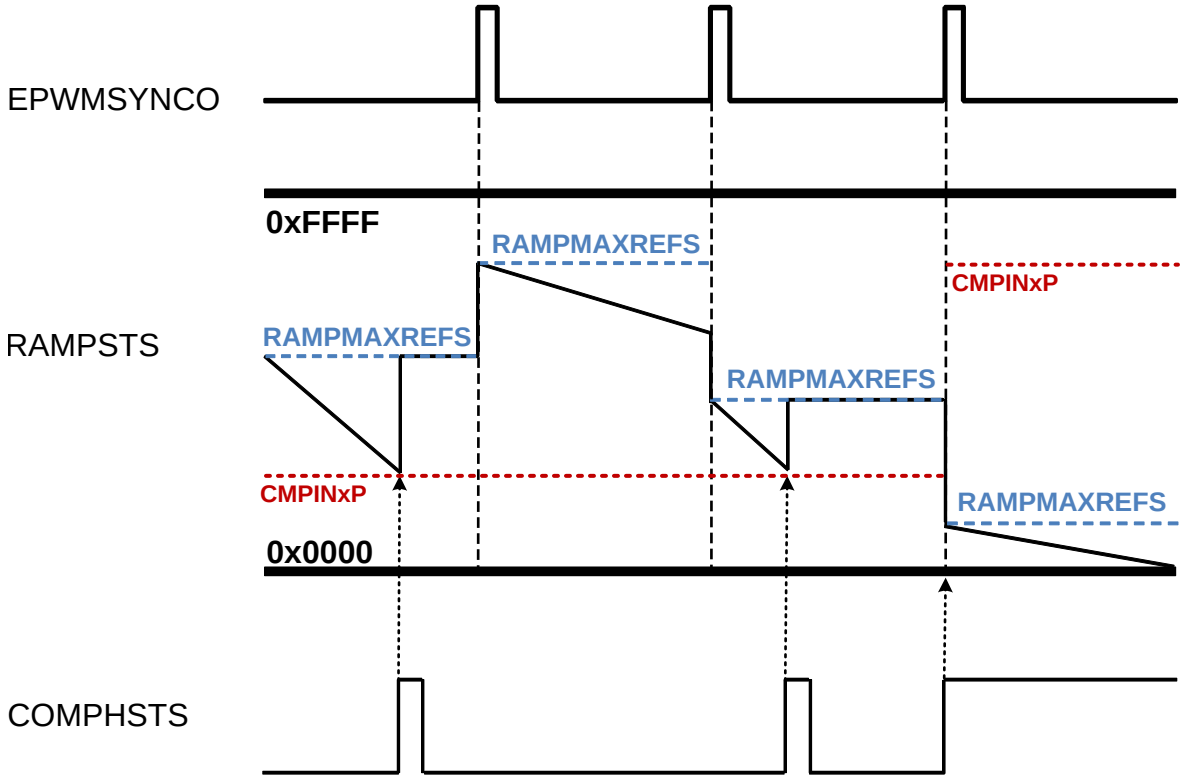
Case 1: COMPHSTS rising edge occurs one or more cycles before EPWMSYNCO rising edge.
RAMPSTS stops decrementing on COMPHSTS rising edge event. RAMPSTS starts decrementing on EPWMSYNCO rising edge event when RAMPDLYA reaches 0.

Case 2: COMPHSTS rising edge occurs simultaneously as EPWMSYNCO rising edge.
EPWMSYNCO rising edge event takes precedence and RAMPSTS starts decrementing when RAMPDLYA reaches 0. COMPHSTS rising edge event is ignored and does not halt RAMPSTS.

Case 3: COMPHSTS rising edge occurs one or more cycles after EPWMSYNCO 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 EPWMSYNCO rising edge.
RAMPSTS does not decrement.

**Figure 16-6. Ramp Generator Behavior**




## 16.5 Digital Filter

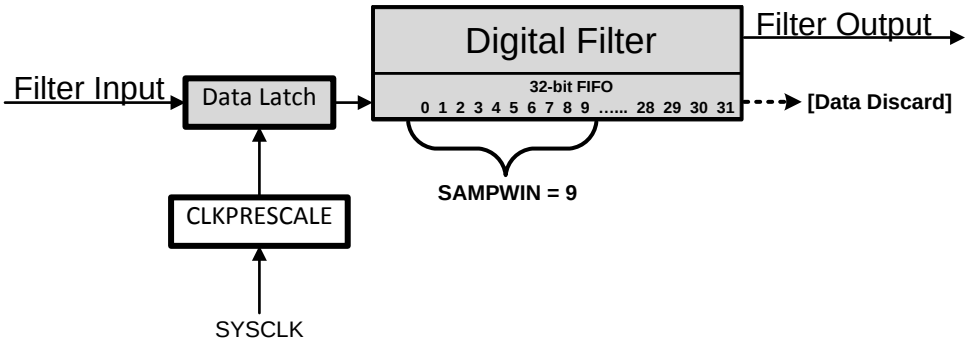
The digital filter works on a window of FIFO samples (SAMPWIN + 1) taken from the input. The filter output resolves to the majority value of the sample window, where majority is defined by the threshold (THRESH) value. If the majority threshold is not satisfied, the filter output remains unchanged.

For proper operation, the value of THRESH must be greater than SAMPWIN / 2.

A prescale function (CLKPRESCALE) determines the filter sampling rate, where the filter FIFO captures one sample every CLKPRESCALE system clocks. Old data from the FIFO is discarded.

A conceptual model of the digital filter is shown in Figure 16-7.

**Figure 16-7. Digital Filter Behavior**




Equivalent C code of the filter implementation is shown below:

```
if (FILTER_OUTPUT == 0) {
    if (Num_1s_in_SAMPWIN >= THRESH) {
        FILTER_OUTPUT = 1;
    }
}
else {
    if (Num_0s_in_SAMPWIN >= THRESH) {
        FILTER_OUTPUT = 0;
    }
}
```

### 16.5.1 Filter Initialization Sequence

To ensure proper operation of the digital filter, the following initialization sequence is recommended:

1. Configure and enable the comparator for operation
2. Configure the digital filter parameters for operation
   - Set SAMPWIN for the number of samples to monitor in the FIFO window
   - Set THRESH for the threshold required for majority qualification
   - Set CLKPRESCALE for the digital filter clock prescale value
3. Initialize the sample values in the digital FIFO window by setting FILINIT
4. Clear COMPSTS latch via COMPSTSCLR if the latched path is desired
5. Configure the CTRIP and CTRIPOUT signal paths
6. If desired, configure the ePWM and GPIO modules to accept the filtered signals

## 16.6 Using the CMPSS

### 16.6.1 LATCHCLR, EPWMSYNCO and EPWMBLANK Signals

The LATCHCLR signal clears the synchronize block, digital filter and the latch. It is activated in software using xLATCHCLR(x= “H” or “L”). It can also be activated by EPWMSYNCO when xSYNCCLREN(x= "H" or "L") is set. If a longer LATCHCLR signal is required, the EPWMBLANK signal can be used to extend it by setting BLANKEN.

### 16.6.2 Enabling and Disabling the CMPSS Clock

If the clock to the CMPSS module is disabled while the comparator is active, the following behavior can be expected:

- The comparator remains unaffected and will continue to trip from voltages on its inputs.
- If the reference 12-bit DAC is driving the negative input of the comparator, the voltage on the negative input remains static and unaffected but DACVALA would no longer be updated from the ramp generator or DACVALS.

- The ramp generator, synchronize block and digital filter freeze on their current states.

Enabling the clock to the CMPSS restores it to the state before the clock was disabled.

## 16.7 Registers

### 16.7.1 CMPSS Base Addresses

**Table 16-1. CMPSS Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| Cmpss1Regs | CMPSS_REGS | 0x0000_5C80 | 0x0000_5C9F |
| Cmpss2Regs | CMPSS_REGS | 0x0000_5CA0 | 0x0000_5CBF |
| Cmpss3Regs | CMPSS_REGS | 0x0000_5CC0 | 0x0000_5CDF |
| Cmpss4Regs | CMPSS_REGS | 0x0000_5CE0 | 0x0000_5CFF |
| Cmpss5Regs | CMPSS_REGS | 0x0000_5D00 | 0x0000_5D1F |
| Cmpss6Regs | CMPSS_REGS | 0x0000_5D20 | 0x0000_5D3F |
| Cmpss7Regs | CMPSS_REGS | 0x0000_5D40 | 0x0000_5D5F |

### 16.7.1.1 CMPSS_REGS Registers

Table 16-2 lists the memory-mapped registers for the CMPSS_REGS. All register offset addresses not listed in Table 16-2 should be considered as reserved locations and the register contents should not be modified.

**Table 16-2. CMPSS_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | COMPCTL | CMPSS Comparator Control Register | EALLOW | 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 16-3 shows the codes that are used for access types in this section.

**Table 16-3. CMPSS_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| W=1 | W | Write |
| WSOnce | SOnce<br>W | Set once<br>Write |
| **Reset or Default Value** | | |
| *-n* | | Value after reset or the default value |
| **Register Array Variables** | | |

**Table 16-3. CMPSS_REGS Access Type Codes (continued)**

| Access Type | Code | Description |
|---|---|---|
| i,j,k,l,m,n | | When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula. |
| y | | When this variable is used in a register name, an offset, or an address it refers to the value of a register array. |

#### 16.7.1.1.1 COMPCTL Register (Offset = 0h) [reset = 0h]

COMPCTL is shown in Figure 16-8 and described in Table 16-4.

Return to Summary Table.

CMPSS Comparator Control Register

**Figure 16-8. 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 16-4. 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 |

**Table 16-4. 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 |

#### 16.7.1.1.2 COMPHYSCTL Register (Offset = 1h) [reset = 0h]

COMPHYSCTL is shown in Figure 16-9 and described in Table 16-5.

Return to Summary Table.

CMPSS Comparator Hysteresis Control Register

**Figure 16-9. COMPHYSCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | COMPHYS | | |
| R-0h | | | | | R/W-0h | | |

**Table 16-5. 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>Reset type: SYSRSn |

#### 16.7.1.1.3 COMPSTS Register (Offset = 2h) [reset = 0h]

COMPSTS is shown in Figure 16-10 and described in Table 16-6.

Return to Summary Table.

CMPSS Comparator Status Register

**Figure 16-10. COMPSTS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | COMPLLATCH | COMPLSTS |
| R-0h | | | | | | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | COMPHLATCH | COMPHSTS |
| R-0h | | | | | | R-0h | R-0h |

**Table 16-6. COMPSTS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-10 | RESERVED | R | 0h | Reserved |
| 9 | COMPLLATCH | R | 0h | Latched value of low comparator digital filter output<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 |

#### 16.7.1.1.4 COMPSTSCLR Register (Offset = 3h) [reset = 0h]

COMPSTSCLR is shown in Figure 16-11 and described in Table 16-7.

Return to Summary Table.

CMPSS Comparator Status Clear Register

**Figure 16-11. COMPSTSCLR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | LSYNCCLREN | LLATCHCLR | RESERVED |
| R-0h | | | | | R/W-0h | R=0/W=1-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | HSYNCCLREN | HLATCHCLR | RESERVED |
| R-0h | | | | | R/W-0h | R=0/W=1-0h | R-0h |

**Table 16-7. COMPSTSCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-11 | RESERVED | R | 0h | Reserved |
| 10 | LSYNCCLREN | R/W | 0h | Low comparator latch EPWMSYNCO clear. Enable EPWMSYNCO reset of low comparator digital filter output latch COMPSTS[COMPLLATCH].<br>0 EPWMSYNCO will not reset latch<br>1 EPWMSYNCO will reset latch<br>Reset type: SYSRSn |
| 9 | LLATCHCLR | R=0/W=1 | 0h | Low comparator latch software clear. Perform software reset of low comparator digital filter output latch COMPSTS[COMPLLATCH]. Reads always return 0.<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 EPWMSYNCO clear. Enable EPWMSYNCO reset of high comparator digital filter output latch COMPSTS[COMPHLATCH].<br>0 EPWMSYNCO will not reset latch<br>1 EPWMSYNCO will reset latch<br>Reset type: SYSRSn |
| 1 | HLATCHCLR | R=0/W=1 | 0h | High comparator latch software clear. Perform software reset of high comparator digital filter output latch COMPSTS[COMPHLATCH]. Reads always return 0.<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 |

#### 16.7.1.1.5 COMPDACCTL Register (Offset = 4h) [reset = 0h]

COMPDACCTL is shown in Figure 16-12 and described in Table 16-8.

Return to Summary Table.

CMPSS DAC Control Register

**Figure 16-12. COMPDACCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| FREESOFT | | RESERVED | BLANKEN | BLANKSOURCE | | | |
| R/W-0h | | R-0h | R/W-0h | R/W-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SWLOADSEL | RAMPLOADSEL | SELREF | RAMPSOURCE | | | | DACSOURCE |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | | | | R/W-0h |

**Table 16-8. COMPDACCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | FREESOFT | R/W | 0h | Free-run or software-run emulation behavior. Behavior of the ramp generator during emulation suspend.<br>00b Ramp generator stops immediately during emulation suspend<br>01b Ramp generator completes current ramp and stops at next EPWMSYNCO during emulation suspend<br>1Xb Ramp generator runs freely<br>Reset type: SYSRSn |
| 13 | RESERVED | R | 0h | Reserved |
| 12 | BLANKEN | R/W | 0h | EPWMBLANK enable. This bit enables the EPWMBLANK signal.<br>0 EPWMBLANK signal is disabled.<br>1 EPWMBLANK signal is enabled.<br>Reset type: SYSRSn |
| 11-8 | BLANKSOURCE | R/W | 0h | EPWMBLANK source select. This bit field determines which EPWMnBLANK is passed on as the EPWMBLANK signal.<br>Where n represents the maximum number of EPWMBLANK signals available on the device:<br>0 EPWM1BLANK<br>1 EPWM2BLANK<br>2 EPWM3BLANK<br>...<br>n-1 EPWMnBLANK<br>Reset type: SYSRSn |
| 7 | SWLOADSEL | R/W | 0h | Software load select. Determines whether DACxVALA is updated from DACxVALS on SYSCLK or EPWMSYNCO.<br>0 DACxVALA is updated from DACxVALS on SYSCLK<br>1 DACxVALA is updated from DACxVALS on EPWMSYNCO<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 |

**Table 16-8. COMPDACCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | SELREF | R/W | 0h | DAC reference select. Determines which voltage supply is used as the reference for the internal comparator DACs.<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 EPWMSYNCO signal is used within the CMPSS module.<br>Where n represents the maximum number of EPWMSYNCO signals available on the device:<br>0 EPWM1SYNCO<br>1 EPWM2SYNCO<br>2 EPWM3SYNCO<br>...<br>n-1 EPWMnSYNCO<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 |

#### 16.7.1.1.6 DACHVALS Register (Offset = 6h) [reset = 0h]

DACHVALS is shown in Figure 16-13 and described in Table 16-9.

Return to Summary Table.

CMPSS High DAC Value Shadow Register

**Figure 16-13. DACHVALS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | DACVAL | | | |
| R-0h | | | | R/W-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DACVAL | | | | | | | |
| R/W-0h | | | | | | | |

**Table 16-9. DACHVALS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-0 | DACVAL | R/W | 0h | High DAC shadow value. When COMPDACCTL[DACSOURCE]=0, the value of DACHVALS is loaded into DACHVALA on the trigger signal selected by COMPDACCTL[SWLOADSEL].<br>Reset type: SYSRSn |

#### 16.7.1.1.7 DACHVALA Register (Offset = 7h) [reset = 0h]

DACHVALA is shown in Figure 16-14 and described in Table 16-10.

Return to Summary Table.

CMPSS High DAC Value Active Register

**Figure 16-14. DACHVALA Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | DACVAL | | | |
| R-0h | | | | R-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DACVAL | | | | | | | |
| R-0h | | | | | | | |

**Table 16-10. DACHVALA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-0 | DACVAL | R | 0h | High DAC active value. Value that is actively driven by the high DAC.<br>Reset type: SYSRSn |

#### 16.7.1.1.8 RAMPMAXREFA Register (Offset = 8h) [reset = 0h]

RAMPMAXREFA is shown in Figure 16-15 and described in Table 16-11.

Return to Summary Table.

CMPSS Ramp Max Reference Active Register

**Figure 16-15. RAMPMAXREFA Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RAMPMAXREF | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 16-11. RAMPMAXREFA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RAMPMAXREF | R | 0h | Ramp maximum reference active value. Latched value to be loaded into ramp generator RAMPSTS.<br>Reset type: SYSRSn |

#### 16.7.1.1.9 RAMPMAXREFS Register (Offset = Ah) [reset = 0h]

RAMPMAXREFS is shown in Figure 16-16 and described in Table 16-12.

Return to Summary Table.

CMPSS Ramp Max Reference Shadow Register

**Figure 16-16. RAMPMAXREFS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RAMPMAXREF | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 16-12. RAMPMAXREFS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RAMPMAXREF | R/W | 0h | Ramp maximum reference shadow. Unlatched value to be loaded into ramp generator RAMPSTS.<br>Reset type: SYSRSn |

#### 16.7.1.1.10 RAMPDECVALA Register (Offset = Ch) [reset = 0h]

RAMPDECVALA is shown in Figure 16-17 and described in Table 16-13.

Return to Summary Table.

CMPSS Ramp Decrement Value Active Register

**Figure 16-17. RAMPDECVALA Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RAMPDECVAL | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 16-13. RAMPDECVALA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RAMPDECVAL | R | 0h | Ramp decrement value active. Latched value that will be subtracted from RAMPSTS.<br>Reset type: SYSRSn |

#### 16.7.1.1.11 RAMPDECVALS Register (Offset = Eh) [reset = 0h]

RAMPDECVALS is shown in Figure 16-18 and described in Table 16-14.

Return to Summary Table.

CMPSS Ramp Decrement Value Shadow Register

**Figure 16-18. RAMPDECVALS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RAMPDECVAL | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 16-14. RAMPDECVALS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RAMPDECVAL | R/W | 0h | Ramp decrement value shadow. Unlatched value to be loaded into RAMPDECVALA.<br>Reset type: SYSRSn |

#### 16.7.1.1.12 RAMPSTS Register (Offset = 10h) [reset = 0h]

RAMPSTS is shown in Figure 16-19 and described in Table 16-15.

Return to Summary Table.

CMPSS Ramp Status Register

**Figure 16-19. RAMPSTS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RAMPVALUE | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 16-15. RAMPSTS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | RAMPVALUE | R | 0h | Ramp value. Present value of ramp generator.<br>Reset type: SYSRSn |

#### 16.7.1.1.13 DACLVALS Register (Offset = 12h) [reset = 0h]

DACLVALS is shown in Figure 16-20 and described in Table 16-16.

Return to Summary Table.

CMPSS Low DAC Value Shadow Register

**Figure 16-20. DACLVALS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | DACVAL | | | |
| R-0h | | | | R/W-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DACVAL | | | | | | | |
| R/W-0h | | | | | | | |

**Table 16-16. DACLVALS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-0 | DACVAL | R/W | 0h | Low DAC shadow value. value to be loaded into DACLVALA on the trigger signal selected by COMPDACCTL[SWLOADSEL].<br>Reset type: SYSRSn |

#### 16.7.1.1.14 DACLVALA Register (Offset = 13h) [reset = 0h]

DACLVALA is shown in Figure 16-21 and described in Table 16-17.

Return to Summary Table.

CMPSS Low DAC Value Active Register

**Figure 16-21. DACLVALA Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | DACVAL | | | |
| R-0h | | | | R-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DACVAL | | | | | | | |
| R-0h | | | | | | | |

**Table 16-17. DACLVALA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-0 | DACVAL | R | 0h | Low DAC active value. Value that is actively driven by the low DAC.<br>Reset type: SYSRSn |

#### 16.7.1.1.15 RAMPDLYA Register (Offset = 14h) [reset = 0h]

RAMPDLYA is shown in Figure 16-22 and described in Table 16-18.

Return to Summary Table.

CMPSS Ramp Delay Active Register

**Figure 16-22. RAMPDLYA Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | DELAY | | | | |
| R-0h | | | R-0h | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DELAY | | | | | | | |
| R-0h | | | | | | | |

**Table 16-18. RAMPDLYA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12-0 | DELAY | R | 0h | Ramp delay active value. Latched value of the number of cycles to delay the start of the ramp generator decrementer after a EPWMSYNCO is received.<br>Reset type: SYSRSn |

#### 16.7.1.1.16 RAMPDLYS Register (Offset = 15h) [reset = 0h]

RAMPDLYS is shown in Figure 16-23 and described in Table 16-19.

Return to Summary Table.

CMPSS Ramp Delay Shadow Register

**Figure 16-23. RAMPDLYS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | DELAY | | | | |
| R-0h | | | R/W-0h | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DELAY | | | | | | | |
| R/W-0h | | | | | | | |

**Table 16-19. RAMPDLYS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12-0 | DELAY | R/W | 0h | Ramp delay shadow value. Unlatched value to be loaded into RAMPDLYA.<br>Reset type: SYSRSn |

#### 16.7.1.1.17 CTRIPLFILCTL Register (Offset = 16h) [reset = 0h]

CTRIPLFILCTL is shown in Figure 16-24 and described in Table 16-20.

Return to Summary Table.

CTRIPL Filter Control Register

**Figure 16-24. CTRIPLFILCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| FILINIT | RESERVED | THRESH | | | | | SAMPWIN |
| R=0/W=1-0h | R-0h | R/W-0h | | | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SAMPWIN | | | | RESERVED | | | |
| R/W-0h | | | | R-0h | | | |

**Table 16-20. CTRIPLFILCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | FILINIT | R=0/W=1 | 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.<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 |

#### 16.7.1.1.18 CTRIPLFILCLKCTL Register (Offset = 17h) [reset = 0h]

CTRIPLFILCLKCTL is shown in Figure 16-25 and described in Table 16-21.

Return to Summary Table.

CTRIPL Filter Clock Control Register

**Figure 16-25. CTRIPLFILCLKCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | CLKPRESCALE | |
| R-0h | | | | | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CLKPRESCALE | | | | | | | |
| R/W-0h | | | | | | | |

**Table 16-21. CTRIPLFILCLKCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-10 | RESERVED | R | 0h | Reserved |
| 9-0 | CLKPRESCALE | R/W | 0h | Low filter sample clock prescale. Number of system clocks between samples.<br>Reset type: SYSRSn |

#### 16.7.1.1.19 CTRIPHFILCTL Register (Offset = 18h) [reset = 0h]

CTRIPHFILCTL is shown in Figure 16-26 and described in Table 16-22.

Return to Summary Table.

CTRIPH Filter Control Register

**Figure 16-26. CTRIPHFILCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| FILINIT | RESERVED | THRESH | | | | | SAMPWIN |
| R=0/W=1-0h | R-0h | R/W-0h | | | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SAMPWIN | | | | RESERVED | | | |
| R/W-0h | | | | R-0h | | | |

**Table 16-22. CTRIPHFILCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | FILINIT | R=0/W=1 | 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.<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 |

#### 16.7.1.1.20 CTRIPHFILCLKCTL Register (Offset = 19h) [reset = 0h]

CTRIPHFILCLKCTL is shown in Figure 16-27 and described in Table 16-23.

Return to Summary Table.

CTRIPH Filter Clock Control Register

**Figure 16-27. CTRIPHFILCLKCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | CLKPRESCALE | |
| R-0h | | | | | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CLKPRESCALE | | | | | | | |
| R/W-0h | | | | | | | |

**Table 16-23. 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.<br>Reset type: SYSRSn |

#### 16.7.1.1.21 COMPLOCK Register (Offset = 1Ah) [reset = 0h]

COMPLOCK is shown in Figure 16-28 and described in Table 16-24.

Return to Summary Table.

CMPSS Lock Register

**Figure 16-28. COMPLOCK Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | RESERVED | CTRIP | DACCTL | COMPHYSCTL | COMPCTL |
| R-0h | | | R-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h | R/WSOnce-0h |

**Table 16-24. COMPLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 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 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 |
| 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 |

# Sigma Delta Filter Module (SDFM)

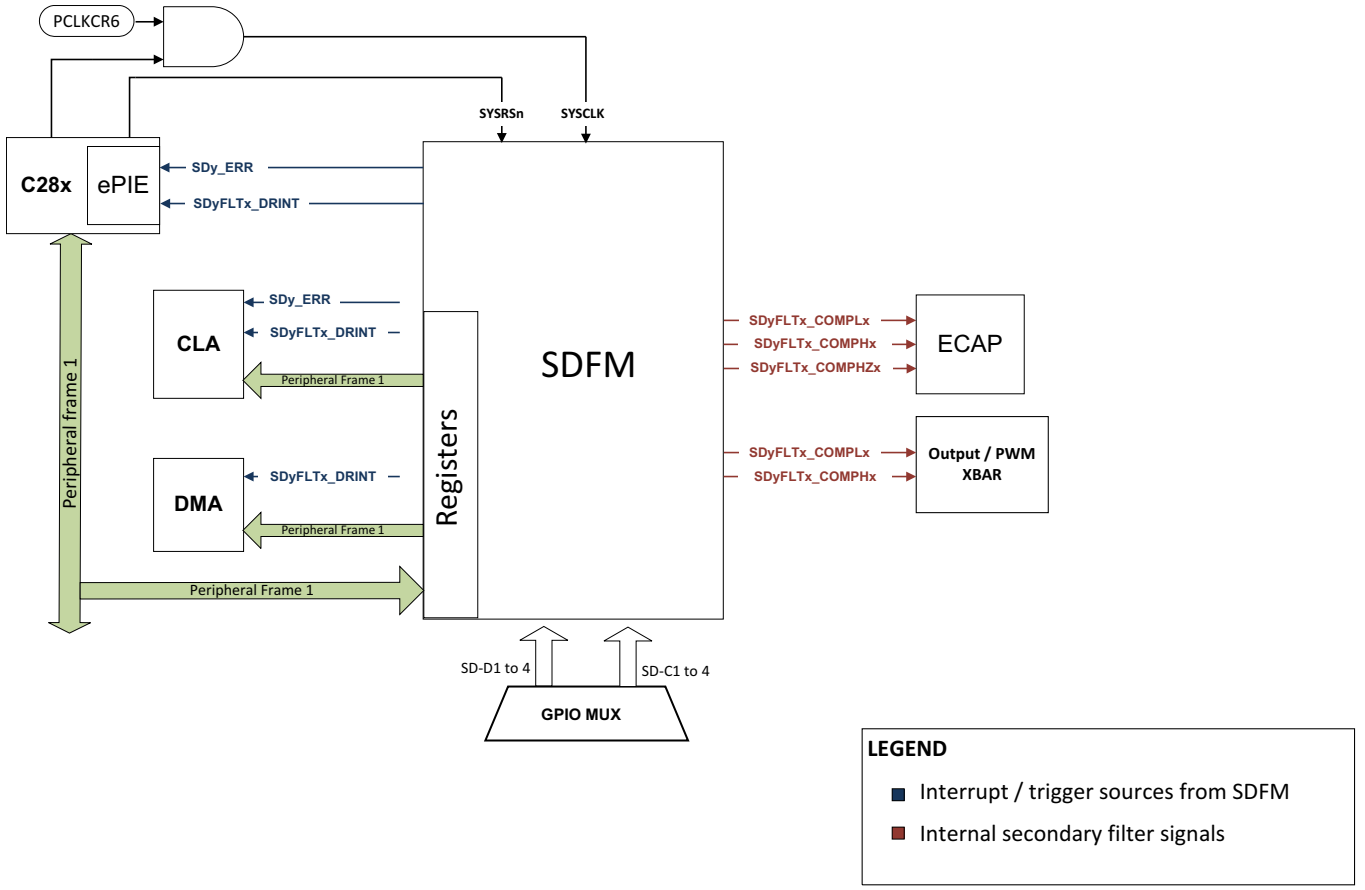
This chapter describes the sigma delta filter module (SDFM). SDFM is a four-channel digital filter designed specifically for current measurement and resolver position decoding in motor control applications. Each input channel can receive an independent delta-sigma (ΔΣ) modulator bit stream. The bit streams 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.

**Topic** **Page**

## 17.1 SDFM Module Overview

Figure 17-1 shows the SDFM CPU interfaces.

**Figure 17-1. Sigma Delta Filter Module (SDFM) CPU Interface**




### 17.1.1 SDFM Features

The SDFM features include:

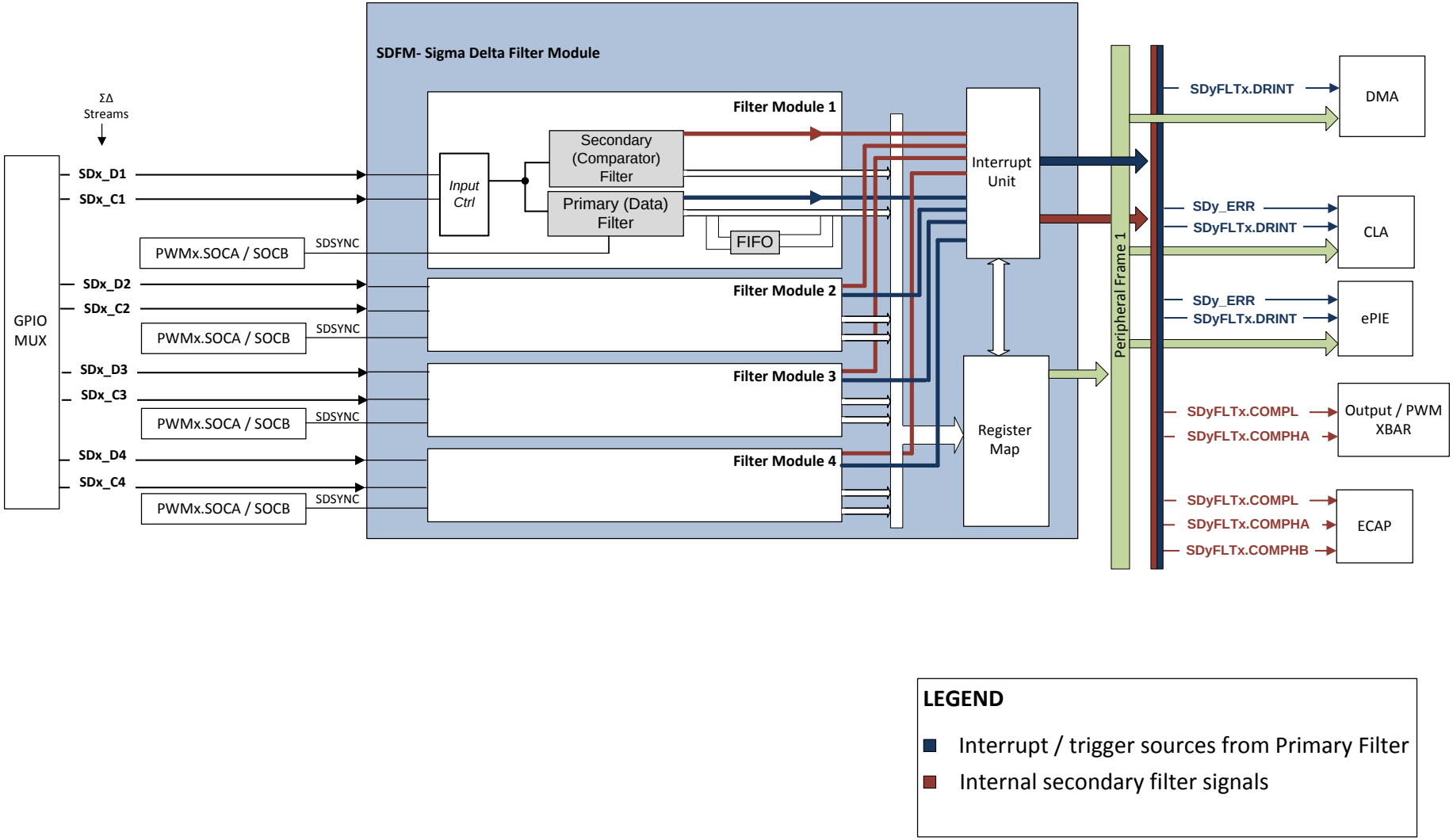
- 8 external pins per SDFM module
  - 4 sigma delta data input pins per SDFM module (SD-Dx, where x = 1 to 4)
  - 4 sigma delta clock input pins per SDFM module (SD-Cx, where x = 1 to 4)
- 4 different configurable modulator clock modes:

  - Mode 0: Modulator clock rate equals modulator data rate
  - Mode 1: Modulator clock rate running at half the modulator data rate
  - Mode 2: Modulator data is Manchester encoded. Modulator clock not required.
  - Mode 3: Modulator clock rate is double that of modulator data rate
- 4 independent configurable secondary filter (comparator) units per SDFM module:
  - 4 different filter type selection (Sinc1/Sinc2/Sincfast/Sinc3) options available
  - Ability to detect over and(or) under value and zero crossing conditions
  - OSR value for comparator filter unit (COSR) programmable from 1 to 32
- 4 independent configurable primary filter (data filter) units per SDFM module:
  - 4 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 and(or) disable individual filter module
  - Ability to synchronize all the 4 independent filters of an SDFM module using the Master Filter Enable (MFE) bit or using PWM signals
- Data filter unit has programmable mode FIFO to reduce interrupt overhead. FIFO has the following features:
  - Primary filter (data filter) has 16 deep x 32 bit FIFO
  - FIFO can interrupt CPU after programmable number of data ready events
  - FIFO Wait-for-Sync feature: Ability to ignore data ready events until PWM synchronization signal (SDSYNC) is received. Once SDSYNC event is received, FIFO is populated on every dat- ready event
  - Data filter output can be represented in either 16 bits or 32 bits
- PWMx.SOCA/SOCB can be configured to serve as SDSYNC source on per data filter channel basis
- PWMs can be used to generate a modulator clock for sigma delta modulators

### 17.1.2 Block Diagram

Figure 17-2 shows the SDFM module block diagram. The SDFM port operation is configured and controlled by the registers listed in Table 17-1.

**Figure 17-2. Sigma Delta Filter Module (SDFM) Block Diagram**



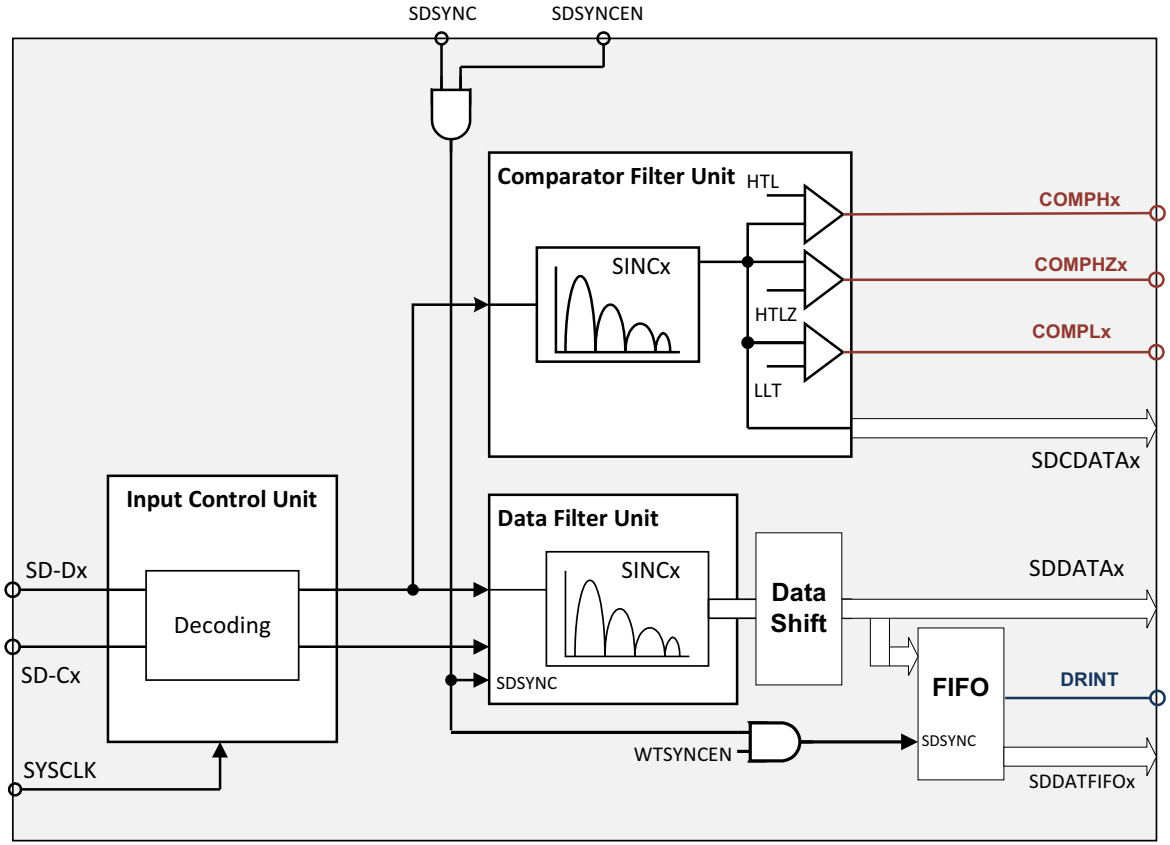

Each SDFM module has four independent filter modules. These filter modules are identical and can be configured independently. Each individual filter module has the following units:

- Input control unit
- Primary filter (data filter) unit

- Secondary filter (comparator filter) unit

Each filter module shown in Figure 17-3 has a primary (data) filter and a secondary (comparator) filter pair which receives the same bit stream. Except for the input bit stream, both the primary and secondary filter are completely independent of each other. Each of these filter modules can be independently configured. So, in a SDFM module, there is a total of four primary filters and four secondary filters.

**Figure 17-3. Block Diagram of One Filter Module**




**LEGEND**

- Interrupt / trigger sources from SDFM
- Internal secondary filter signals

## 17.2 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

Some IO functionality is defined by GPIO register settings independent of this peripheral. For input signals, the GPIO input qualification should be set to asynchronous mode by setting the appropriate GPxQSELn register bits to 11b. The internal pullups can be configured in the GPyPUD register.

See the *GPIO* chapter for more details on GPIO mux and settings.

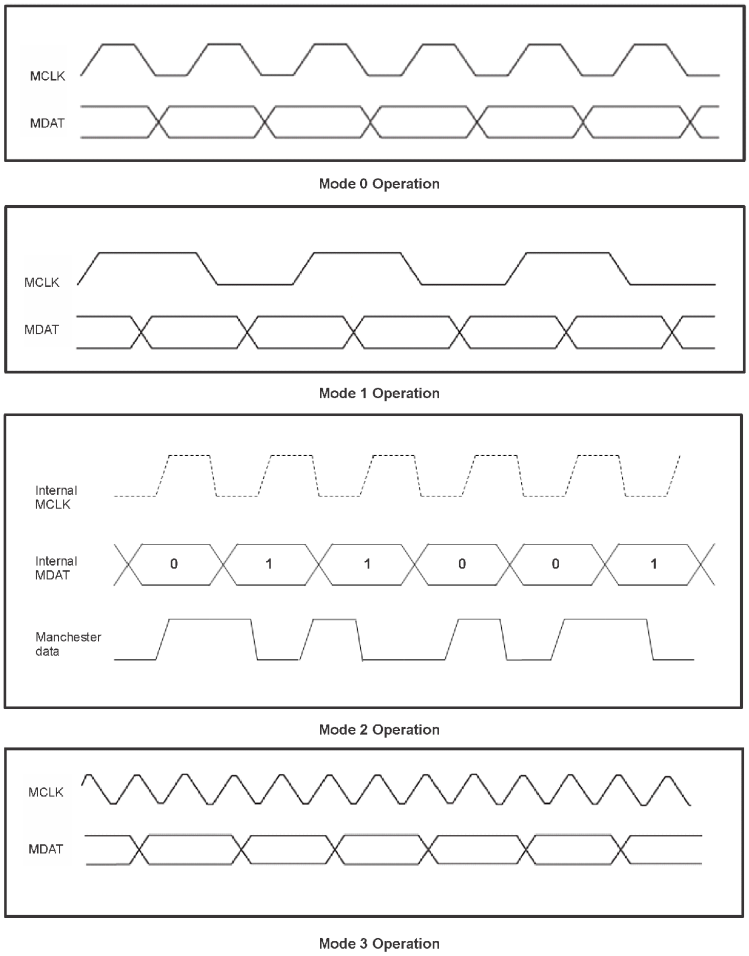
## 17.3 Input Control Unit

The input control unit receives sigma delta modulated data and a sigma delta modulated clock. The modulated data received is captured and passed on to the data filter unit and comparator unit. This unit can be configured to receive the modulated data in four different modes. Table 17-1 and Figure 17-4 show how SDCTLPARMx.MOD bits can be configured in these four different modes.

**Table 17-1. Modulator Clock Modes**

| MODULATOR MODE [MOD] | DESCRIPTION |
|---|---|
| 0 | The modulator clock is running with the modulator data rate. The modulator data is strobed at every rising edge of the modulator clock. |
| 1 | The modulator clock is running with half of the modulator data rate. The modulator data is strobed at every edge of the modulator clock. |
| 2 | The modulator clock is off and the modulator data is Manchester-encoded. |
| 3 | The modulator clock is running with double the modulator data rate. The modulator data is strobed at every other positive modulator clock edge. |

**Figure 17-4. Different Modulator Modes Supported**




When MOD=2, data and modulated clock signals are encoded into modulated data as shown in Mode 2 of Figure 17-4. In this mode, the clock input SD-Cx pin should be left floating. The input control unit performs continuous automatic calibration to achieve optimum decoding performance.

## 17.4 Primary (Data) Filter Unit

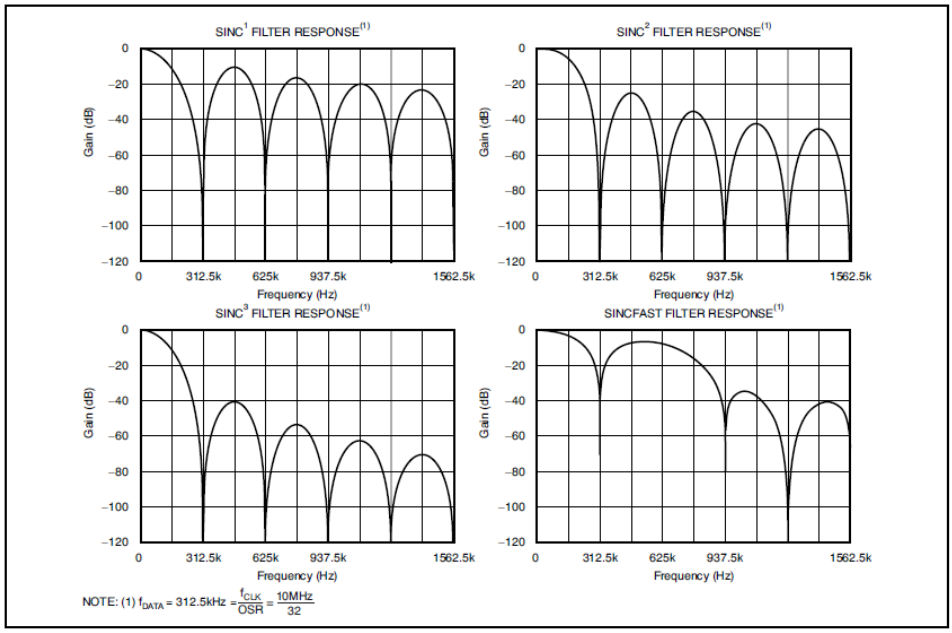
The data filter is a configurable Sinc filter which supports the following filter types: Sinc1, Sinc2, Sinc3 and SincFast. The data filter OSR (DOSR) settings can be configured from 1 to 256 and is independent of the comparator filter. By default, the data filter is disabled and setting of SDDFPARMx.FEN = 1 enables the data filter. The data filter output is represented in 26-bit signed integer in two's complement format. This filter unit translates a low input signal as '-1' and a high input signal as '1'. The resulting calculation gives both positive and negative values for the output of the data filter. Table 17-2 shows the different full-scale values that the data filter can store using different OSRs. Figure 17-5 shows the frequency response of different filter structures when OSR = 32 and when sigma delta modulator frequency is 10 MHz.

Effective resolution of the comparator filter, effective number of bits (ENOB) depends upon filter type, DOSR and sigma-delta modulator used. Typically higher resolution ENOB can be achieved by higher DOSR for a given filter type; however, the tradeoff is increased filter delay. It is important for the user to choose the right sigma-delta modulator by studying the optimal speed versus resolution tradeoff. Please refer to corresponding sigma-delta modulator datasheet to determine the effective resolution for a given comparator filter configuration.

**Table 17-2. 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 |

**Figure 17-5. Frequency Response of Various Primary (Data) Filters**



### 17.4.1 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 = 256

  – SDIPARMx.DR = 0

  The data filter with a 26-bit signed output value can be in the range of –16,777,216 to 16,777,216. But, 16-bit signed output can support values only from –32,768 to 32,767. Therefore, it is required to configure shift control bits (SDDPARMx.SH) to 9 to represent the data filter output correctly in 16-bit format. Table 17-3 shows the configuration settings of shift control bits for different OSR and filter types.

**Table 17-3. Shift Control Bit Configuration Settings**

| OSR | SINC1 | SINC2 | SINCFAST | SINC3 |
|---|---|---|---|---|
| 1 to 31 | 0 | 0 | 0 | 0 |
| 32 to 40 | 0 | 0 | 0 | 1 |
| 41 to 50 | 0 | 0 | 0 | 2 |
| 51 to 63 | 0 | 0 | 0 | 3 |
| 64 to 80 | 0 | 0 | 0 | 4 |
| 81 to 101 | 0 | 0 | 0 | 5 |
| 102 to 127 | 0 | 0 | 0 | 6 |
| 128 to 161 | 0 | 0 | 1 | 7 |
| 162 to 181 | 0 | 0 | 1 | 8 |
| 182 to 203 | 0 | 1 | 2 | 8 |
| 204 to 255 | 0 | 1 | 2 | 9 |
| 256 | 0 | 2 | 3 | 10 |

**WARNING**

**Configuring shift control bits incorrectly will result in getting incorrect 16-bit data filter output.**

### 17.4.2 Data FIFO

Each primary (data) filter channel has a 16-level deep, 32-bit FIFO.

FIFOs can be configured to collect a programmable number of data filter samples before issuing data-ready interrupt. This reduces the number of data-ready interrupts generated and resulting interrupt overhead for managed data flow.

By default, FIFO operation is disabled. FIFOs can be enabled by setting SDFIFOCTLx.FFEN = 1. When FIFO is enabled, each data-ready event from the data filter will populate the FIFO and the status of the FIFO at any given time is updated in the SDFIFOCTLx.SDFFST bit field.

**Setting up FIFO to interrupt after receiving programmable number of data ready events:**

- Enable SDFM FIFO (Set SDFIFOCTLx.FFEN = 1)
- Enable SDFM FIFO interrupt (Set SDFIFOCTLx.FFIEN = 1)
- Configure SDFIFOCTLx.SDFFIL bit field to any value between 0 to 16
- Configure SDFM data ready event to interrupt on FIFO interrupt (SDFFINT) (Set SDFFINTx = 1)
- Select data-Ready interrupt source is SDFFINTx (DRINTx = SDFFINTx) (SDFIFOCTLx.DRINTSEL = 1)

When the SDFIFOCTLx.SDFFST >= SDFIFOCTLx.SDFFIL condition is met, the SDIFLG.SDFFINTx bit is set and an interrupt generated on the DRINTx. SDIFLG.SDFFINTx flag can be cleared by setting the SDIFLGCLR. SDFFINTx bit field.

**Wait for Sync feature:**

The FIFO wait for sync feature can be used to ignore data-ready events from the data filter until the SDSYNC (from PWM) event is triggered.

By default, the Wait for Sync feature is disabled.

**When the wait for sync feature is disabled:**

FIFOs get populated on every data ready event until the FIFO gets full (or) when SDFIFOCTLx.SDFFST >= SDFIFOCTLx.SDFFIL.

**When the wait for sync feature enabled:**

FIFOs will not be get populated on every data ready event until it receives a SDSYNC event. On a SYSYNC event, it sets SDSYNCx.WTSYNFLG = 1 and data ready events from primary filter will start populating FIFO until either the FIFOs get full (or) when SDFIFOCTLx.SDFFST >= SDFIFOCTLx.SDFFIL. WTSYNFLG can be cleared either automatically (or) manually.

**WTSYNFLG automatic clear mode:**

By default, this mode is enabled. When SDSYNCx.WTSCLREN = 1, WTSYNFLG is automatically cleared on SDFFINT event

**WTSYNFLG manual clear mode:**

Setting SDSYNCx.WTSYNCLR = 1 can be used to clear WTSYNFLG manually

When WTSYNFLG = 0, FIFOs contents are frozen and subsequent data ready events don't get populate FIFO until next SDSYNC event.

**Clearing FIFO contents:**

- Disabling FIFO (SDFIFOCTLx.FFEN = 0) will clear FIFO contents
- FIFO contents can be automatically cleared upon receiving SDSYNC event. By default, this feature is disabled and this feature can be enabled by setting FIFO Clear-on-SDSYNC enable (SDSYNCx.FFSYNCCLREN = 1)

  **Note:** The above feature is only enabled when wait for sync feature is enabled (SDSYNCx. WTSYNCEN = 1).
- Disabling data filter module either by clearing master filter enable (MFE) = 0 (or) clearing FEN (Filter Enable) = 0 will also clear FIFO contents

**FIFO debug access behavior:**

Debug access of SDDATFIFOx registers doesn't affect FIFO pointers. In a CPU / CLA / DMA access to SDDATFIFOx register, FIFO read pointers would be advanced to next available entry in FIFO.

### *17.4.3 SDSYNC Event*

Primary (data) filters can be synchronized with respect to the PWM event (called SDSYNC event). The SDSYNC signal from the PWM module is used to reset the DOSR counter. This feature is by default disabled and can be enabled by setting SDDFPARMx.SDSYNCEN = 1.

Each primary filter can be synchronized from any of the available PWMx. SOCA / SOCB signals (see Table 17-4). The SDSYNCx.SDSYNCSEL bits allow the user to configure which PWM signal provides the SDSYNC pulse to primary filter.

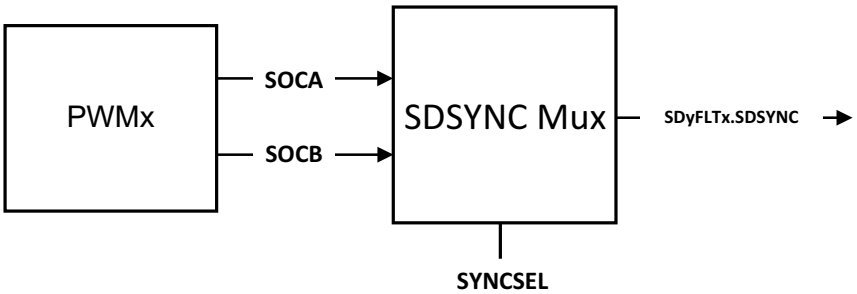
**Table 17-4. SDSYNCx.SYNCSEL**

| SDSNYx.SYNCSEL[1:0] | | | | |
|---|---|---|---|---|
| SYNCSEL[5:2] | 0 | 1 | 2 | 3 |
| 0 | PWM1.SOCA | PWM1.SOCB | Rsvd | Rsvd |
| 1 | PWM2.SOCA | PWM2.SOCB | Rsvd | Rsvd |
| 2 | PWM3.SOCA | PWM3.SOCB | Rsvd | Rsvd |
| 3 | PWM4.SOCA | PWM4.SOCB | Rsvd | Rsvd |
| 4 | PWM5.SOCA | PWM5.SOCB | Rsvd | Rsvd |
| 5 | PWM6.SOCA | PWM6.SOCB | Rsvd | Rsvd |
| 6 | PWM7.SOCA | PWM7.SOCB | Rsvd | Rsvd |
| 7 | PWM8.SOCA | PWM8.SOCB | Rsvd | Rsvd |
| 8 | Rsvd | Rsvd | Rsvd | Rsvd |
| 9 | Rsvd | Rsvd | Rsvd | Rsvd |
| 10 | Rsvd | Rsvd | Rsvd | Rsvd |
| 11 | Rsvd | Rsvd | Rsvd | Rsvd |
| 12 | Rsvd | Rsvd | Rsvd | Rsvd |
| 13 | Rsvd | Rsvd | Rsvd | Rsvd |
| 14 | Rsvd | Rsvd | Rsvd | Rsvd |
| 15 | Rsvd | Rsvd | Rsvd | Rsvd |

**NOTE:** Users MUST ensure that ONLY ONE SDSYNC event will be generated per PWM timer period. Using PWM in up-count (or) down-count mode would automatically ensure that you get ONLY SDSYNC event. But, if you want to use up-down count mode, then make sure that only one SDSYNC event per PWM cycle is generated; otherwise, the filter synchronizer will corrupt SDFM timing by providing two pulses per PWM cycle.

Figure 17-6 shows how the PWM signals are connected to SDFM.

**Figure 17-6. SDSYNC Event**




Because of the inherent architecture of the Sinc filter (Sinc1, Sinc2, SincFast, and Sinc3), the first few samples, depending upon the filter type, will be incorrect. Table 17-5 tabulates the number of incorrect samples after the filter is enabled and configured. This is an expected behavior.

**Table 17-5. 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 |

**Table 17-5. Number of Incorrect Samples Tabulated (continued)**

| Filter type | Number of incorrect samples after the filter is enabled and configured |
|---|---|
| SincFast | The first two samples of the SincFast filter are incorrect |
| Sinc3 | The first two samples of the Sinc3 filter are incorrect |

In the case of the SDFM data filter, each time the filter is enabled or reconfigured, or the filter is reset by the PWM sync pulse, or the filter is reset by SDDFPARMx.FEN, depending upon the filter type, there will be some incorrect samples as mentioned in Table 17-5.

## 17.5 Secondary (Comparator) Filter Unit

Most control systems require protection of the system by tripping the PWM in case the current or voltage sensed by the SDFM goes out of bounds. The primary purpose of the secondary (comparator) filter is to allow the user to monitor input conditions with a fast settling time. This allows the user to trip PWMs to protect the system from potential damage.

**NOTE:** The secondary (comparator) filter cannot be synchronized with respect to the PWM event (SDSYNC event).

The comparator filter is a configurable Sinc filter which supports the following filter types: - Sinc1, Sinc2, Sinc3 and SincFast. The comparator OSR (COSR) settings can be configured from 1 to 32 and is independent of the data filter. By default, the comparator filter is disabled and setting SDCPARMx.CEN = 1 enables the comparator filter. The comparator filter output is represented in 16-bit unsigned format. This filter unit translates a low input signal as ‘0’ and a high input signal as ‘1’. The resulting calculations give only positive values for the output of the comparator filter. Table 17-6 shows the different full-scale values that the comparator filter can store using different OSRs.

**Table 17-6. 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 |

Effective resolution of the comparator filter, effective number of bits (ENOB,) depends upon filter type, COSR, and sigma-delta modulator used. Typically higher resolution (ENOB) can be achieved by higher COSR for a given filter type but the tradeoff increases filter delay. It is important for the user to choose the right sigma-delta modulator by studying the optimal speed versus resolution tradeoff. Please refer to the corresponding sigma-delta modulator datasheet to determine the effective resolution for a given comparator filter configuration.

Output of the comparator filter is memory-mapped and can be read in the SDCDATAx register. This new register, SDCDATAx, is updated every COSR number of SD-Cx cycles. Please refer to Section 17.6 to understand how to calculate comparator filter (data rate) and latency of comparator filter. The output of the comparator filter is compared with three programmable threshold levels mentioned below.

**Higher Threshold (A) comparator**

- High Threshold (A) comparator can be used to detect over-value condition
- When SDCPARMx.IEH = 1, it enables the comparator to generate Higher Threshold (A) event. Higher Threshold (A) event (COMPHx) (or) IEHx is triggered when
  - Comparator data > = SDCMPHx.HLT

**Higher Threshold (B) comparator**

- High Threshold (B) comparator can be used in conjunction with ECAP to measure the frequency / duty

cycle of Threshold crossing events

- When SDCPARMx.HZEN = 1, it enables the comparator to generate Higher Threshold (B) event (COMPHZx) (or) HZx is triggered when
  - Comparator data > = SDCMPHZx.HLTZ

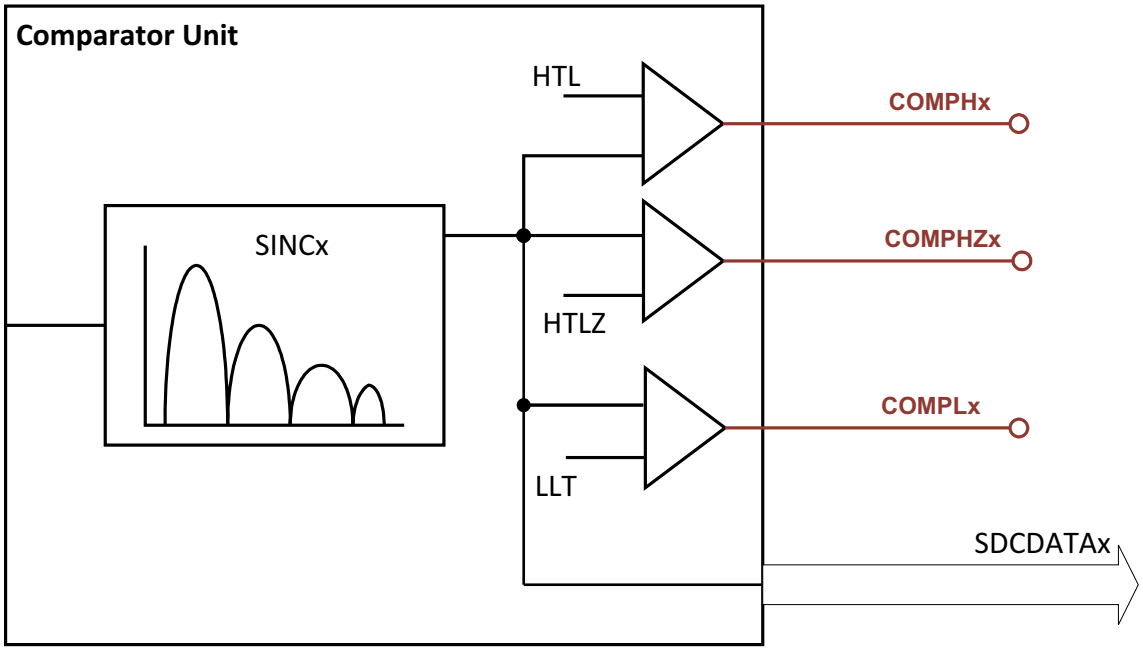
***Example***

To measure the frequency / duty cycle of zero crossing event, the user needs to configure SDCMPHZx.HLTZ = 0x4000 and enable High level (B) Threshold crossing (SDCPARMx.HZEN = 1). The ECCTL0.INPUTSEL register bit field should also be configured to pass on the COMPHZx signal to ECAP for frequency / duty cycle measurement.

**Lower Threshold comparator**

- Lower Threshold comparator can be used to detect under value condition.
- When SDCPARMx.IEL= 1, it enables the comparator to generate Lower Threshold event. Lower Threshold event (COMPLx) (or) IELx is triggered when
  - Comparator data <= SDCMPLx.LLT

**Figure 17-7. Comparator Unit Structure**




In order to achieve the maximum value, the delta-sigma modulator is operated at absolute maximum positive or negative full-scale, which is outside of the recommended full-scale range of 80% of most delta-sigma modulators.

## 17.6 Data Rate and Latency of the Sinc Filter

The order of the sinc filter is shown in Table 17-7.

**Table 17-7. Order of Sinc Filter**

| filter Type | Order of Sinc Filter |
|---|---|
| SincFast | 3 |
| Sinc1 | 1 |
| Sinc2 | 2 |
| Sinc3 | 3 |

The data rate of the sinc filter (filter throughput) represented in samples/sec can be calculated by the formula shown here:

$$Data\ rate\ of\ Sinc\ filter = \frac{Modulator\ data\ rate}{OSR}$$

The latency of the sinc filter represented in secs is defined as the amount of time taken by a sinc filter type to deliver the correct filtered output upon initiation. For a given filter type, latency can be calculated as shown in this equation:

$$Latency\ of\ Sinc\ filter = \frac{Order\ of\ Sinc\ filter}{Data\ rate\ of\ Sinc\ filter}$$

***Example configuration:***

| | |
|---|---|
| Sinc filter type | = sinc3 |
| Modulator data rate | = 10 MHz |
| OSR | = 256 |
| Data rate of Sinc Filter | = 10 MHz / 256 = 39.1 K samples / sec |
| Sinc filter latency | = 76.8 µs |

| | |
|---|---|
| Sinc filter type | = sinc2 |
| Modulator data rate | = 10 MHz |
| OSR | = 256 |
| Data rate of Sinc Filter | = 10 MHz / 256 = 39.1 K samples / sec |
| Sinc filter latency | =51.2 µs |

## 17.7 Interrupt Unit

Figure 17-8 shows the structure of the interrupt unit. Each SDFM module can generate a CPU interrupt. An SDFM interrupt can be triggered by any of these 16 interrupt sources.

**Four Lower Threshold interrupts (IELx)**

When the secondary filter output is less than or equal to the lower threshold (SDCDATAx ≤ SDCMPLx.LLT), it sets the SDIFLG.IFLx bit and generates the SD_ERR interrupt. This flag can be cleared by setting the SDIFLGCLR.IFLx bit and if the interrupt source is no longer active.

***Enable Lower Threshold Interrupt (IELx)***

On top of filter configurations, the below configurations have to be made:

- Enable secondary filter lower threshold interrupt (SDCPARMx.IEL = 1) and
- Enable master interrupt enable (SDCTL.MIE = 1)

**Four Higher Threshold interrupts (A) (IEHx)**

When the secondary filter output is greater than (or) equal higher threshold (SDCDATAx ≥ SDCMPHx.HLT), it sets SDIFLG.IFHx bit and generates SD_ERR interrupt. This flag can be cleared by setting SDIFLGCLR.IFHx bit and if the interrupt source is no longer active.

***Enable Higher Threshold interrupt (A) (IEHx)***

On top of filter configurations, the below configurations have to be made:

- Enable secondary filter higher threshold interrupt (SDCPARMx.IEH = 1) and
- Enable master interrupt enable (SDCTL.MIE = 1)

**Four Modulator Failure (MFx)**

When the modulator clock (SD-Cx) from SD modulator goes missing, it sets SDIFLG.MFx bit and generates SD_ERR interrupt. Modulator clock is considered missing if SD-Cx does not toggle for 64-system clock SYSCLK.

***Enable Modulator Failure (MFx)***

- Enable modulator clock failure interrupt source (SDCPARMx.MFIE = 1) and
- Enable master interrupt enable (SDCTL.MIE = 1)

**Four FIFO overflow (SDFFOVFx)**

The number of data available in FIFO at any given point can be tracked in SDFIFOCTLx.SDFFST. If the number of words received in FIFO is greater than FIFO depth (16), new primary filter data will be lost and the SDIFLG.SDFFOVFx bit is set and SD_ERR interrupt is generated. The SDIFLG.SDFFOVFx flag can be cleared by setting the SDIFLGCLR.SDFFOVFx bit field.

***Enable FIFO overflow interrupt:***

On top of data filter and FIFO configurations, the below configurations have to be made:

- Enable SDFM FIFO (Set SDFIFOCTLx.FFEN = 1)
- Enable SDFM FIFO overflow interrupt (Set SDFIFOCTLx.OVFIEN = 1) and
- Enable Master interrupt enable (Set SDCTL.MIE = 1)

**Four filter data acknowledge (AFx)**

When the primary filter is ready with a new filter data, it sets SDIFLG.AFx bit and generates DRINTx interrupt. This flag can be cleared by setting SDIFLGCLR.AFx.

***Enable filter data acknowledge***

- Enable individual filter interrupts (SDDFPARMx. AE = 1) and
- Select data-ready interrupt source AFx (DRINTx = AFx) (SDFIFOCTLx.DRINTSEL = 0)

**Four FIFO Data ready interrupt (SDFFINTx)**

The number of data available in FIFO at any given point can be tracked in SDFIFOCTLx.SDFFST.

If the SDFIFOCTLx.SDFFST >= SDFIFOCTLx.SDFFIL condition is met, the SDIFLG.SDFFINTx bit is set and an interrupt is generated on the DRINTx. SDIFLG.DFFINTx flag can be cleared by setting the SDIFLGCLR. SDFFINTx bit field.

Table 17-8 shows how the DRINTx output is selected.

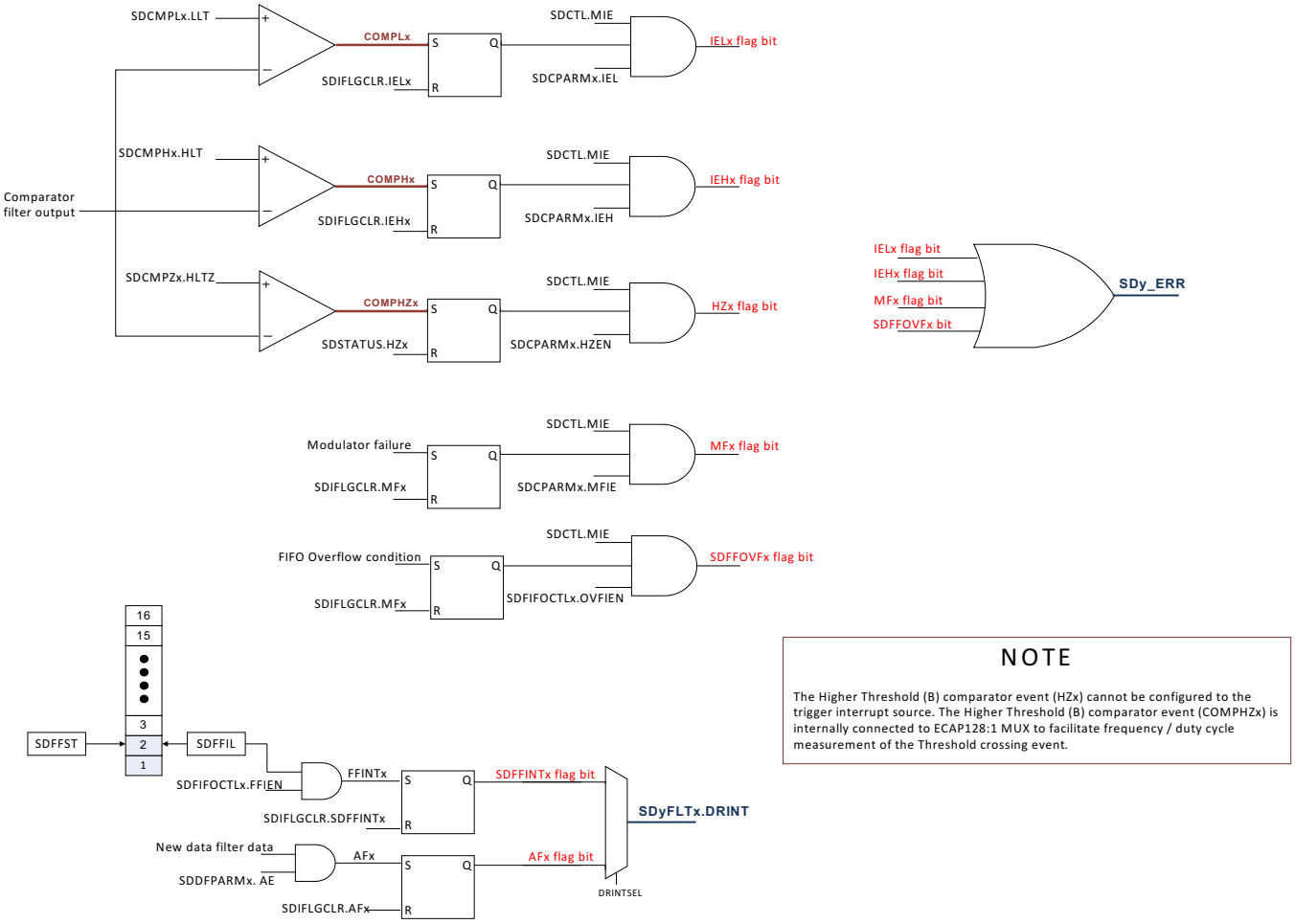
***Enable FIFO data ready interrupt***

- Enable SDFM FIFO (Set SDFIFOCTLx.FFEN = 1) and
- Enable SDFM FIFO interrupt (Set SDFIFOCTLx.FFIEN = 1)
- Select data-Ready interrupt source is SDFFINTx (DRINTx = SDFFINTx) (SDFIFOCTLx.DRINTSEL = 1)

**Table 17-8. DRINTx (Data-Ready Interrupt) Output Selection**

| DRINTSEL | AE | FFIEN | FFEN | DRINTx |
|---|---|---|---|---|
| 0 | 0 | x | X | 0 |
| 0 | 1 | x | X | AFx |
| 1 | x | 0 | X | 0 |
| 1 | x | x | 0 | 0 |
| 1 | x | 1 | 1 | SDFFINTx |

**Figure 17-8. SDFM Interrupt Unit**




**NOTE**

The Higher Threshold (B) comparator event (HZx) cannot be configured to the trigger interrupt source. The Higher Threshold (B) comparator event (COMPHZx) is internally connected to ECAP128:1 MUX to facilitate frequency / duty cycle measurement of the Threshold crossing event.

## 17.8 Registers

### 17.8.1 Sigma Delta Filter Base Addresses

**Table 17-9. SDFM Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| Sdfm1Regs | SDFM_REGS | 0x0000_5E00 | 0x0000_5E7F |

### 17.8.1.1 SDFM_REGS Registers

Table 17-10 lists the memory-mapped registers for the SDFM_REGS. All register offset addresses not listed in Table 17-10 should be considered as reserved locations and the register contents should not be modified.

**Table 17-10. SDFM_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | SDIFLG | SD Interrupt Flag Register | | Go |
| 2h | SDIFLGCLR | SD Interrupt Flag Clear Register | | Go |
| 4h | SDCTL | SD Control Register | EALLOW | Go |
| 6h | SDMFILEN | SD Master Filter Enable | EALLOW | Go |
| 7h | SDSTATUS | SD Status Register | | Go |
| 10h | SDCTLPARM1 | Control Parameter Register for Ch1 | EALLOW | Go |
| 11h | SDDFPARM1 | Data Filter Parameter Register for Ch1 | EALLOW | Go |
| 12h | SDDPARM1 | Data 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 Filter Parameter Register for Ch1 | EALLOW | Go |
| 16h | SDDATA1 | Data Filter Data Register (16 or 32bit) for Ch1 | | Go |
| 18h | SDDATFIFO1 | Filter Data FIFO Output(32b) for Ch1 | | Go |
| 1Ah | SDCDATA1 | Comparator Filter Data Register (16b) for Ch1 | | Go |
| 1Ch | SDCMPHZ1 | High-level (Z) Threshold Register for Ch1 | EALLOW | Go |
| 1Dh | SDFIFOCTL1 | FIFO Control Register for Ch1 | EALLOW | Go |
| 1Eh | SDSYNC1 | SD Filter Sync control for Ch1 | EALLOW | Go |
| 20h | SDCTLPARM2 | Control Parameter Register for Ch2 | EALLOW | Go |
| 21h | SDDFPARM2 | Data Filter Parameter Register for Ch2 | EALLOW | Go |
| 22h | SDDPARM2 | Data 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 Filter Parameter Register for Ch2 | EALLOW | Go |
| 26h | SDDATA2 | Data Filter Data Register (16 or 32bit) for Ch2 | | Go |
| 28h | SDDATFIFO2 | Filter Data FIFO Output(32b) for Ch2 | | Go |
| 2Ah | SDCDATA2 | Comparator Filter Data Register (16b) for Ch2 | | Go |
| 2Ch | SDCMPHZ2 | High-level (Z) Threshold Register for Ch2 | EALLOW | Go |
| 2Dh | SDFIFOCTL2 | FIFO Control Register for Ch2 | EALLOW | Go |
| 2Eh | SDSYNC2 | SD Filter Sync control for Ch2 | EALLOW | Go |
| 30h | SDCTLPARM3 | Control Parameter Register for Ch3 | EALLOW | Go |
| 31h | SDDFPARM3 | Data Filter Parameter Register for Ch3 | EALLOW | Go |
| 32h | SDDPARM3 | Data 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 Filter Parameter Register for Ch3 | EALLOW | Go |
| 36h | SDDATA3 | Data Filter Data Register (16 or 32bit) for Ch3 | | Go |
| 38h | SDDATFIFO3 | Filter Data FIFO Output(32b) for Ch3 | | Go |
| 3Ah | SDCDATA3 | Comparator Filter Data Register (16b) for Ch3 | | Go |
| 3Ch | SDCMPHZ3 | High-level (Z) Threshold Register for Ch3 | EALLOW | Go |
| 3Dh | SDFIFOCTL3 | FIFO Control Register for Ch3 | EALLOW | Go |
| 3Eh | SDSYNC3 | SD Filter Sync control for Ch3 | EALLOW | Go |
| 40h | SDCTLPARM4 | Control Parameter Register for Ch4 | EALLOW | Go |
| 41h | SDDFPARM4 | Data Filter Parameter Register for Ch4 | EALLOW | Go |

**Table 17-10. SDFM_REGS Registers (continued)**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 42h | SDDPARM4 | Data 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 Filter Parameter Register for Ch4 | EALLOW | Go |
| 46h | SDDATA4 | Data Filter Data Register (16 or 32bit) for Ch4 | | Go |
| 48h | SDDATFIFO4 | Filter Data FIFO Output(32b) for Ch4 | | Go |
| 4Ah | SDCDATA4 | Comparator Filter Data Register (16b) for Ch4 | | Go |
| 4Ch | SDCMPHZ4 | High-level (Z) Threshold Register for Ch4 | EALLOW | Go |
| 4Dh | SDFIFOCTL4 | FIFO Control Register for Ch4 | EALLOW | Go |
| 4Eh | SDSYNC4 | SD Filter Sync control for Ch4 | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 17-11 shows the codes that are used for access types in this section.

**Table 17-11. SDFM_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| W=1 | 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. |

#### 17.8.1.1.1 SDIFLG Register (Offset = 0h) [reset = 0h]

SDIFLG is shown in Figure 17-9 and described in Table 17-12.

Return to Summary Table.

SD Interrupt Flag Register

**Figure 17-9. SDIFLG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MIF | RESERVED | | | | | | |
| R-0h | R-0h | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| SDFFINT4 | SDFFINT3 | SDFFINT2 | SDFFINT1 | SDFFOVF4 | SDFFOVF3 | SDFFOVF2 | SDFFOVF1 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| AF4 | AF3 | AF2 | AF1 | MF4 | MF3 | MF2 | MF1 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IFL4 | IFH4 | IFL3 | IFH3 | IFL2 | IFH2 | IFL1 | IFH1 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 17-12. SDIFLG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MIF | R | 0h | Set whenever any "error" interrupt (MF1-4,IFL1-4,IFH1-4,SDFFOVF1-4) is active<br>Reset type: SYSRSn |
| 30-24 | RESERVED | R | 0h | Reserved |
| 23 | SDFFINT4 | R | 0h | SDFIFO data ready interrupt for Ch4<br>Reset type: SYSRSn |
| 22 | SDFFINT3 | R | 0h | SDFIFO data ready interrupt for Ch3<br>Reset type: SYSRSn |
| 21 | SDFFINT2 | R | 0h | SDFIFO data ready interrupt for Ch2<br>Reset type: SYSRSn |
| 20 | SDFFINT1 | R | 0h | SDFIFO data ready interrupt for Ch1<br>0: SDFIFO data ready interrupt has NOT occurred<br>1: SDFIFO data ready interrupt has occurred<br>Reset type: SYSRSn |
| 19 | SDFFOVF4 | R | 0h | FIFO Overflow Flag for Ch4<br>Reset type: SYSRSn |
| 18 | SDFFOVF3 | R | 0h | FIFO Overflow Flag for Ch3<br>Reset type: SYSRSn |
| 17 | SDFFOVF2 | R | 0h | FIFO Overflow Flag for Ch2<br>Reset type: SYSRSn |
| 16 | SDFFOVF1 | R | 0h | FIFO Overflow Flag for Ch1<br>0 - FIFO has not overflowed<br>1 - FIFO overflowed. # words received in FIFO > FIFO depth (16), NEW word is lost<br>Reset type: SYSRSn |

**Table 17-12. SDIFLG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | AF4 | R | 0h | Acknowledge flag for Filter 4<br>0: No new data available for Filter (in non-FIFO mode)<br>1: New data available for Filter (in non-FIFO mode)<br>Reset type: SYSRSn |
| 14 | AF3 | R | 0h | Acknowledge flag for Filter 3<br>0: No new data available for Filter (in non-FIFO mode)<br>1: New data available for Filter (in non-FIFO mode)<br>Reset type: SYSRSn |
| 13 | AF2 | R | 0h | Acknowledge flag for Filter 2<br>0: No new data available for Filter (in non-FIFO mode)<br>1: New data available for Filter (in non-FIFO mode)<br>Reset type: SYSRSn |
| 12 | AF1 | R | 0h | Acknowledge flag for Filter 1<br>0: No new data available for Filter (in non-FIFO mode)<br>1: New data available for Filter (in non-FIFO mode)<br>Reset type: SYSRSn |
| 11 | MF4 | R | 0h | Modulator Failure for Filter 4<br>0: Modulator is operating normally for Filter<br>1: Modulator failure for Filter<br>Reset type: SYSRSn |
| 10 | MF3 | R | 0h | Modulator Failure for Filter 3<br>0: Modulator is operating normally for Filter<br>1: Modulator failure for Filter<br>Reset type: SYSRSn |
| 9 | MF2 | R | 0h | Modulator Failure for Filter 2<br>0: Modulator is operating normally for Filter<br>1: Modulator failure for Filter<br>Reset type: SYSRSn |
| 8 | MF1 | R | 0h | Modulator Failure for Filter 1<br>0: Modulator is operating normally for Filter<br>1: Modulator failure for Filter<br>Reset type: SYSRSn |
| 7 | IFL4 | R | 0h | Low-level Interrupt flag for Ch4<br>0: Comparator filter output > SDCMPL4.LLT<br>1: Comparator filter output <= SDCMPL4.LLT<br>Reset type: SYSRSn |
| 6 | IFH4 | R | 0h | High-level Interrupt flag for Ch4<br>0: Comparator filter output < SDCMPH4.HLT<br>1: Comparator filter output >= SDCMPH4.HLT<br>Reset type: SYSRSn |

**Table 17-12. SDIFLG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5 | IFL3 | R | 0h | Low-level Interrupt flag for Ch3<br>0: Comparator filter output > SDCMPL3.LLT<br>1: Comparator filter output <= SDCMPL3.LLT<br>Reset type: SYSRSn |
| 4 | IFH3 | R | 0h | High-level Interrupt flag for Ch3<br>0: Comparator filter output < SDCMPH3.HLT<br>1: Comparator filter output >= SDCMPH3.HLT<br>Reset type: SYSRSn |
| 3 | IFL2 | R | 0h | Low-level Interrupt flag for Ch2<br>0: Comparator filter output > SDCMPL2.LLT<br>1: Comparator filter output <= SDCMPL2.LLT<br>Reset type: SYSRSn |
| 2 | IFH2 | R | 0h | High-level Interrupt flag for Ch2<br>0: Comparator filter output < SDCMPH2.HLT<br>1: Comparator filter output >= SDCMPH2.HLT<br>Reset type: SYSRSn |
| 1 | IFL1 | R | 0h | Low-level Interrupt flag for Ch1<br>0: Comparator filter output > SDCMPL1.LLT<br>1: Comparator filter output <= SDCMPL1.LLT<br>Reset type: SYSRSn |
| 0 | IFH1 | R | 0h | High-level Interrupt flag for Ch1<br>0: Comparator filter output < SDCMPH1.HLT<br>1: Comparator filter output >= SDCMPH1.HLT<br>Reset type: SYSRSn |

#### 17.8.1.1.2 SDIFLGCLR Register (Offset = 2h) [reset = 0h]

SDIFLGCLR is shown in Figure 17-10 and described in Table 17-13.

Return to Summary Table.

SD Module Interrupt Flag Clear Bits:

Writing a "1" will clear the respective flag bit in the SDIFLG register.
Writes of "0" are ignored.

Note: If user writes a "1" to clear a bit on the same cycle that the hardware is trying to set the bit to "1", then hardware has priority and the bit will not be cleared.

**Figure 17-10. SDIFLGCLR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MIF | RESERVED | | | | | | |
| R=0/W=1-0h | R-0h | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| SDFFINT4 | SDFFINT3 | SDFFINT2 | SDFFINT1 | SDFFOVF4 | SDFFOVF3 | SDFFOVF2 | SDFFOVF1 |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| AF4 | AF3 | AF2 | AF1 | MF4 | MF3 | MF2 | MF1 |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IFL4 | IFH4 | IFL3 | IFH3 | IFL2 | IFH2 | IFL1 | IFH1 |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 17-13. SDIFLGCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MIF | R=0/W=1 | 0h | Flag-clear bit for SDFM Master Interrupt flag.<br>Writing a 1 to clear MIF flag in SDIFLG register<br>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 |
| 30-24 | RESERVED | R | 0h | Reserved |
| 23 | SDFFINT4 | R=0/W=1 | 0h | SDFIFO data ready Interrupt flag-clear bit for Ch4<br>Reset type: SYSRSn |
| 22 | SDFFINT3 | R=0/W=1 | 0h | SDFIFO data ready Interrupt flag-clear bit for Ch3<br>Reset type: SYSRSn |
| 21 | SDFFINT2 | R=0/W=1 | 0h | SDFIFO data ready Interrupt flag-clear bit for Ch2<br>Reset type: SYSRSn |
| 20 | SDFFINT1 | R=0/W=1 | 0h | SDFIFO data ready Interrupt flag-clear bit for Ch1<br>Reset type: SYSRSn |
| 19 | SDFFOVF4 | R=0/W=1 | 0h | SDFIFO overflow clear Ch4<br>Reset type: SYSRSn |
| 18 | SDFFOVF3 | R=0/W=1 | 0h | SDFIFO overflow clear Ch3<br>Reset type: SYSRSn |
| 17 | SDFFOVF2 | R=0/W=1 | 0h | SDFIFO overflow clear Ch2<br>Reset type: SYSRSn |
| 16 | SDFFOVF1 | R=0/W=1 | 0h | SDFIFO overflow clear Ch1<br>Reset type: SYSRSn |

**Table 17-13. SDIFLGCLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | AF4 | R=0/W=1 | 0h | Flag-clear bit for Acknowledge flag for Filter 4<br>Reset type: SYSRSn |
| 14 | AF3 | R=0/W=1 | 0h | Flag Clear bit for AF3<br>Reset type: SYSRSn |
| 13 | AF2 | R=0/W=1 | 0h | Flag Clear bit for AF2<br>Reset type: SYSRSn |
| 12 | AF1 | R=0/W=1 | 0h | Flag Clear bit for AF1<br>Reset type: SYSRSn |
| 11 | MF4 | R=0/W=1 | 0h | Flag Clear bit for MF4<br>Reset type: SYSRSn |
| 10 | MF3 | R=0/W=1 | 0h | Flag Clear bit for MF3<br>Reset type: SYSRSn |
| 9 | MF2 | R=0/W=1 | 0h | Flag Clear bit for MF2<br>Reset type: SYSRSn |
| 8 | MF1 | R=0/W=1 | 0h | Flag Clear bit for MF1<br>Reset type: SYSRSn |
| 7 | IFL4 | R=0/W=1 | 0h | Flag Clear bit for IFL4<br>Reset type: SYSRSn |
| 6 | IFH4 | R=0/W=1 | 0h | Flag Clear bit for IFH4<br>Reset type: SYSRSn |
| 5 | IFL3 | R=0/W=1 | 0h | Flag Clear bit for IFL3<br>Reset type: SYSRSn |
| 4 | IFH3 | R=0/W=1 | 0h | Flag Clear bit for IFH3<br>Reset type: SYSRSn |
| 3 | IFL2 | R=0/W=1 | 0h | Flag Clear bit for IFL2<br>Reset type: SYSRSn |
| 2 | IFH2 | R=0/W=1 | 0h | Flag Clear bit for IFH2<br>Reset type: SYSRSn |
| 1 | IFL1 | R=0/W=1 | 0h | Flag Clear bit for IFL1<br>Reset type: SYSRSn |
| 0 | IFH1 | R=0/W=1 | 0h | Flag Clear bit for IFH1<br>Reset type: SYSRSn |

#### 17.8.1.1.3 SDCTL Register (Offset = 4h) [reset = 0h]

SDCTL is shown in Figure 17-11 and described in Table 17-14.

Return to Summary Table.

SD Control Register

**Figure 17-11. SDCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | MIE | RESERVED | | | | |
| R-0h | R-0h | R/W-0h | R-0h | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | HZ4 | HZ3 | HZ2 | HZ1 |
| R-0h | | | | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 17-14. SDCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14 | RESERVED | R | 0h | Reserved |
| 13 | MIE | R/W | 0h | Master SDy_ERR interrupt enable<br>0: SDy_ERR Interrupt and interrupt flags are disabled<br>1: SDy_ERR Interrupt and interrupt flags are enabled<br>Reset type: SYSRSn |
| 12-4 | RESERVED | R | 0h | Reserved |
| 3 | HZ4 | R=0/W=1 | 0h | Flag Clear bit for HZ4<br>Reset type: SYSRSn |
| 2 | HZ3 | R=0/W=1 | 0h | Flag Clear bit for HZ3<br>Reset type: SYSRSn |
| 1 | HZ2 | R=0/W=1 | 0h | Flag Clear bit for HZ2<br>Reset type: SYSRSn |
| 0 | HZ1 | R=0/W=1 | 0h | Flag Clear bit for HZ1<br>Reset type: SYSRSn |

#### 17.8.1.1.4 SDMFILEN Register (Offset = 6h) [reset = 0h]

SDMFILEN is shown in Figure 17-12 and described in Table 17-15.

Return to Summary Table.

SD Master Filter Enable

**Figure 17-12. SDMFILEN Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | RESERVED | MFE | RESERVED | RESERVED | RESERVED |
| R-0h | | | R-0h | R/W-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | RESERVED | | | RESERVED | | | |
| R-0h | R-0h | | | R-0h | | | |

**Table 17-15. SDMFILEN Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12 | RESERVED | R | 0h | Reserved |
| 11 | MFE | R/W | 0h | Master Filter Enable<br>0: All the four data filter units of SDFM module are disabled. All FIFOs<br>are cleared<br>1: Data filter units can be enabled if bit FEN is '1'.<br>Reset type: SYSRSn |
| 10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |
| 8-7 | RESERVED | R | 0h | Reserved |
| 6-4 | RESERVED | R | 0h | Reserved |
| 3-0 | RESERVED | R | 0h | Reserved |

#### 17.8.1.1.5 SDSTATUS Register (Offset = 7h) [reset = 0h]

SDSTATUS is shown in Figure 17-13 and described in Table 17-16.

Return to Summary Table.

SD Status Register

**Figure 17-13. SDSTATUS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | HZ4 | HZ3 | HZ2 | HZ1 |
| R-0h | | | | R-0h | R-0h | R-0h | R-0h |

**Table 17-16. SDSTATUS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14 | RESERVED | R | 0h | Reserved |
| 13 | RESERVED | R | 0h | Reserved |
| 12 | RESERVED | R | 0h | Reserved |
| 11 | RESERVED | R | 0h | Reserved |
| 10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 7-4 | RESERVED | R | 0h | Reserved |
| 3 | HZ4 | R | 0h | High-level Threshold crossing (Z) flag Ch4<br>Primarily intended for detecting "zero"-crossing events. Unlike the primary comparator IFHx flag, it does not have the ability to generate an interrupt.<br>0: Comparator filter output < SDCMPHZ4.HLTZ<br>1: Comparator filter output >= SDCMPHZ4.HLTZ<br>Reset type: SYSRSn |
| 2 | HZ3 | R | 0h | High-level Threshold crossing (Z) flag Ch3<br>Primarily intended for detecting "zero"-crossing events. Unlike the primary comparator IFHx flag, it does not have the ability to generate an interrupt.<br>0: Comparator filter output < SDCMPHZ3.HLTZ<br>1: Comparator filter output >= SDCMPHZ3.HLTZ<br>Reset type: SYSRSn |
| 1 | HZ2 | R | 0h | High-level Threshold crossing (Z) flag Ch2<br>Primarily intended for detecting "zero"-crossing events. Unlike the primary comparator IFHx flag, it does not have the ability to generate an interrupt.<br>0: Comparator filter output < SDCMPHZ2.HLTZ<br>1: Comparator filter output >= SDCMPHZ2.HLTZ<br>Reset type: SYSRSn |

**Table 17-16. SDSTATUS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | HZ1 | R | 0h | High-level Threshold crossing (Z) flag Ch1<br><br>Primarily intended for detecting "zero"-crossing events. Unlike the primary comparator IFHx flag, it does not have the ability to generate an interrupt.<br><br>0: Comparator filter output < SDCMPHZ1.HLTZ<br><br>1: Comparator filter output >= SDCMPHZ1.HLTZ<br>Reset type: SYSRSn |

#### 17.8.1.1.6 SDCTLPARM1 Register (Offset = 10h) [reset = 0h]

SDCTLPARM1 is shown in Figure 17-14 and described in Table 17-17.

Return to Summary Table.

Control Parameter Register for Ch1

**Figure 17-14. 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-0h | R-0h | R-0h | R/W-0h | |

**Table 17-17. SDCTLPARM1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1-0 | MOD | R/W | 0h | Modulator clock modes<br>MODE 0: Modulator clock running at 1x data rate<br>MODE 1: modulator clock running at 1/2 data rate (dbl-edge clocking)<br>MODE 2: modulator clock absent (Manchester encoded data)<br>MODE 3: modulator clock running at 2x data rate<br>Reset type: SYSRSn |

#### 17.8.1.1.7 SDDFPARM1 Register (Offset = 11h) [reset = 0h]

SDDFPARM1 is shown in Figure 17-15 and described in Table 17-18.

Return to Summary Table.

Data Filter Parameter Register for Ch1

**Figure 17-15. 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 17-18. SDDFPARM1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12 | SDSYNCEN | R/W | 0h | PWM synchronization (SDSYNC) of data filter<br>0: PWM synchronization of data filter is disabled<br>1: PWM synchronization of data filter is enabled<br>Note: SDSYNCx.SYNCSEL bits define which PWM signal is used to synchronize PWMs<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 | Data filter 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 data filter is disabled and no data is produced<br>1: The data filter is enabled and data are produced in the data filter<br>Note: When filter is disabled, DOSR counter held in reset, filter data erased. Also resets FIFO pointers and clears the FIFO<br>Reset type: SYSRSn |
| 7-0 | DOSR | R/W | 0h | Data filter Oversampling ratio<br>The actual oversampling ratio of data filter is DOSR + 1<br>These bits set the oversampling ratio of the data filter.<br>0x0FF represents an oversampling ratio of 256.<br>Reset type: SYSRSn |

#### 17.8.1.1.8 SDDPARM1 Register (Offset = 12h) [reset = 0h]

SDDPARM1 is shown in Figure 17-16 and described in Table 17-19.

Return to Summary Table.

Data Parameter Register for Ch1

**Figure 17-16. SDDPARM1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SH | | | | | DR | RESERVED | |
| R/W-0h | | | | | R/W-0h | R-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |

**Table 17-19. 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 filter Data representation<br>0: Data stored in 16b 2's complement<br>1: Data stored in 32b 2's complement<br>Reset type: SYSRSn |
| 9-0 | RESERVED | R | 0h | Reserved |

#### 17.8.1.1.9 SDCMPH1 Register (Offset = 13h) [reset = 7FFFh]

SDCMPH1 is shown in Figure 17-17 and described in Table 17-20.

Return to Summary Table.

High-level Threshold Register for Ch1

**Figure 17-17. SDCMPH1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | HLT | | | | | | |
| R-0h | R/W-7FFFh | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HLT | | | | | | | |
| R/W-7FFFh | | | | | | | |

**Table 17-20. SDCMPH1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14-0 | HLT | R/W | 7FFFh | Unsigned high-level threshold for the comparator filter output.<br>Reset type: SYSRSn |

#### 17.8.1.1.10 SDCMPL1 Register (Offset = 14h) [reset = 0h]

SDCMPL1 is shown in Figure 17-18 and described in Table 17-21.

Return to Summary Table.

Low-level Threshold Register for Ch1

**Figure 17-18. SDCMPL1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | LLT | | | | | | |
| R-0h | R/W-0h | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LLT | | | | | | | |
| R/W-0h | | | | | | | |

**Table 17-21. SDCMPL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14-0 | LLT | R/W | 0h | Unsigned low-level threshold for the comparator filter output.<br>Reset type: SYSRSn |

#### 17.8.1.1.11 SDCPARM1 Register (Offset = 15h) [reset = 0h]

SDCPARM1 is shown in Figure 17-19 and described in Table 17-22.

Return to Summary Table.

Comparator Filter Parameter Register for Ch1

**Figure 17-19. SDCPARM1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | CEN | RESERVED | | HZEN | MFIE | CS1_CS0 |
| R-0h | | R/W-0h | R-0h | | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CS1_CS0 | IEL | IEH | COSR | | | | |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | | | | |

**Table 17-22. SDCPARM1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | RESERVED | R | 0h | Reserved |
| 13 | CEN | R/W | 0h | Comparator Filter enable<br>0: Disable comparator filter<br>1: Enable comparator filter<br>Reset type: SYSRSn |
| 12-11 | RESERVED | R | 0h | Reserved |
| 10 | HZEN | R/W | 0h | High level (Z) Threshold crossing output enable<br>0: Disable Higher level Threshold (Z) crossing<br>1: Enable Higher level Threhold (Z) crossing<br>Reset type: SYSRSn |
| 9 | MFIE | R/W | 0h | Modulator Failure Interrupt Enable<br>0: Disable modulator failure interrupt and its flag<br>1: Enable modulator failure interrupt and its flag<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: Disable Lower Threshold interrupt<br>1: Enable Lower Threshold interrupt<br>Reset type: SYSRSn |
| 5 | IEH | R/W | 0h | High-level interrupt enable<br>0: Disable Higher Threshold interrupt<br>1: Enable Higher Threshold interrupt<br>Reset type: SYSRSn |

**Table 17-22. SDCPARM1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4-0 | COSR | R/W | 0h | Comparator Oversampling ratio.<br>The actual rate is COSR + 1.<br>These bits set the oversampling ratio of the filter.<br>0x1F represents an oversampling ratio of 32<br>Reset type: SYSRSn |

#### 17.8.1.1.12 SDDATA1 Register (Offset = 16h) [reset = 0h]

SDDATA1 is shown in Figure 17-20 and described in Table 17-23.

Return to Summary Table.

Data Filter Data Register (16 or 32bit) for Ch1

**Figure 17-20. SDDATA1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA32HI | | | | | | | | | | | | | | | | DATA16 | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | |

**Table 17-23. SDDATA1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | DATA32HI | R | 0h | Hi-order 16b in 32b mode, 16-bit Data in 16b mode<br>Reset type: SYSRSn |
| 15-0 | DATA16 | R | 0h | Lo-order 16b in 32b mode<br>Reset type: SYSRSn |

#### 17.8.1.1.13 SDDATFIFO1 Register (Offset = 18h) [reset = 0h]

SDDATFIFO1 is shown in Figure 17-21 and described in Table 17-24.

Return to Summary Table.

Filter Data FIFO Output(32b) for Ch1

**Figure 17-21. SDDATFIFO1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA32HI | | | | | | | | | | | | | | | | DATA16 | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | |

**Table 17-24. SDDATFIFO1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | DATA32HI | R | 0h | Hi-order 16b in 32b mode, 16-bit Data in 16b mode<br>Reset type: SYSRSn |
| 15-0 | DATA16 | R | 0h | Lo-order 16b in 32b mode<br>Reset type: SYSRSn |

#### 17.8.1.1.14 SDCDATA1 Register (Offset = 1Ah) [reset = 0h]

SDCDATA1 is shown in Figure 17-22 and described in Table 17-25.

Return to Summary Table.

Comparator Filter Data Register (16b) for Ch1

**Figure 17-22. SDCDATA1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA16 | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 17-25. SDCDATA1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | DATA16 | R | 0h | Comparator Data output - 16b only<br>Reset type: SYSRSn |

#### 17.8.1.1.15 SDCMPHZ1 Register (Offset = 1Ch) [reset = 0h]

SDCMPHZ1 is shown in Figure 17-23 and described in Table 17-26.

Return to Summary Table.

High-level (Z) Threshold Register for Ch1

**Figure 17-23. SDCMPHZ1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | HLTZ | | | | | | |
| R-0h | R/W-0h | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HLTZ | | | | | | | |
| R/W-0h | | | | | | | |

**Table 17-26. SDCMPHZ1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14-0 | HLTZ | R/W | 0h | Unsigned High-level threshold (Z) for the comparator filter output<br>Primarily intended for detecting "zero"-crossing events. Unlike the primary comparator SDCMPHx, it does not have the ability to generate an interrupt.<br>Reset type: SYSRSn |

#### 17.8.1.1.16 SDFIFOCTL1 Register (Offset = 1Dh) [reset = 0h]

SDFIFOCTL1 is shown in Figure 17-24 and described in Table 17-27.

Return to Summary Table.

FIFO Control Register for Ch1

**Figure 17-24. SDFIFOCTL1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| OVFIEN | DRINTSEL | FFEN | FFIEN | RESERVED | SDFFST | | |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R-0h | R-0h | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SDFFST | | RESERVED | SDFFIL | | | | |
| R-0h | | R-0h | R/W-0h | | | | |

**Table 17-27. SDFIFOCTL1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | OVFIEN | R/W | 0h | SDFIFO Overflow interrupt enable<br>0: SDFIFO Overflow condition will not generate an interrupt<br>1: SDFIFO overflow condition generates an interrupt on SDy_ERR<br>Reset type: SYSRSn |
| 14 | DRINTSEL | R/W | 0h | Data-Ready Interrupt (DRINT) source select<br>0 = AF1 (Select non-FIFO data-ready interrupt)<br>1 = SDFFINT1 (Select FIFO data-ready interrupt)<br>Reset type: SYSRSn |
| 13 | FFEN | R/W | 0h | SDFIFO Enable<br>0: Disable FIFO operation<br>1: Enable FIFO operation<br>Note: When FIFO is disabled, FIFO contents are cleared<br>Reset type: SYSRSn |
| 12 | FFIEN | R/W | 0h | SDFIFO data ready Interrupt Enable<br>Reset type: SYSRSn |
| 11 | RESERVED | R | 0h | Reserved |
| 10-6 | SDFFST | R | 0h | SDFIFO Status<br>00000 FIFO empty<br>00001 FIFO has 1 word<br>. . . .<br>10000 FIFO has 16 words<br>Reset type: SYSRSn |
| 5 | RESERVED | R | 0h | Reserved |
| 4-0 | SDFFIL | R/W | 0h | SDFIFO interrupt level bits<br>The FIFO will generate an interrupt when the FIFO status (SDFFST) >= FIFO level (SDFFIL )<br>Reset type: SYSRSn |

#### 17.8.1.1.17 SDSYNC1 Register (Offset = 1Eh) [reset = 400h]

SDSYNC1 is shown in Figure 17-25 and described in Table 17-28.

Return to Summary Table.

SD Filter Sync control for Ch1

**Figure 17-25. SDSYNC1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | WTSCLREN | FFSYNCCLREN | WTSYNCLR |
| R-0h | | | | | R/W-1h | R/W-0h | R=0/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WTSYNFLG | WTSYNCEN | SYNCSEL | | | | | |
| R-0h | R/W-0h | R/W-0h | | | | | |

**Table 17-28. SDSYNC1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-11 | RESERVED | R | 0h | Reserved |
| 10 | WTSCLREN | R/W | 1h | WTSYNFLG Clear-on-FIFOINT Enable<br>0: WTSYNFLG can only be cleared manually (using WTSYNCLR bit)<br>1: WTSYNFLG is cleared automatically on SDFFINT<br>Reset type: SYSRSn |
| 9 | FFSYNCCLREN | R/W | 0h | FIFO Clear-on-SDSYNC Enable<br>0: SDFIFO is not automaticaly cleared upon receiving SDSYNC<br>1: SDFIFO is automaticaly cleared upon receiving SDSYNC<br>Reset type: SYSRSn |
| 8 | WTSYNCLR | R=0/W | 0h | Wait-for-Sync Flag Clear (always reads 0)<br>0: Write of 0 has no affect<br>1: Write of 1 clears WTSYNFLG<br>Reset type: SYSRSn |
| 7 | WTSYNFLG | R | 0h | Wait-for-Sync Flag<br>0: SDSYNC event has not occurred<br>1: SDSYNC event occurred.<br>Reset type: SYSRSn |
| 6 | WTSYNCEN | R/W | 0h | Wait-for-Sync Enable<br>0: Incoming Data written to SDFIFO on every Data-Ready (DR) Event<br>1: Incoming Data written to SDFIFO on DR event only after SDSYNC event occurs<br>Reset type: SYSRSn |
| 5-0 | SYNCSEL | R/W | 0h | Defines source for the SDSYNC Input on this channel<br>Refer SDSYNCx.SYNCSEL table<br>Reset type: SYSRSn |

#### 17.8.1.1.18 SDCTLPARM2 Register (Offset = 20h) [reset = 0h]

SDCTLPARM2 is shown in Figure 17-26 and described in Table 17-29.

Return to Summary Table.

Control Parameter Register for Ch2

**Figure 17-26. 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-0h | R-0h | R-0h | R/W-0h | |

**Table 17-29. SDCTLPARM2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1-0 | MOD | R/W | 0h | Modulator clock modes<br>MODE 0: Modulator clock running at 1x data rate<br>MODE 1: modulator clock running at 1/2 data rate (dbl-edge clocking)<br>MODE 2: modulator clock absent (Manchester encoded data)<br>MODE 3: modulator clock running at 2x data rate<br>Reset type: SYSRSn |

#### 17.8.1.1.19 SDDFPARM2 Register (Offset = 21h) [reset = 0h]

SDDFPARM2 is shown in Figure 17-27 and described in Table 17-30.

Return to Summary Table.

Data Filter Parameter Register for Ch2

**Figure 17-27. 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 17-30. SDDFPARM2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12 | SDSYNCEN | R/W | 0h | PWM synchronization (SDSYNC) of data filter<br>0: PWM synchronization of data filter is disabled<br>1: PWM synchronization of data filter is enabled<br>Note: SDSYNCx.SYNCSEL bits define which PWM signal is used to synchronize PWMs<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 | Data filter 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 data filter is disabled and no data is produced<br>1: The data filter is enabled and data are produced in the data filter<br>Note: When filter is disabled, DOSR counter held in reset, filter data erased. Also resets FIFO pointers and clears the FIFO<br>Reset type: SYSRSn |
| 7-0 | DOSR | R/W | 0h | Data filter Oversampling ratio<br>The actual oversampling ratio of data filter is DOSR + 1<br>These bits set the oversampling ratio of the data filter.<br>0x0FF represents an oversampling ratio of 256.<br>Reset type: SYSRSn |

#### 17.8.1.1.20 SDDPARM2 Register (Offset = 22h) [reset = 0h]

SDDPARM2 is shown in Figure 17-28 and described in Table 17-31.

Return to Summary Table.

Data Parameter Register for Ch2

**Figure 17-28. SDDPARM2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SH | | | | | DR | RESERVED | |
| R/W-0h | | | | | R/W-0h | R-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |

**Table 17-31. 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 filter Data representation<br>0: Data stored in 16b 2's complement<br>1: Data stored in 32b 2's complement<br>Reset type: SYSRSn |
| 9-0 | RESERVED | R | 0h | Reserved |

#### 17.8.1.1.21 SDCMPH2 Register (Offset = 23h) [reset = 7FFFh]

SDCMPH2 is shown in Figure 17-29 and described in Table 17-32.

Return to Summary Table.

High-level Threshold Register for Ch2

**Figure 17-29. SDCMPH2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | HLT | | | | | | |
| R-0h | R/W-7FFFh | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HLT | | | | | | | |
| R/W-7FFFh | | | | | | | |

**Table 17-32. SDCMPH2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14-0 | HLT | R/W | 7FFFh | Unsigned high-level threshold for the comparator filter output.<br>Reset type: SYSRSn |

#### 17.8.1.1.22 SDCMPL2 Register (Offset = 24h) [reset = 0h]

SDCMPL2 is shown in Figure 17-30 and described in Table 17-33.

Return to Summary Table.

Low-level Threshold Register for Ch2

**Figure 17-30. SDCMPL2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | LLT | | | | | | |
| R-0h | R/W-0h | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LLT | | | | | | | |
| R/W-0h | | | | | | | |

**Table 17-33. SDCMPL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14-0 | LLT | R/W | 0h | Unsigned low-level threshold for the comparator filter output.<br>Reset type: SYSRSn |

#### 17.8.1.1.23 SDCPARM2 Register (Offset = 25h) [reset = 0h]

SDCPARM2 is shown in Figure 17-31 and described in Table 17-34.

Return to Summary Table.

Comparator Filter Parameter Register for Ch2

**Figure 17-31. SDCPARM2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | CEN | RESERVED | | HZEN | MFIE | CS1_CS0 |
| R-0h | | R/W-0h | R-0h | | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CS1_CS0 | IEL | IEH | COSR | | | | |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | | | | |

**Table 17-34. SDCPARM2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | RESERVED | R | 0h | Reserved |
| 13 | CEN | R/W | 0h | Comparator Filter enable<br>0: Disable comparator filter<br>1: Enable comparator filter<br>Reset type: SYSRSn |
| 12-11 | RESERVED | R | 0h | Reserved |
| 10 | HZEN | R/W | 0h | High level (Z) Threshold crossing output enable<br>0: Disable Higher level Threshold (Z) crossing<br>1: Enable Higher level Threhold (Z) crossing<br>Reset type: SYSRSn |
| 9 | MFIE | R/W | 0h | Modulator Failure Interrupt Enable<br>0: Disable modulator failure interrupt and its flag<br>1: Enable modulator failure interrupt and its flag<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: Disable Lower Threshold interrupt<br>1: Enable Lower Threshold interrupt<br>Reset type: SYSRSn |
| 5 | IEH | R/W | 0h | High-level interrupt enable<br>0: Disable Higher Threshold interrupt<br>1: Enable Higher Threshold interrupt<br>Reset type: SYSRSn |

**Table 17-34. SDCPARM2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4-0 | COSR | R/W | 0h | Comparator Oversampling ratio.<br>The actual rate is COSR + 1.<br>These bits set the oversampling ratio of the filter.<br>0x1F represents an oversampling ratio of 32<br>Reset type: SYSRSn |

#### 17.8.1.1.24 SDDATA2 Register (Offset = 26h) [reset = 0h]

SDDATA2 is shown in Figure 17-32 and described in Table 17-35.

Return to Summary Table.

Data Filter Data Register (16 or 32bit) for Ch2

**Figure 17-32. SDDATA2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA32HI | | | | | | | | | | | | | | | | DATA16 | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | |

**Table 17-35. SDDATA2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | DATA32HI | R | 0h | Hi-order 16b in 32b mode, 16-bit Data in 16b mode<br>Reset type: SYSRSn |
| 15-0 | DATA16 | R | 0h | Lo-order 16b in 32b mode<br>Reset type: SYSRSn |

#### 17.8.1.1.25 SDDATFIFO2 Register (Offset = 28h) [reset = 0h]

SDDATFIFO2 is shown in Figure 17-33 and described in Table 17-36.

Return to Summary Table.

Filter Data FIFO Output(32b) for Ch2

**Figure 17-33. SDDATFIFO2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA32HI | | | | | | | | | | | | | | | | DATA16 | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | |

**Table 17-36. SDDATFIFO2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | DATA32HI | R | 0h | Hi-order 16b in 32b mode, 16-bit Data in 16b mode<br>Reset type: SYSRSn |
| 15-0 | DATA16 | R | 0h | Lo-order 16b in 32b mode<br>Reset type: SYSRSn |

#### 17.8.1.1.26 SDCDATA2 Register (Offset = 2Ah) [reset = 0h]

SDCDATA2 is shown in Figure 17-34 and described in Table 17-37.

Return to Summary Table.

Comparator Filter Data Register (16b) for Ch2

**Figure 17-34. SDCDATA2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA16 | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 17-37. SDCDATA2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | DATA16 | R | 0h | Comparator Data output - 16b only<br>Reset type: SYSRSn |

#### 17.8.1.1.27 SDCMPHZ2 Register (Offset = 2Ch) [reset = 0h]

SDCMPHZ2 is shown in Figure 17-35 and described in Table 17-38.

Return to Summary Table.

High-level (Z) Threshold Register for Ch2

**Figure 17-35. SDCMPHZ2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | HLTZ | | | | | | |
| R-0h | R/W-0h | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HLTZ | | | | | | | |
| R/W-0h | | | | | | | |

**Table 17-38. SDCMPHZ2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14-0 | HLTZ | R/W | 0h | Unsigned High-level threshold (Z) for the comparator filter output<br>Primarily intended for detecting "zero"-crossing events. Unlike the primary comparator SDCMPHx, it does not have the ability to generate an interrupt.<br>Reset type: SYSRSn |

#### 17.8.1.1.28 SDFIFOCTL2 Register (Offset = 2Dh) [reset = 0h]

SDFIFOCTL2 is shown in Figure 17-36 and described in Table 17-39.

Return to Summary Table.

FIFO Control Register for Ch2

**Figure 17-36. SDFIFOCTL2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| OVFIEN | DRINTSEL | FFEN | FFIEN | RESERVED | SDFFST | | |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R-0h | R-0h | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SDFFST | | RESERVED | SDFFIL | | | | |
| R-0h | | R-0h | R/W-0h | | | | |

**Table 17-39. SDFIFOCTL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | OVFIEN | R/W | 0h | SDFIFO Overflow interrupt enable<br>0: SDFIFO Overflow condition will not generate an interrupt<br>1: SDFIFO overflow condition generates an interrupt on SDy_ERR<br>Reset type: SYSRSn |
| 14 | DRINTSEL | R/W | 0h | Data-Ready Interrupt (DRINT) source select<br>0 = AF1 (Select non-FIFO data-ready interrupt)<br>1 = SDFFINT1 (Select FIFO data-ready interrupt)<br>Reset type: SYSRSn |
| 13 | FFEN | R/W | 0h | SDFIFO Enable<br>0: Disable FIFO operation<br>1: Enable FIFO operation<br>Note: When FIFO is disabled, FIFO contents are cleared<br>Reset type: SYSRSn |
| 12 | FFIEN | R/W | 0h | SDFIFO data ready Interrupt Enable<br>Reset type: SYSRSn |
| 11 | RESERVED | R | 0h | Reserved |
| 10-6 | SDFFST | R | 0h | SDFIFO Status<br>00000 FIFO empty<br>00001 FIFO has 1 word<br>. . . .<br>10000 FIFO has 16 words<br>Reset type: SYSRSn |
| 5 | RESERVED | R | 0h | Reserved |
| 4-0 | SDFFIL | R/W | 0h | SDFIFO interrupt level bits<br>The FIFO will generate an interrupt when the FIFO status (SDFFST) >= FIFO level (SDFFIL )<br>Reset type: SYSRSn |

#### 17.8.1.1.29 SDSYNC2 Register (Offset = 2Eh) [reset = 400h]

SDSYNC2 is shown in Figure 17-37 and described in Table 17-40.

Return to Summary Table.

SD Filter Sync control for Ch2

**Figure 17-37. SDSYNC2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | WTSCLREN | FFSYNCCLREN | WTSYNCLR |
| R-0h | | | | | R/W-1h | R/W-0h | R=0/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WTSYNFLG | WTSYNCEN | SYNCSEL | | | | | |
| R-0h | R/W-0h | R/W-0h | | | | | |

**Table 17-40. SDSYNC2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-11 | RESERVED | R | 0h | Reserved |
| 10 | WTSCLREN | R/W | 1h | WTSYNFLG Clear-on-FIFOINT Enable<br>0: WTSYNFLG can only be cleared manually (using WTSYNCLR bit)<br>1: WTSYNFLG is cleared automatically on SDFFINT<br>Reset type: SYSRSn |
| 9 | FFSYNCCLREN | R/W | 0h | FIFO Clear-on-SDSYNC Enable<br>0: SDFIFO is not automaticaly cleared upon receiving SDSYNC<br>1: SDFIFO is automaticaly cleared upon receiving SDSYNC<br>Reset type: SYSRSn |
| 8 | WTSYNCLR | R=0/W | 0h | Wait-for-Sync Flag Clear (always reads 0)<br>0: Write of 0 has no affect<br>1: Write of 1 clears WTSYNFLG<br>Reset type: SYSRSn |
| 7 | WTSYNFLG | R | 0h | Wait-for-Sync Flag<br>0: SDSYNC event has not occurred<br>1: SDSYNC event occurred.<br>Reset type: SYSRSn |
| 6 | WTSYNCEN | R/W | 0h | Wait-for-Sync Enable<br>0: Incoming Data written to SDFIFO on every Data-Ready (DR) Event<br>1: Incoming Data written to SDFIFO on DR event only after SDSYNC event occurs<br>Reset type: SYSRSn |
| 5-0 | SYNCSEL | R/W | 0h | Defines source for the SDSYNC Input on this channel<br>Refer SDSYNCx.SYNCSEL table<br>Reset type: SYSRSn |

#### 17.8.1.1.30 SDCTLPARM3 Register (Offset = 30h) [reset = 0h]

SDCTLPARM3 is shown in Figure 17-38 and described in Table 17-41.

Return to Summary Table.

Control Parameter Register for Ch3

**Figure 17-38. 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-0h | R-0h | R-0h | R/W-0h | |

**Table 17-41. SDCTLPARM3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1-0 | MOD | R/W | 0h | Modulator clock modes<br>MODE 0: Modulator clock running at 1x data rate<br>MODE 1: modulator clock running at 1/2 data rate (dbl-edge clocking)<br>MODE 2: modulator clock absent (Manchester encoded data)<br>MODE 3: modulator clock running at 2x data rate<br>Reset type: SYSRSn |

#### 17.8.1.1.31 SDDFPARM3 Register (Offset = 31h) [reset = 0h]

SDDFPARM3 is shown in Figure 17-39 and described in Table 17-42.

Return to Summary Table.

Data Filter Parameter Register for Ch3

**Figure 17-39. 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 17-42. SDDFPARM3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12 | SDSYNCEN | R/W | 0h | PWM synchronization (SDSYNC) of data filter<br>0: PWM synchronization of data filter is disabled<br>1: PWM synchronization of data filter is enabled<br>Note: SDSYNCx.SYNCSEL bits define which PWM signal is used to synchronize PWMs<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 | Data filter 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 data filter is disabled and no data is produced<br>1: The data filter is enabled and data are produced in the data filter<br>Note: When filter is disabled, DOSR counter held in reset, filter data erased. Also resets FIFO pointers and clears the FIFO<br>Reset type: SYSRSn |
| 7-0 | DOSR | R/W | 0h | Data filter Oversampling ratio<br>The actual oversampling ratio of data filter is DOSR + 1<br>These bits set the oversampling ratio of the data filter.<br>0x0FF represents an oversampling ratio of 256.<br>Reset type: SYSRSn |

#### 17.8.1.1.32 SDDPARM3 Register (Offset = 32h) [reset = 0h]

SDDPARM3 is shown in Figure 17-40 and described in Table 17-43.

Return to Summary Table.

Data Parameter Register for Ch3

**Figure 17-40. SDDPARM3 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SH | | | | | DR | RESERVED | |
| R/W-0h | | | | | R/W-0h | R-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |

**Table 17-43. 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 filter Data representation<br>0: Data stored in 16b 2's complement<br>1: Data stored in 32b 2's complement<br>Reset type: SYSRSn |
| 9-0 | RESERVED | R | 0h | Reserved |

#### 17.8.1.1.33 SDCMPH3 Register (Offset = 33h) [reset = 7FFFh]

SDCMPH3 is shown in Figure 17-41 and described in Table 17-44.

Return to Summary Table.

High-level Threshold Register for Ch3

**Figure 17-41. SDCMPH3 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | HLT | | | | | | |
| R-0h | R/W-7FFFh | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HLT | | | | | | | |
| R/W-7FFFh | | | | | | | |

**Table 17-44. SDCMPH3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14-0 | HLT | R/W | 7FFFh | Unsigned high-level threshold for the comparator filter output.<br>Reset type: SYSRSn |

#### 17.8.1.1.34 SDCMPL3 Register (Offset = 34h) [reset = 0h]

SDCMPL3 is shown in Figure 17-42 and described in Table 17-45.

Return to Summary Table.

Low-level Threshold Register for Ch3

**Figure 17-42. SDCMPL3 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | LLT | | | | | | |
| R-0h | R/W-0h | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LLT | | | | | | | |
| R/W-0h | | | | | | | |

**Table 17-45. SDCMPL3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14-0 | LLT | R/W | 0h | Unsigned low-level threshold for the comparator filter output.<br>Reset type: SYSRSn |

#### 17.8.1.1.35 SDCPARM3 Register (Offset = 35h) [reset = 0h]

SDCPARM3 is shown in Figure 17-43 and described in Table 17-46.

Return to Summary Table.

Comparator Filter Parameter Register for Ch3

**Figure 17-43. SDCPARM3 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | CEN | RESERVED | | HZEN | MFIE | CS1_CS0 |
| R-0h | | R/W-0h | R-0h | | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CS1_CS0 | IEL | IEH | COSR | | | | |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | | | | |

**Table 17-46. SDCPARM3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | RESERVED | R | 0h | Reserved |
| 13 | CEN | R/W | 0h | Comparator Filter enable<br>0: Disable comparator filter<br>1: Enable comparator filter<br>Reset type: SYSRSn |
| 12-11 | RESERVED | R | 0h | Reserved |
| 10 | HZEN | R/W | 0h | High level (Z) Threshold crossing output enable<br>0: Disable Higher level Threshold (Z) crossing<br>1: Enable Higher level Threhold (Z) crossing<br>Reset type: SYSRSn |
| 9 | MFIE | R/W | 0h | Modulator Failure Interrupt Enable<br>0: Disable modulator failure interrupt and its flag<br>1: Enable modulator failure interrupt and its flag<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: Disable Lower Threshold interrupt<br>1: Enable Lower Threshold interrupt<br>Reset type: SYSRSn |
| 5 | IEH | R/W | 0h | High-level interrupt enable<br>0: Disable Higher Threshold interrupt<br>1: Enable Higher Threshold interrupt<br>Reset type: SYSRSn |

**Table 17-46. SDCPARM3 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4-0 | COSR | R/W | 0h | Comparator Oversampling ratio.<br>The actual rate is COSR + 1.<br>These bits set the oversampling ratio of the filter.<br>0x1F represents an oversampling ratio of 32<br>Reset type: SYSRSn |

#### 17.8.1.1.36 SDDATA3 Register (Offset = 36h) [reset = 0h]

SDDATA3 is shown in Figure 17-44 and described in Table 17-47.

Return to Summary Table.

Data Filter Data Register (16 or 32bit) for Ch3

**Figure 17-44. SDDATA3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA32HI | | | | | | | | | | | | | | | | DATA16 | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | |

**Table 17-47. SDDATA3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | DATA32HI | R | 0h | Hi-order 16b in 32b mode, 16-bit Data in 16b mode<br>Reset type: SYSRSn |
| 15-0 | DATA16 | R | 0h | Lo-order 16b in 32b mode<br>Reset type: SYSRSn |

#### 17.8.1.1.37 SDDATFIFO3 Register (Offset = 38h) [reset = 0h]

SDDATFIFO3 is shown in Figure 17-45 and described in Table 17-48.

Return to Summary Table.

Filter Data FIFO Output(32b) for Ch3

**Figure 17-45. SDDATFIFO3 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA32HI | | | | | | | | | | | | | | | | DATA16 | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | |

**Table 17-48. SDDATFIFO3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | DATA32HI | R | 0h | Hi-order 16b in 32b mode, 16-bit Data in 16b mode<br>Reset type: SYSRSn |
| 15-0 | DATA16 | R | 0h | Lo-order 16b in 32b mode<br>Reset type: SYSRSn |

#### 17.8.1.1.38 SDCDATA3 Register (Offset = 3Ah) [reset = 0h]

SDCDATA3 is shown in Figure 17-46 and described in Table 17-49.

Return to Summary Table.

Comparator Filter Data Register (16b) for Ch3

**Figure 17-46. SDCDATA3 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA16 | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 17-49. SDCDATA3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | DATA16 | R | 0h | Comparator Data output - 16b only<br>Reset type: SYSRSn |

#### 17.8.1.1.39 SDCMPHZ3 Register (Offset = 3Ch) [reset = 0h]

SDCMPHZ3 is shown in Figure 17-47 and described in Table 17-50.

Return to Summary Table.

High-level (Z) Threshold Register for Ch3

**Figure 17-47. SDCMPHZ3 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | HLTZ | | | | | | |
| R-0h | R/W-0h | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HLTZ | | | | | | | |
| R/W-0h | | | | | | | |

**Table 17-50. SDCMPHZ3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14-0 | HLTZ | R/W | 0h | Unsigned High-level threshold (Z) for the comparator filter output<br>Primarily intended for detecting "zero"-crossing events. Unlike the primary comparator SDCMPHx, it does not have the ability to generate an interrupt.<br>Reset type: SYSRSn |

#### 17.8.1.1.40 SDFIFOCTL3 Register (Offset = 3Dh) [reset = 0h]

SDFIFOCTL3 is shown in Figure 17-48 and described in Table 17-51.

Return to Summary Table.

FIFO Control Register for Ch3

**Figure 17-48. SDFIFOCTL3 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| OVFIEN | DRINTSEL | FFEN | FFIEN | RESERVED | SDFFST | | |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R-0h | R-0h | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SDFFST | | RESERVED | SDFFIL | | | | |
| R-0h | | R-0h | R/W-0h | | | | |

**Table 17-51. SDFIFOCTL3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | OVFIEN | R/W | 0h | SDFIFO Overflow interrupt enable<br>0: SDFIFO Overflow condition will not generate an interrupt<br>1: SDFIFO overflow condition generates an interrupt on SDy_ERR<br>Reset type: SYSRSn |
| 14 | DRINTSEL | R/W | 0h | Data-Ready Interrupt (DRINT) source select<br>0 = AF1 (Select non-FIFO data-ready interrupt)<br>1 = SDFFINT1 (Select FIFO data-ready interrupt)<br>Reset type: SYSRSn |
| 13 | FFEN | R/W | 0h | SDFIFO Enable<br>0: Disable FIFO operation<br>1: Enable FIFO operation<br>Note: When FIFO is disabled, FIFO contents are cleared<br>Reset type: SYSRSn |
| 12 | FFIEN | R/W | 0h | SDFIFO data ready Interrupt Enable<br>Reset type: SYSRSn |
| 11 | RESERVED | R | 0h | Reserved |
| 10-6 | SDFFST | R | 0h | SDFIFO Status<br>00000 FIFO empty<br>00001 FIFO has 1 word<br>. . . .<br>10000 FIFO has 16 words<br>Reset type: SYSRSn |
| 5 | RESERVED | R | 0h | Reserved |
| 4-0 | SDFFIL | R/W | 0h | SDFIFO interrupt level bits<br>The FIFO will generate an interrupt when the FIFO status (SDFFST) >= FIFO level (SDFFIL )<br>Reset type: SYSRSn |

#### 17.8.1.1.41 SDSYNC3 Register (Offset = 3Eh) [reset = 400h]

SDSYNC3 is shown in Figure 17-49 and described in Table 17-52.

Return to Summary Table.

SD Filter Sync control for Ch3

**Figure 17-49. SDSYNC3 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | WTSCLREN | FFSYNCCLREN | WTSYNCLR |
| R-0h | | | | | R/W-1h | R/W-0h | R=0/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WTSYNFLG | WTSYNCEN | SYNCSEL | | | | | |
| R-0h | R/W-0h | R/W-0h | | | | | |

**Table 17-52. SDSYNC3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-11 | RESERVED | R | 0h | Reserved |
| 10 | WTSCLREN | R/W | 1h | WTSYNFLG Clear-on-FIFOINT Enable<br>0: WTSYNFLG can only be cleared manually (using WTSYNCLR bit)<br>1: WTSYNFLG is cleared automatically on SDFFINT<br>Reset type: SYSRSn |
| 9 | FFSYNCCLREN | R/W | 0h | FIFO Clear-on-SDSYNC Enable<br>0: SDFIFO is not automaticaly cleared upon receiving SDSYNC<br>1: SDFIFO is automaticaly cleared upon receiving SDSYNC<br>Reset type: SYSRSn |
| 8 | WTSYNCLR | R=0/W | 0h | Wait-for-Sync Flag Clear (always reads 0)<br>0: Write of 0 has no affect<br>1: Write of 1 clears WTSYNFLG<br>Reset type: SYSRSn |
| 7 | WTSYNFLG | R | 0h | Wait-for-Sync Flag<br>0: SDSYNC event has not occurred<br>1: SDSYNC event occurred.<br>Reset type: SYSRSn |
| 6 | WTSYNCEN | R/W | 0h | Wait-for-Sync Enable<br>0: Incoming Data written to SDFIFO on every Data-Ready (DR) Event<br>1: Incoming Data written to SDFIFO on DR event only after SDSYNC event occurs<br>Reset type: SYSRSn |
| 5-0 | SYNCSEL | R/W | 0h | Defines source for the SDSYNC Input on this channel<br>Refer SDSYNCx.SYNCSEL table<br>Reset type: SYSRSn |

#### 17.8.1.1.42 SDCTLPARM4 Register (Offset = 40h) [reset = 0h]

SDCTLPARM4 is shown in Figure 17-50 and described in Table 17-53.

Return to Summary Table.

Control Parameter Register for Ch4

**Figure 17-50. 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-0h | R-0h | R-0h | R/W-0h | |

**Table 17-53. SDCTLPARM4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1-0 | MOD | R/W | 0h | Modulator clock modes<br>MODE 0: Modulator clock running at 1x data rate<br>MODE 1: modulator clock running at 1/2 data rate (dbl-edge clocking)<br>MODE 2: modulator clock absent (Manchester encoded data)<br>MODE 3: modulator clock running at 2x data rate<br>Reset type: SYSRSn |

#### 17.8.1.1.43 SDDFPARM4 Register (Offset = 41h) [reset = 0h]

SDDFPARM4 is shown in Figure 17-51 and described in Table 17-54.

Return to Summary Table.

Data Filter Parameter Register for Ch4

**Figure 17-51. 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 17-54. SDDFPARM4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12 | SDSYNCEN | R/W | 0h | PWM synchronization (SDSYNC) of data filter<br>0: PWM synchronization of data filter is disabled<br>1: PWM synchronization of data filter is enabled<br>Note: SDSYNCx.SYNCSEL bits define which PWM signal is used to synchronize PWMs<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 | Data filter 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 data filter is disabled and no data is produced<br>1: The data filter is enabled and data are produced in the data filter<br>Note: When filter is disabled, DOSR counter held in reset, filter data erased. Also resets FIFO pointers and clears the FIFO<br>Reset type: SYSRSn |
| 7-0 | DOSR | R/W | 0h | Data filter Oversampling ratio<br>The actual oversampling ratio of data filter is DOSR + 1<br>These bits set the oversampling ratio of the data filter.<br>0x0FF represents an oversampling ratio of 256.<br>Reset type: SYSRSn |

#### 17.8.1.1.44 SDDPARM4 Register (Offset = 42h) [reset = 0h]

SDDPARM4 is shown in Figure 17-52 and described in Table 17-55.

Return to Summary Table.

Data Parameter Register for Ch4

**Figure 17-52. SDDPARM4 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SH | | | | | DR | RESERVED | |
| R/W-0h | | | | | R/W-0h | R-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |

**Table 17-55. 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 filter Data representation<br>0: Data stored in 16b 2's complement<br>1: Data stored in 32b 2's complement<br>Reset type: SYSRSn |
| 9-0 | RESERVED | R | 0h | Reserved |

#### 17.8.1.1.45 SDCMPH4 Register (Offset = 43h) [reset = 7FFFh]

SDCMPH4 is shown in Figure 17-53 and described in Table 17-56.

Return to Summary Table.

High-level Threshold Register for Ch4

**Figure 17-53. SDCMPH4 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | HLT | | | | | | |
| R-0h | R/W-7FFFh | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HLT | | | | | | | |
| R/W-7FFFh | | | | | | | |

**Table 17-56. SDCMPH4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14-0 | HLT | R/W | 7FFFh | Unsigned high-level threshold for the comparator filter output.<br>Reset type: SYSRSn |

#### 17.8.1.1.46 SDCMPL4 Register (Offset = 44h) [reset = 0h]

SDCMPL4 is shown in Figure 17-54 and described in Table 17-57.

Return to Summary Table.

Low-level Threshold Register for Ch4

**Figure 17-54. SDCMPL4 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | LLT | | | | | | |
| R-0h | R/W-0h | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LLT | | | | | | | |
| R/W-0h | | | | | | | |

**Table 17-57. SDCMPL4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14-0 | LLT | R/W | 0h | Unsigned low-level threshold for the comparator filter output.<br>Reset type: SYSRSn |

#### 17.8.1.1.47 SDCPARM4 Register (Offset = 45h) [reset = 0h]

SDCPARM4 is shown in Figure 17-55 and described in Table 17-58.

Return to Summary Table.

Comparator Filter Parameter Register for Ch4

**Figure 17-55. SDCPARM4 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | CEN | RESERVED | | HZEN | MFIE | CS1_CS0 |
| R-0h | | R/W-0h | R-0h | | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CS1_CS0 | IEL | IEH | COSR | | | | |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | | | | |

**Table 17-58. SDCPARM4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | RESERVED | R | 0h | Reserved |
| 13 | CEN | R/W | 0h | Comparator Filter enable<br>0: Disable comparator filter<br>1: Enable comparator filter<br>Reset type: SYSRSn |
| 12-11 | RESERVED | R | 0h | Reserved |
| 10 | HZEN | R/W | 0h | High level (Z) Threshold crossing output enable<br>0: Disable Higher level Threshold (Z) crossing<br>1: Enable Higher level Threhold (Z) crossing<br>Reset type: SYSRSn |
| 9 | MFIE | R/W | 0h | Modulator Failure Interrupt Enable<br>0: Disable modulator failure interrupt and its flag<br>1: Enable modulator failure interrupt and its flag<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: Disable Lower Threshold interrupt<br>1: Enable Lower Threshold interrupt<br>Reset type: SYSRSn |
| 5 | IEH | R/W | 0h | High-level interrupt enable<br>0: Disable Higher Threshold interrupt<br>1: Enable Higher Threshold interrupt<br>Reset type: SYSRSn |

**Table 17-58. SDCPARM4 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4-0 | COSR | R/W | 0h | Comparator Oversampling ratio.<br>The actual rate is COSR + 1.<br>These bits set the oversampling ratio of the filter.<br>0x1F represents an oversampling ratio of 32<br>Reset type: SYSRSn |

#### 17.8.1.1.48 SDDATA4 Register (Offset = 46h) [reset = 0h]

SDDATA4 is shown in Figure 17-56 and described in Table 17-59.

Return to Summary Table.

Data Filter Data Register (16 or 32bit) for Ch4

**Figure 17-56. SDDATA4 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA32HI | | | | | | | | | | | | | | | | DATA16 | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | |

**Table 17-59. SDDATA4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | DATA32HI | R | 0h | Hi-order 16b in 32b mode, 16-bit Data in 16b mode<br>Reset type: SYSRSn |
| 15-0 | DATA16 | R | 0h | Lo-order 16b in 32b mode<br>Reset type: SYSRSn |

#### 17.8.1.1.49 SDDATFIFO4 Register (Offset = 48h) [reset = 0h]

SDDATFIFO4 is shown in Figure 17-57 and described in Table 17-60.

Return to Summary Table.

Filter Data FIFO Output(32b) for Ch4

**Figure 17-57. SDDATFIFO4 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA32HI | | | | | | | | | | | | | | | | DATA16 | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | R-0h | | | | | | | | | | | | | | | |

**Table 17-60. SDDATFIFO4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | DATA32HI | R | 0h | Hi-order 16b in 32b mode, 16-bit Data in 16b mode<br>Reset type: SYSRSn |
| 15-0 | DATA16 | R | 0h | Lo-order 16b in 32b mode<br>Reset type: SYSRSn |

#### 17.8.1.1.50 SDCDATA4 Register (Offset = 4Ah) [reset = 0h]

SDCDATA4 is shown in Figure 17-58 and described in Table 17-61.

Return to Summary Table.

Comparator Filter Data Register (16b) for Ch4

**Figure 17-58. SDCDATA4 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA16 | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 17-61. SDCDATA4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | DATA16 | R | 0h | Comparator Data output - 16b only<br>Reset type: SYSRSn |

#### 17.8.1.1.51 SDCMPHZ4 Register (Offset = 4Ch) [reset = 0h]

SDCMPHZ4 is shown in Figure 17-59 and described in Table 17-62.

Return to Summary Table.

High-level (Z) Threshold Register for Ch4

**Figure 17-59. SDCMPHZ4 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | HLTZ | | | | | | |
| R-0h | R/W-0h | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HLTZ | | | | | | | |
| R/W-0h | | | | | | | |

**Table 17-62. SDCMPHZ4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14-0 | HLTZ | R/W | 0h | Unsigned High-level threshold (Z) for the comparator filter output<br>Primarily intended for detecting "zero"-crossing events. Unlike the primary comparator SDCMPHx, it does not have the ability to generate an interrupt.<br>Reset type: SYSRSn |

#### 17.8.1.1.52 SDFIFOCTL4 Register (Offset = 4Dh) [reset = 0h]

SDFIFOCTL4 is shown in Figure 17-60 and described in Table 17-63.

Return to Summary Table.

FIFO Control Register for Ch4

**Figure 17-60. SDFIFOCTL4 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| OVFIEN | DRINTSEL | FFEN | FFIEN | RESERVED | SDFFST | | |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R-0h | R-0h | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SDFFST | | RESERVED | SDFFIL | | | | |
| R-0h | | R-0h | R/W-0h | | | | |

**Table 17-63. SDFIFOCTL4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | OVFIEN | R/W | 0h | SDFIFO Overflow interrupt enable<br>0: SDFIFO Overflow condition will not generate an interrupt<br>1: SDFIFO overflow condition generates an interrupt on SDy_ERR<br>Reset type: SYSRSn |
| 14 | DRINTSEL | R/W | 0h | Data-Ready Interrupt (DRINT) source select<br>0 = AF1 (Select non-FIFO data-ready interrupt)<br>1 = SDFFINT1 (Select FIFO data-ready interrupt)<br>Reset type: SYSRSn |
| 13 | FFEN | R/W | 0h | SDFIFO Enable<br>0: Disable FIFO operation<br>1: Enable FIFO operation<br>Note: When FIFO is disabled, FIFO contents are cleared<br>Reset type: SYSRSn |
| 12 | FFIEN | R/W | 0h | SDFIFO data ready Interrupt Enable<br>Reset type: SYSRSn |
| 11 | RESERVED | R | 0h | Reserved |
| 10-6 | SDFFST | R | 0h | SDFIFO Status<br>00000 FIFO empty<br>00001 FIFO has 1 word<br>. . . .<br>10000 FIFO has 16 words<br>Reset type: SYSRSn |
| 5 | RESERVED | R | 0h | Reserved |
| 4-0 | SDFFIL | R/W | 0h | SDFIFO interrupt level bits<br>The FIFO will generate an interrupt when the FIFO status (SDFFST) >= FIFO level (SDFFIL )<br>Reset type: SYSRSn |

#### 17.8.1.1.53 SDSYNC4 Register (Offset = 4Eh) [reset = 400h]

SDSYNC4 is shown in Figure 17-61 and described in Table 17-64.

Return to Summary Table.

SD Filter Sync control for Ch4

**Figure 17-61. SDSYNC4 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | WTSCLREN | FFSYNCCLREN | WTSYNCLR |
| R-0h | | | | | R/W-1h | R/W-0h | R=0/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WTSYNFLG | WTSYNCEN | SYNCSEL | | | | | |
| R-0h | R/W-0h | R/W-0h | | | | | |

**Table 17-64. SDSYNC4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-11 | RESERVED | R | 0h | Reserved |
| 10 | WTSCLREN | R/W | 1h | WTSYNFLG Clear-on-FIFOINT Enable<br>0: WTSYNFLG can only be cleared manually (using WTSYNCLR bit)<br>1: WTSYNFLG is cleared automatically on SDFFINT<br>Reset type: SYSRSn |
| 9 | FFSYNCCLREN | R/W | 0h | FIFO Clear-on-SDSYNC Enable<br>0: SDFIFO is not automaticaly cleared upon receiving SDSYNC<br>1: SDFIFO is automaticaly cleared upon receiving SDSYNC<br>Reset type: SYSRSn |
| 8 | WTSYNCLR | R=0/W | 0h | Wait-for-Sync Flag Clear (always reads 0)<br>0: Write of 0 has no affect<br>1: Write of 1 clears WTSYNFLG<br>Reset type: SYSRSn |
| 7 | WTSYNFLG | R | 0h | Wait-for-Sync Flag<br>0: SDSYNC event has not occurred<br>1: SDSYNC event occurred.<br>Reset type: SYSRSn |
| 6 | WTSYNCEN | R/W | 0h | Wait-for-Sync Enable<br>0: Incoming Data written to SDFIFO on every Data-Ready (DR) Event<br>1: Incoming Data written to SDFIFO on DR event only after SDSYNC event occurs<br>Reset type: SYSRSn |
| 5-0 | SYNCSEL | R/W | 0h | Defines source for the SDSYNC Input on this channel<br>Refer SDSYNCx.SYNCSEL table<br>Reset type: SYSRSn |

# Enhanced Pulse Width Modulator (ePWM)

The enhanced pulse width modulator (ePWM) peripheral is a key element in controlling many of the power electronic systems found in both commercial and industrial equipment. These systems include digital motor control, switch mode power supply control, uninterruptible power supplies (UPS), and other forms of power conversion. The ePWM peripheral performs a digital to analog (DAC) function, where the duty cycle is equivalent to a DAC analog value; it is sometimes referred to as a power DAC.

This chapter is applicable for ePWM type 4 with added register protection capability. See the *TMS320x28xx, 28xxx DSP Peripheral Reference Guide* for a list of all devices with an ePWM submodule of the same type, to determine the differences between the types, and for a list of device-specific differences within a type.

Further information about this device can be found in the following document(s):

- *Flexible PWMs Enable Multi-Axis Drives, Multi-Level Inverters*

**Topic** **Page**

## 18.1 Introduction

This chapter includes an overview and information about each submodule:

- Time-Base Submodule
- Counter Compare Submodule
- Action Qualifier Submodule
- Dead-Band Generator Submodule
- PWM Chopper (PC) Submodule
- Trip Zone Submodule
- Event Trigger Submodule
- Digital Compare Submodule

The ePWM Type 4 is functionally compatible to Type 2 (a Type 3 does not exist). Type 4 has the following enhancements in addition to the Type 2 features:

- **Register Address Map**

  Additional registers are required for new features on ePWM Type 4. The ePWM register address space has been remapped for better alignment and easy usage.
- **Delayed Trip Functionality**

  Changes have been added to achieve deadband insertion capabilities to support, for example, delayed trip functionality needed for peak current mode control type application scenarios.
- **Dead-Band Generator Submodule Enhancements**

  Shadowing of DBCTL register to allow dynamic configuration changes.
- **One Shot and Global Reload of Registers**

  The ePWM Type 4 allows one shot and global reload capability from shadow to active registers to avoid partial reloads in, for example, multi-phase applications. It also allows a programmable prescale of shadow to active reload events.
- **Trip Zone Submodule Enhancements**

  Independent flags have been added to reflect the trip status for each of the TZ sources. Changes have been made to the trip zone submodule to support certain power converter switching techniques like valley switching.
- **Digital Compare Submodule Enhancements**

  Blanking window filter register width has been increased from 8 to 16 bits. DCCAP functionality has been enhanced to provide more programmability.
- **PWM SYNC Related Enhancements**

  The ePWM Type 4 allows PWM SYNCOUT generation based on CMPC and CMPD events. These events can also be used for PWMSYNC pulse selection.

The ePWM Type 2 is fully compatible to Type 1. Type 2 has the following enhancements in addition to the Type 1 features:

- **High Resolution Dead-Band Capability**

  High resolution capability is added to dead-band RED and FED in half-cycle clocking mode.
- **Dead-Band Generator Submodule Enhancements**

  The ePWM Type 2 has features to enable both RED and FED on either PWM outputs. Provides increased dead band with 14-bit counters and dead-band / dead-band high-resolution registers are shadowed
- **High Resolution Extension available on ePWMxB outputs**

  Provides the ability to enable high-resolution period and duty cycle control on ePWMxB outputs. This is discussed in more detail in the *HRPWM* chapter in this manual.
- **Counter Compare Submodule Enhancements**

  The ePWM Type 2 allows Interrupts and SOC events to be generated by additional counter compares CMPC and CMPD.

- **Event Trigger Submodule Enhancements**

  Prescaling logic to issue interrupt requests and ADC start of conversion expanded up to every 15 events. It allows Software initialization of event counters on SYNC event.
- **Digital Compare Submodule Enhancements**

  Digital Compare Trip Select logic [DCTRIPSEL] has up to 12 external trip sources selected by the Input X-BAR logic in addition to an ability to OR all of them (up to 14 [external and internal sources]) to create the respective DCxEVTs.
- **Simultaneous Writes to TBPRD and CMPx Registers**

  This feature allows writes to TBPRD, CMPA:CMPAHR, CMPB:CMPBHR, CMPC and CMPD of any ePWM submodule to be tied to any other ePWM submodule, and also allows all ePWM submodules to be tied to a particular ePWM submodule if desired.
- **Shadow to Active Load on SYNC of TBPRD and CMP Registers**

  This feature supports simultaneous writes of TBPRD and CMPA/B/C/D registers.

An effective PWM peripheral must be able to generate complex pulse width waveforms with minimal CPU overhead or intervention. It needs to be highly programmable and very flexible while being easy to understand and use. The ePWM unit described here addresses these requirements by allocating all needed timing and control resources on a per PWM channel basis. Cross coupling or sharing of resources has been avoided; instead, the ePWM is built up from smaller single channel submodules with separate resources that can operate together as required to form a system. This modular approach results in an orthogonal architecture and provides a more transparent view of the peripheral structure, helping users to understand its operation quickly.

In this document, the letter x within a signal or submodule name is used to indicate a generic ePWM instance on a device. For example, output signals EPWMxA and EPWMxB refer to the output signals from the ePWMx instance. Thus, EPWM1A and EPWM1B belong to ePWM1 and likewise EPWM4A and EPWM4B belong to ePWM4.

**Type0 to Type1 Enhancements**

- **Increased Dead-Band Resolution**

  Dead-band clocking has been enhanced to allow half-cycle clocking to double resolution.
- **Enhanced Interrupt and SOC Generation**

  Interrupts and ADC start-of-conversion can now be generated on both the TBCTR == zero and TBCTR == period events. This feature enables dual edge PWM control. Additionally, the ADC start-of-conversion can be generated from an event defined in the digital compare submodule.
- **High Resolution Period Capability**

  Provides the ability to enable high-resolution period. This is discussed in more detail in the device-specific HRPWM Reference Guide.
- **Digital Compare Subodule**

  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.

### *18.1.1 Submodule Overview*

The ePWM submodule represents one complete PWM channel composed of two PWM outputs: EPWMxA and EPWMxB. Multiple ePWM submodules are instanced within a device as shown in Figure 18-1. Each ePWM instance is identical with one exception. Some instances include a hardware extension that allows more precise control of the PWM outputs. This extension is the high-resolution pulse width modulator (HRPWM) and is described in the device-specific *High-Resolution Pulse Width Modulator (HRPWM)* chapter. See the device-specific data manual to determine which ePWM instances include this feature. Each ePWM submodule 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.

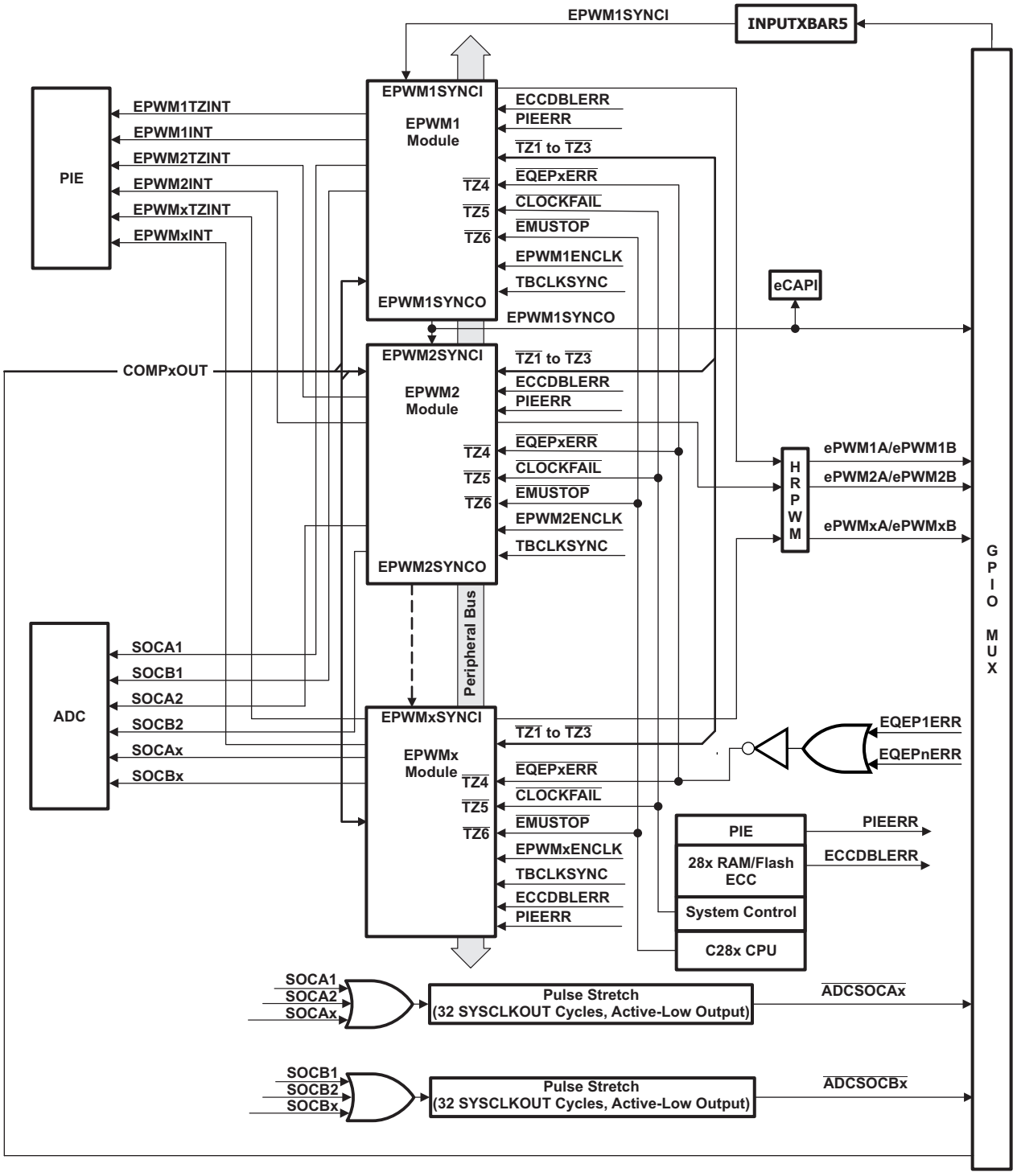
The ePWM submodules are chained together via a clock synchronization scheme that allows them to operate as a single system when required. Additionally, this synchronization scheme can be extended to the capture peripheral submodules (eCAP). The number of submodules is device-dependent and based on target application needs. Submodules can also operate standalone.

Each ePWM submodule 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 submodules.
- 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 submodule is connected to the input/output signals shown in Figure 18-1. The signals are described in detail in subsequent sections.

**Figure 18-1. Multiple ePWM Modules**




A This signal exists only on devices with an eQEP submodule.

The order in which the ePWM submodules are connected may differ from what is shown in Figure 18-1. See Section 18.3.2.3.3 for the synchronization scheme for a particular device. Each ePWM submodule consists of eight submodules and is connected within a system via the signals shown in Figure 18-2.

**Figure 18-2. Submodules and Signal Connections for an ePWM Module**

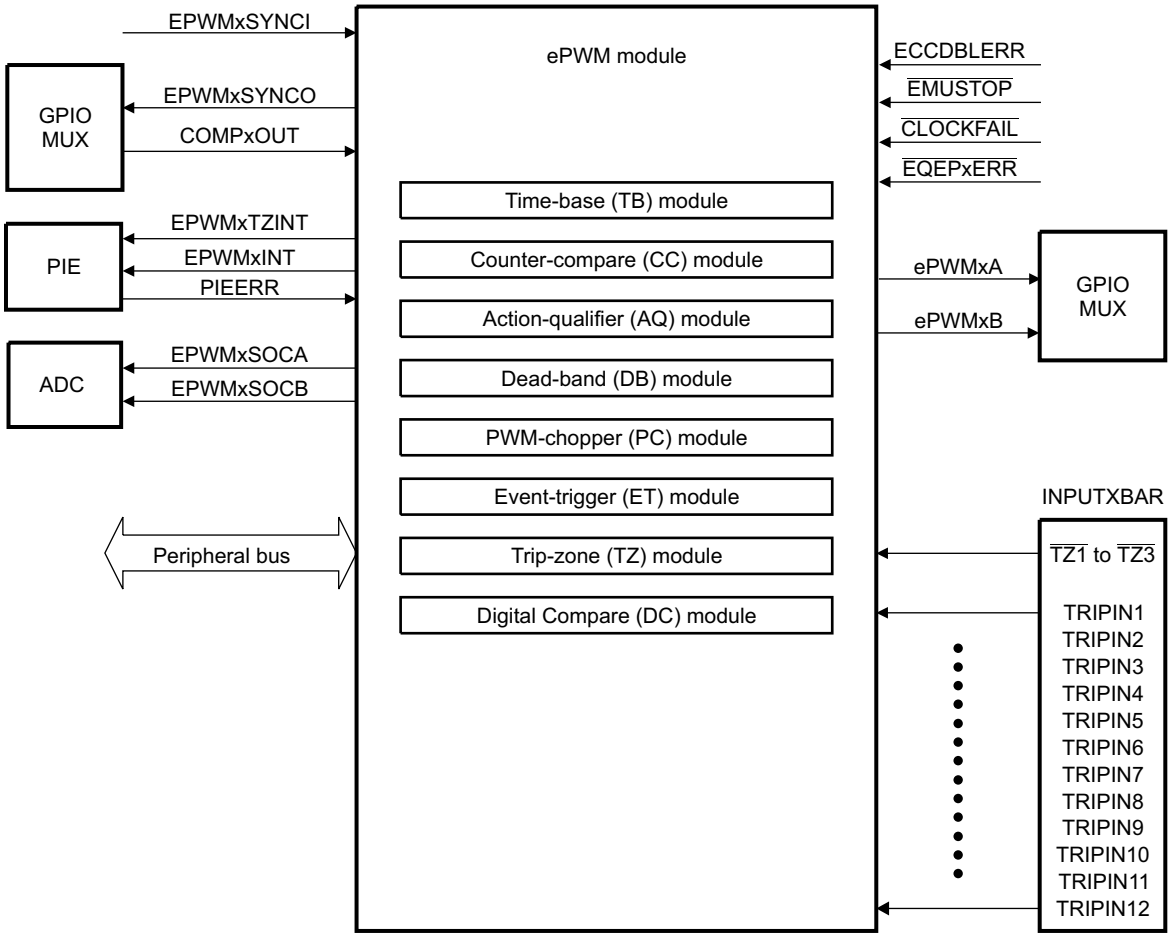



Figure 18-3 shows more internal details of a single ePWM module. The main signals used by the ePWM submodule are:

- **PWM output signals (EPWMxA and EPWMxB).**

  The PWM output signals are made available external to the device through the GPIO peripheral described in the *System Control and Interrupts* chapter for your device.

- **Trip-zone signals ($\overline{TZ1}$ to $\overline{TZ6}$).**

  These input signals alert the ePWM submodule 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 18-49. $\overline{TZ4}$ 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 you to configure a trip action when the clock fails or the CPU halts.

- **Time-base synchronization input (EPWMxSYNCI) and output (EPWMxSYNCO) signals**.

  The synchronization signals daisy chain the ePWM submodules together. Eachsubmodule can be configured via INPUTXBAR6 to either use or ignore its synchronization input. The clock synchronization input and output signal are brought out to pins only for ePWM1 (ePWM module #1). The ePWM submodules are separate into groups of three for syncing purposes. An external sync signal (EXTSYNCIN1 or EXTSYNCIN2) may be used to issue a sync signal to the first ePWM submodule in each chain. These same submodules can also send their EPWMxSYNCOUT signal to a GPIO. For more information, see Section 18.3.2.3.3.

- **ADC start-of-conversion signals (EPWMxSOCA and EPWMxSOCB).**

Each ePWM submodule has two ADC start of conversion signals. Any ePWM submodule can trigger a start of conversion. Whichever event triggers the start of conversion is configured in the event-trigger submodule of the ePWM.

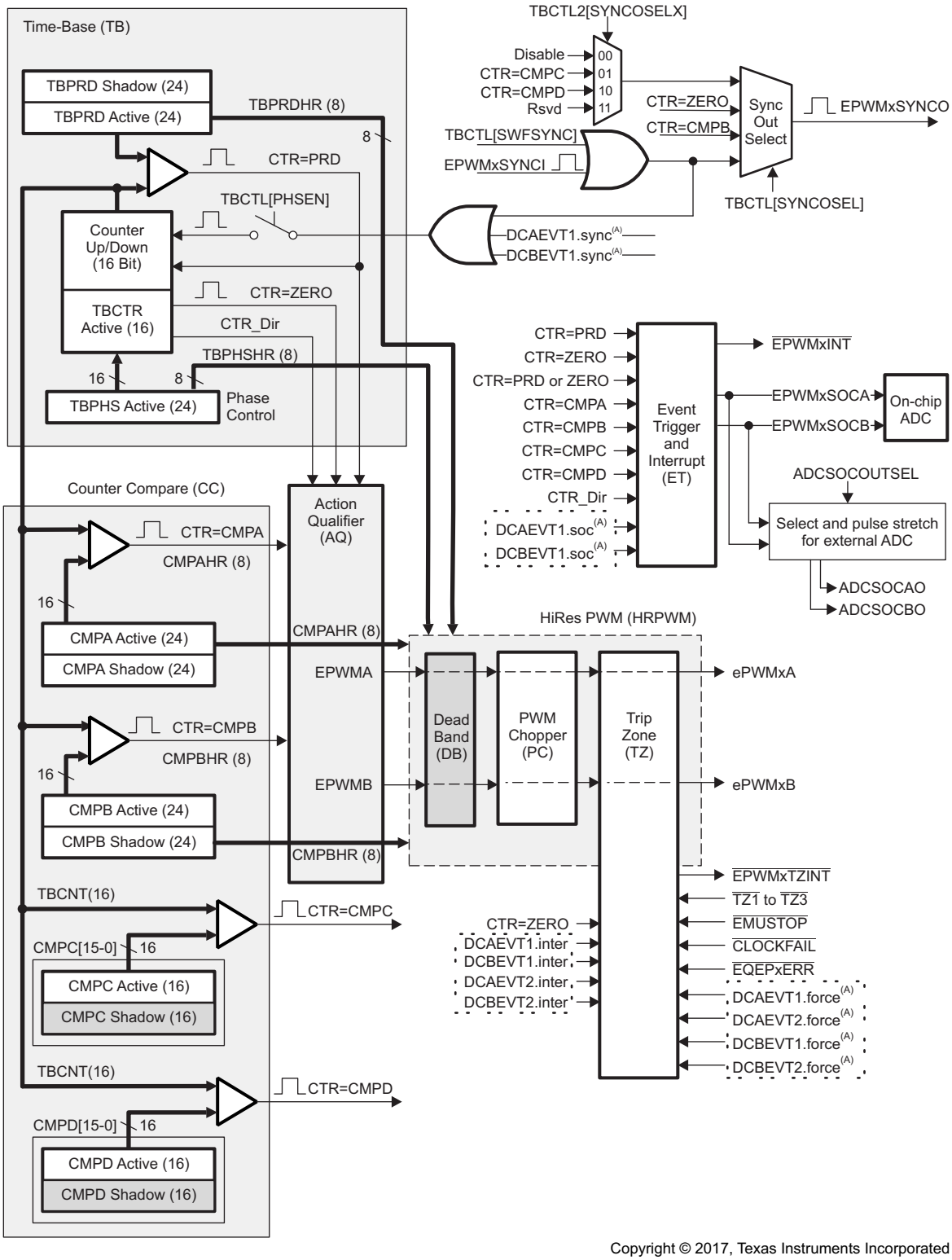
- **Comparator output signals (COMPxOUT).**

  Output signals from the comparator module can be fed through the Input X-BAR to one or all of the 12 trip inputs [TRIPIN1 - TRIPIN12] and in conjunction with the trip zone signals can generate digital compare events.

- **Peripheral Bus**

  The peripheral bus is 32-bits wide and allows both 16-bit and 32-bit writes to the ePWM register file.

**Figure 18-3. ePWM Submodules and Critical Internal Signal Interconnects**






A --*These events are generated by the ePWM digital compare (DC) submodule based on the levels of the TRIPIN inputs.

B This signal exists only on devices with the eQEP module

## 18.2 Configuring Device Pins

To connect the device input pins to the module, the Input X-BAR must be used. Some examples of when an external signal may be needed are TZx, TRIPx, and EXTSYNCIN. Any GPIO on the device can be configured as an input. The GPIO input qualification should be set to asynchronous mode by setting the appropriate GPxQSEL register bits to 11b. The internal pullups can be configured in the GPyPUD register. Since the GPIO mode is used, the GPyINV register can invert the signals. Additionally, some TRIPx (TRIP4-12 excluding TRIP6) signals must be routed through the EPWM X-Bar in addition to the Input X-Bar.

The GPIO mux registers must be configured for this peripheral. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

See the *GPIO* chapter for more details on GPIO mux, GPIO settings, and XBAR configuration.

## 18.3 ePWM Submodules

Eight submodules are included in every ePWM peripheral. Each of these submodules performs specific tasks that can be configured by software.

### *18.3.1 Overview*

Table 18-1 lists the eight key submodules together with a list of their main configuration parameters. For example, if you need to adjust or control the duty cycle of a PWM waveform, then you should see the counter-compare submodule in Section 18.3.3 for relevant details.

**Table 18-1. 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 will behave 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 reload of registers in this module. |
| Counter Compare (CC) | • Specify the PWM duty cycle for output EPWMxA and/or 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 reload of registers in this module. |

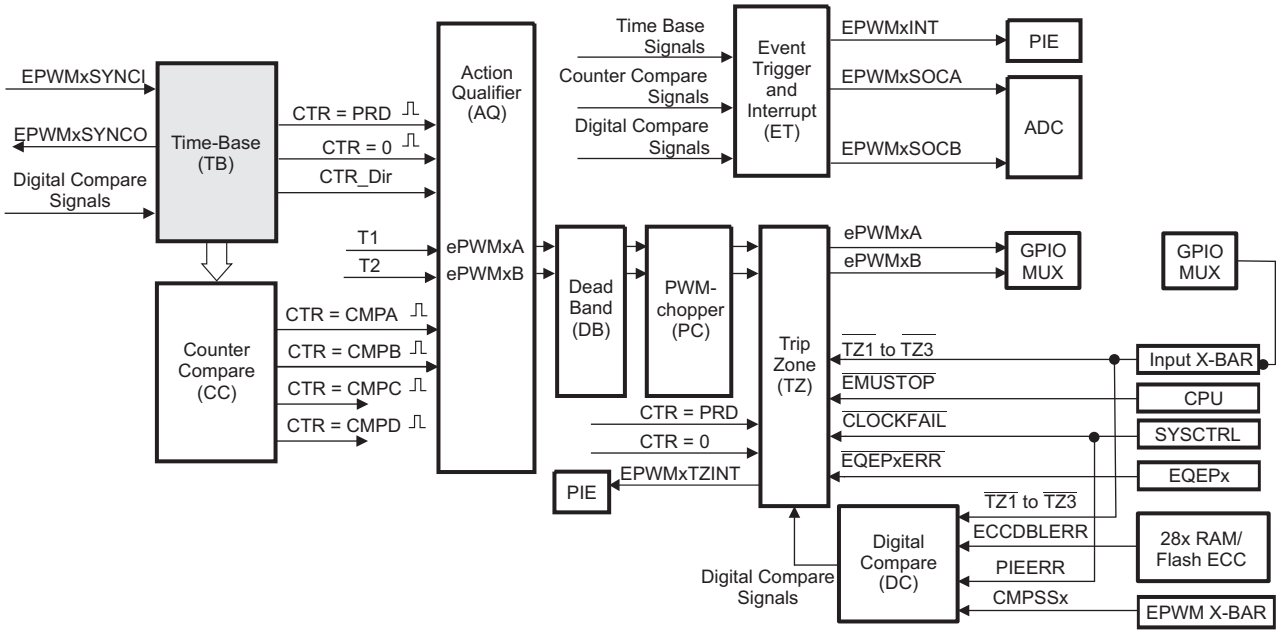
**Table 18-1. Submodule Configuration Parameters (continued)**

| 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/or EPWMxB switched high<br>– Output EPWMxA and/or EPWMxB switched low<br>– Output EPWMxA and/or 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 reload of registers in this module. |
| Dead Band (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 reload 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/or EPWMxB high<br>– Force EPWMxA and/or EPWMxB low<br>– Force EPWMxA and/or EPWMxB to a high-impedance state<br>– Configure EPWMxA and/or EPWMxB to ignore any trip condition.<br>• Configure how often the ePWM will react 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 will trigger an interrupt.<br>• Enable ePWM events that will 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. |

### 18.3.2 Time-Base (TB) Submodule

Each ePWM module has its own time-base submodule that determines all of the event timing for the ePWM module. Built-in synchronization logic allows the time-base of multiple ePWM modules to work together as a single system. Figure 18-4 illustrates the time-base module's place within the ePWM.

**Figure 18-4. Time-Base Submodule**




#### 18.3.2.1 Purpose of the Time-Base Submodule

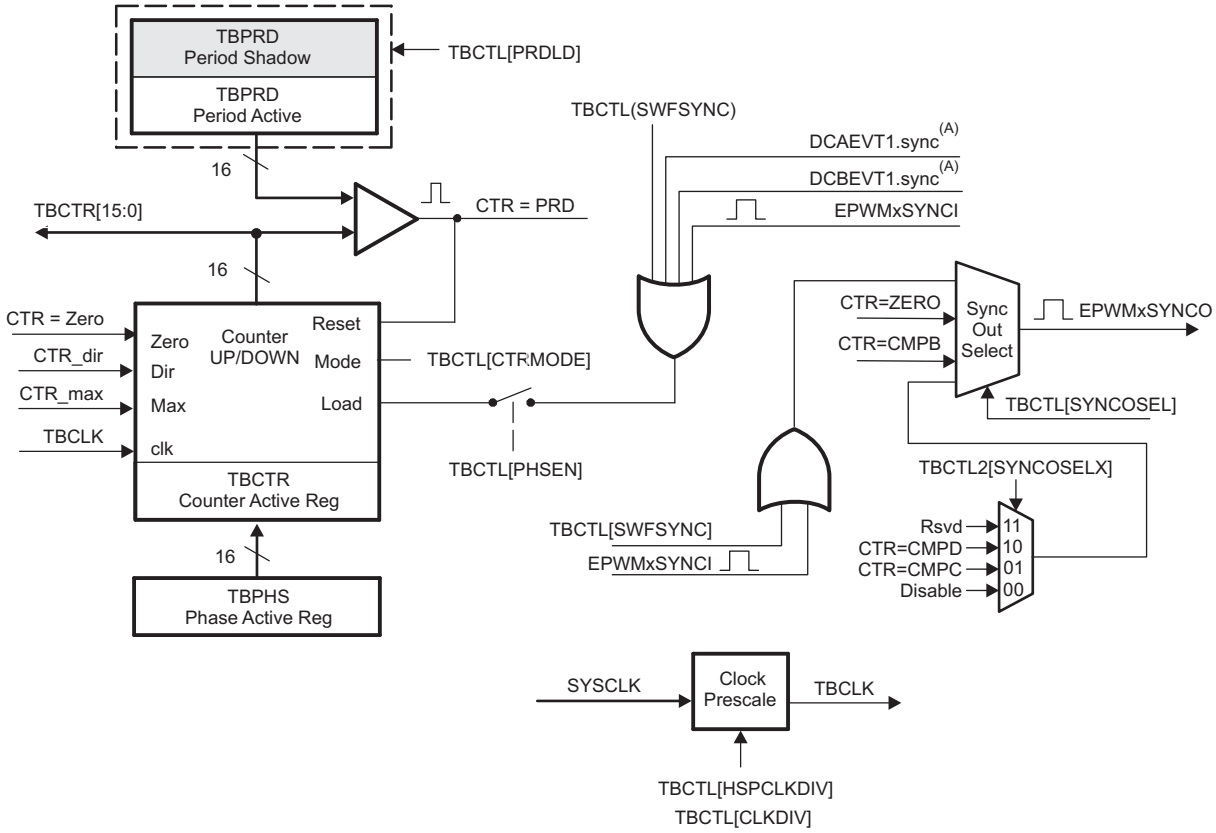
You can configure the time-base submodule for the following:

- Specify the ePWM time-base counter (TBCTR) frequency or period to control how often events occur.
- Manage time-base synchronization with other ePWM modules.
- Maintain a phase relationship with other ePWM modules.
- Set the time-base counter to count-up, count-down, or count-up-and-down mode.
- Generate the following events:
  - CTR = PRD: Time-base counter equal to the specified period (TBCTR = TBPRD).
  - CTR = Zero: Time-base counter equal to zero (TBCTR = 0x00).
- Configure the rate of the time-base clock; a prescaled version of the EPWM clock (EPWMCLK). This allows the time-base counter to increment/decrement at a slower rate.

### 18.3.2.2 Controlling and Monitoring the Time-Base Submodule

The block diagram in Figure 18-5 shows the critical signals and registers of the time-base submodule. Table 18-2 provides descriptions of the key signals associated with the time-base submodule.

**Figure 18-5. Time-Base Submodule Signals and Registers**




A. These signals are generated by the digital compare (DC) submodule.

**Table 18-2. Key Time-Base Signals**

| Signal | Description |
|---|---|
| EPWMxSYNCI | Time-base synchronization input.<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 may come from a device pin via INPUT5 or INPUT6 of the Input X-BAR or from a previous ePWM module. For subsequent ePWM modules in each chain, this signal is passed from another ePWM peripheral. For example, EPWM2SYNCI is generated by the ePWM1 peripheral, EPWM3SYNCI is generated by ePWM2 and so forth. For information on the synchronization order of a particular device, see Section 18.3.2.3.3. |
| 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. |
| 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. |

**Table 18-2. Key Time-Base Signals (continued)**

| Signal | Description |
|---|---|
| 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. This signal is high when the counter is increasing and low when it is decreasing. |
| CTR_max | Time-base counter equal max value. (TBCTR = 0xFFFF)<br>Generated event when the TBCTR value reaches its 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. |

#### 18.3.2.3 Calculating PWM Period and Frequency

The frequency of PWM events is controlled by the time-base period (TBPRD) register and the mode of the time-base counter. Figure 18-6 shows the period ($T_{pwm}$) and frequency ($F_{pwm}$) relationships for the up-count, down-count, and up-down-count time-base counter modes when the period is set to 4 (TBPRD = 4). The time increment for each step is defined by the time-base clock (TBCLK) which is a prescaled version of the EPWM clock (EPWMCLK).

The time-base counter has three modes of operation selected by the time-base control register (TBCTL):

- **Up-Down-Count Mode:**

  In up-down-count mode, the time-base counter starts from zero and increments until the period (TBPRD) value is reached. When the period value is reached, the time-base counter then decrements until it reaches zero. At this point the counter repeats the pattern and begins to increment.

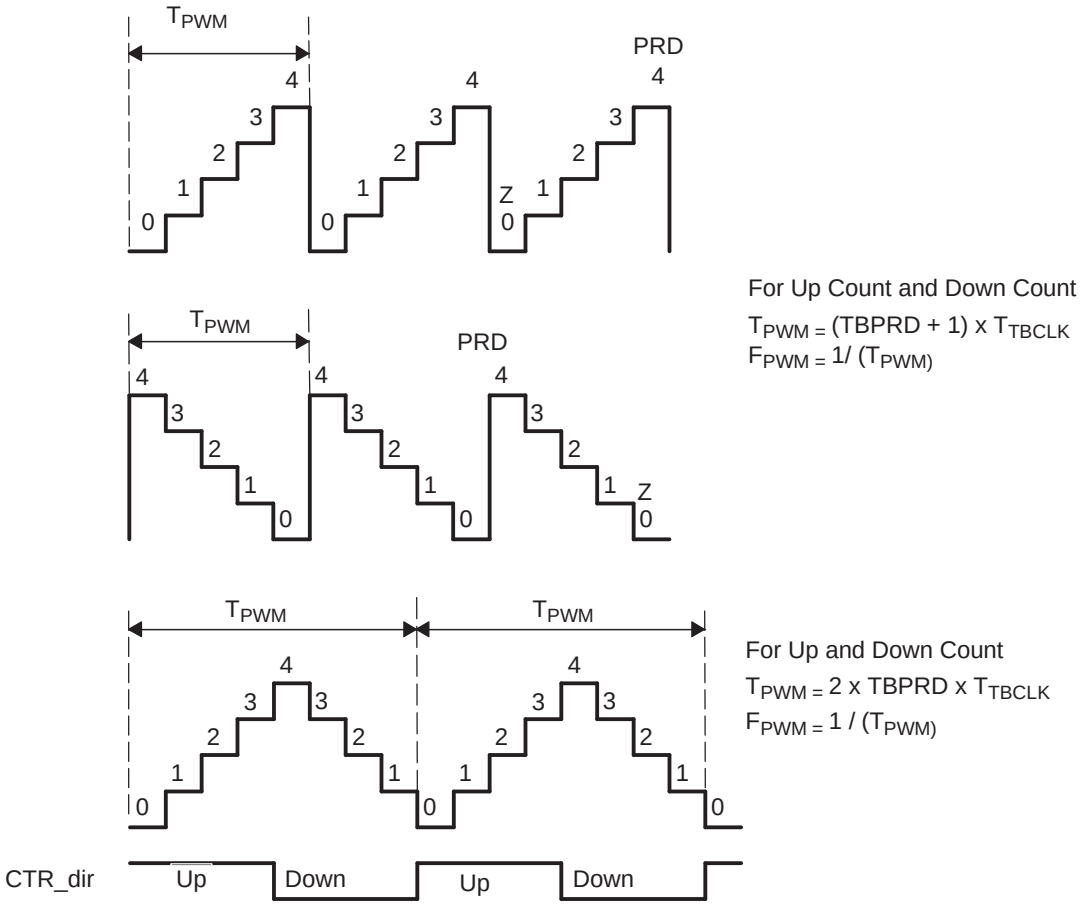
- **Up-Count Mode:**

  In this mode, the time-base counter starts from zero and increments until it reaches the value in the period register (TBPRD). When the period value is reached, the time-base counter resets to zero and begins to increment once again.

- **Down-Count Mode:**

  In down-count mode, the time-base counter starts from the period (TBPRD) value and decrements until it reaches zero. When it reaches zero, the time-base counter is reset to the period value and it begins to decrement once again.

**Figure 18-6. Time-Base Frequency and Period**




### 18.3.2.3.1 Time-Base Period Shadow Register

The time-base period register (TBPRD) has a shadow register. Shadowing allows the register update to be synchronized with the hardware. The following definitions are used to describe all shadow registers in the ePWM module:

- **Active Register**

  The active register controls the hardware and is responsible for actions that the hardware causes or invokes.

- **Shadow Register**

  The shadow register buffers or provides a temporary holding location for the active register. It has no direct effect on any control hardware. At a strategic point in time the shadow register's content is transferred to the active register. This prevents corruption or spurious operation due to the register being asynchronously modified by software.

The memory address of the shadow period register is the same as the active register. Which register is written to or read from is determined by the TBCTL[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 18.3.2.3.3.

The global reload control mechanism can also be used with the time-base period register by configuring the appropriate bits in the global load configuration register (GLDCFG). When global reload mode is selected the transfer of contents from shadow register to active register, for all registers that have this mode enabled, occurs at the same event as defined by the configuration bits in Global Shadow to Active Load Control Register (GLDCTL). Global reload control mechanism is explained in Section 18.3.2.7.

- **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.

#### *18.3.2.3.2 Time-Base Clock Synchronization*

The TBCLKSYNC bit in the peripheral clock enable registers allows all users to globally synchronize all enabled ePWM modules to the time-base clock (TBCLK). When set, all enabled ePWM module clocks are started with the first rising edge of TBCLK aligned. For perfectly synchronized TBCLKs, the prescalers for each ePWM module must be set identically.

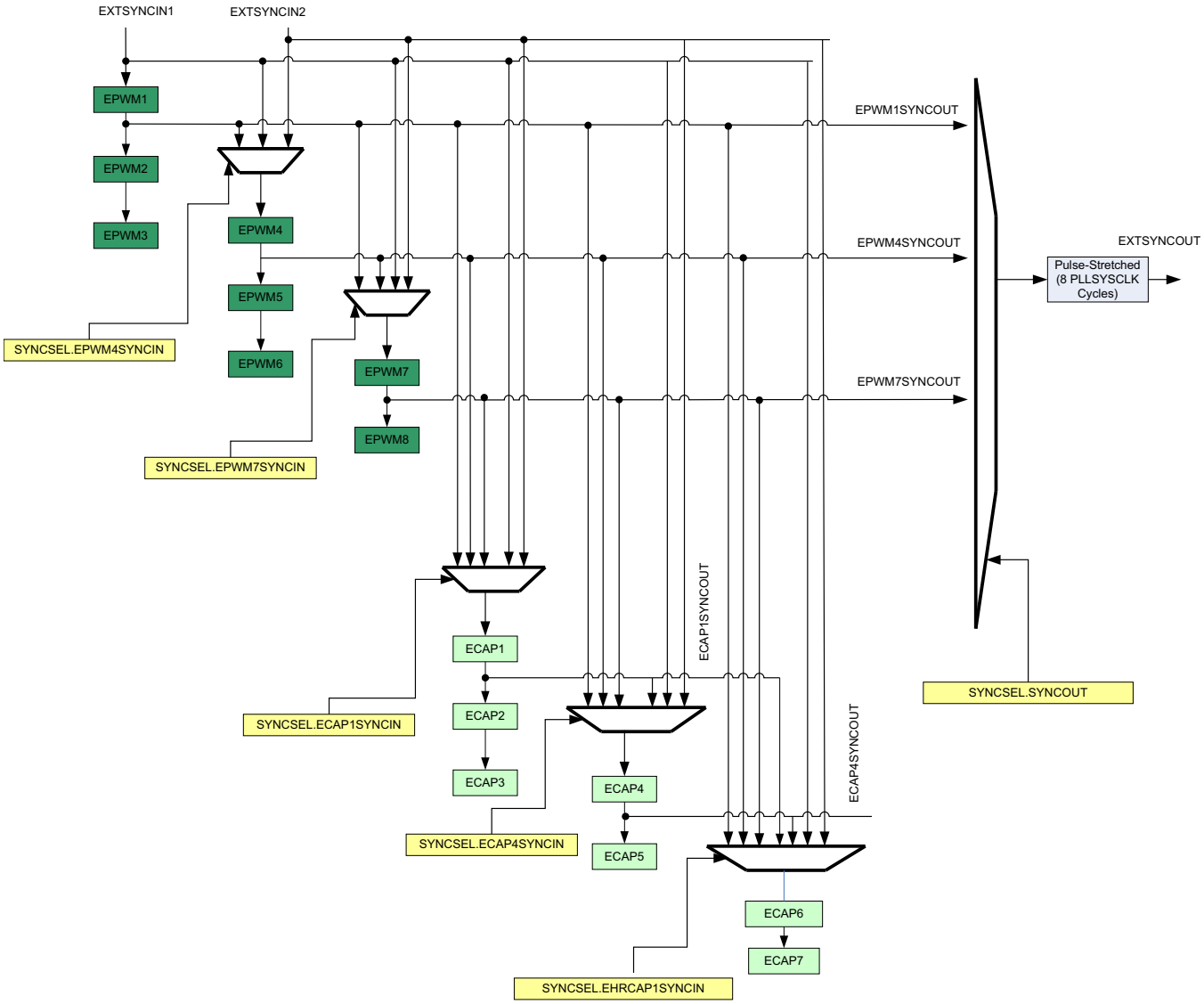
The proper procedure for enabling ePWM clocks is as follows:

1. Enable ePWM module clocks in the PCLKCRx register
2. Set TBCLKSYNC= 0
3. Configure ePWM modules
4. Set TBCLKSYNC=1

#### *18.3.2.3.3 Time-Base Counter Synchronization*

The ePWM type 4 introduces a new synchronization scheme that allows for increased flexibility of synchronization of the ePWM modules. Each ePWM module has a synchronization input (SYNCI) and a synchronization output (SYNCO). In Figure 20-8, EXTSYNC1 is sourced from INPUTXBAR5 and EXTSYNC2 is sourced from INPUTXBAR6, which can be configured to select any GPIO as the synchronization input. When configuring the sync chain propagation path using the SYNCSEL registers, make sure that the longest path does not exceed four ePWM/eCAP modules.

**Figure 18-7. Time-Base Counter Synchronization Scheme**




**NOTE:** See the data manual for the number of ePWM and eCAP modules available on your specific device.

Each ePWM module can be configured to use or ignore the synchronization input. If the TBCTL[PHSEN] bit is set, then the time-base counter (TBCTR) of the ePWM module will be automatically loaded with the phase register (TBPHS) contents when one of the following conditions occur:

- **EPWMxSYNCI: Synchronization Input Pulse:**

  The value of the phase register is loaded into the counter register when an input synchronization pulse is detected (TBPHS → TBCTR). This operation occurs on the next valid time-base clock (TBCLK) edge.

  The delay from internal master module to slave modules is given by:

  – if ( TBCLK = EPWMCLK): 2 x EPWMCLK
  – if ( TBCLK = EPWMCLK):1 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.

This feature enables the ePWM module to be automatically synchronized to the time base of another ePWM module. Lead or lag phase control can be added to the waveforms generated by different ePWM modules to synchronize them. In up-down-count mode, the TBCTL[PSHDIR] bit configures the direction of the time-base counter immediately after a synchronization event. The new direction is independent of the direction prior to the synchronization event. The PHSDIR bit is ignored in count-up or count-down modes. See Figure 18-8 through Figure 18-11 for examples.

Clearing the TBCTL[PHSEN] bit configures the ePWM to ignore the synchronization input pulse. The synchronization pulse can still be allowed to flow-through to the EPWMxSYNCO and be used to synchronize other ePWM modules. In this way, you can set up a master time-base (for example, ePWM1) and downstream modules (ePWM2 - ePWMx) may elect to run in synchronization with the master. See Section 18.4 for more details on synchronization strategies.

#### 18.3.2.4 Phase Locking the Time-Base Clocks of Multiple ePWM Modules

The TBCLKSYNC bit can be used to globally synchronize the time-base clocks of all enabled ePWM modules on a device. This bit is part of the device's clock enable registers and is described in the *System Control and Interrupts* section of this manual. When TBCLKSYNC = 0, the time-base clock of all ePWM modules is stopped (default). When TBCLKSYNC = 1, all ePWM time-base clocks are started with the rising edge of TBCLK aligned. For perfectly synchronized TBCLKs, the prescaler bits in the TBCTL register of each ePWM module must be set identically. The proper procedure for enabling the ePWM clocks is as follows:

1. Enable the individual ePWM module clocks. This is described in the *System Control and Interrupts* chapter.
2. Set TBCLKSYNC = 0. This will stop the time-base clock within any enabled ePWM module.
3. Configure the prescaler values and desired ePWM modes.
4. Set TBCLKSYNC = 1.

#### 18.3.2.5 Simultaneous Writes to TBPRD and CMPx Registers Between ePWM Modules

For variable frequency applications, there is a need for simultaneous writes of TBPRD and CMPx registers between ePWM modules. This prevents situations where a CTR = 0 or CTR = PRD pulse forces a shadow to active load of these registers before all registers are updated between ePWM modules (resulting in some registers being loaded from new shadow values while others are loaded from old shadow values). To support this, an ePWM register linking scheme for TBPRD:TBPRDHR, CMPA:CMPAHR, CMPB:CMPBHR, CMPC, and CMPD registers between PWM modules has been added.

For a particular ePWM module # A , user code writes “B+1”, to the linked register bit-field in EPWMXLINK. “B” is the ePWM module # being linked to (that is, writes to the ePWM module “B” TBPRD:TBPRDHR, CMPA:CMPAHR, CMPB:CMPBHR, or CMPC will simultaneously be written to corresponding register in ePWM module “A”). For instance if ePWM3 EPWMXLINK register is configured so that CMPA:CMPAHR are linked to ePWM1, then a write to CMPA:CMPAHR in ePWM 1 will simultaneously write the same value to CMPA:CMPAHR in ePWM3. If ePWM4 also has its CMPA:CMPAHR registers linked to ePWM1, then a write to ePWM 1 will write the same value to the CMPA:CMPAHR registers in both ePWM3 and ePWM4.

The register description for EPWMXLINK clearly explains the linked register bit-field values for corresponding ePWM.

#### 18.3.2.6 Time-Base Counter Modes and Timing Waveforms

The time-base counter operates in one of four modes:

- Up-count mode which is asymmetrical
- Down-count mode which is asymmetrical
- Up-down-count which is symmetrical

- Frozen where the time-base counter is held constant at the current value

To illustrate the operation of the first three modes, the following timing diagrams show when events are generated and how the time-base responds to an EPWMxSYNCI signal.

**Figure 18-8. Time-Base Up-Count Mode Waveforms**

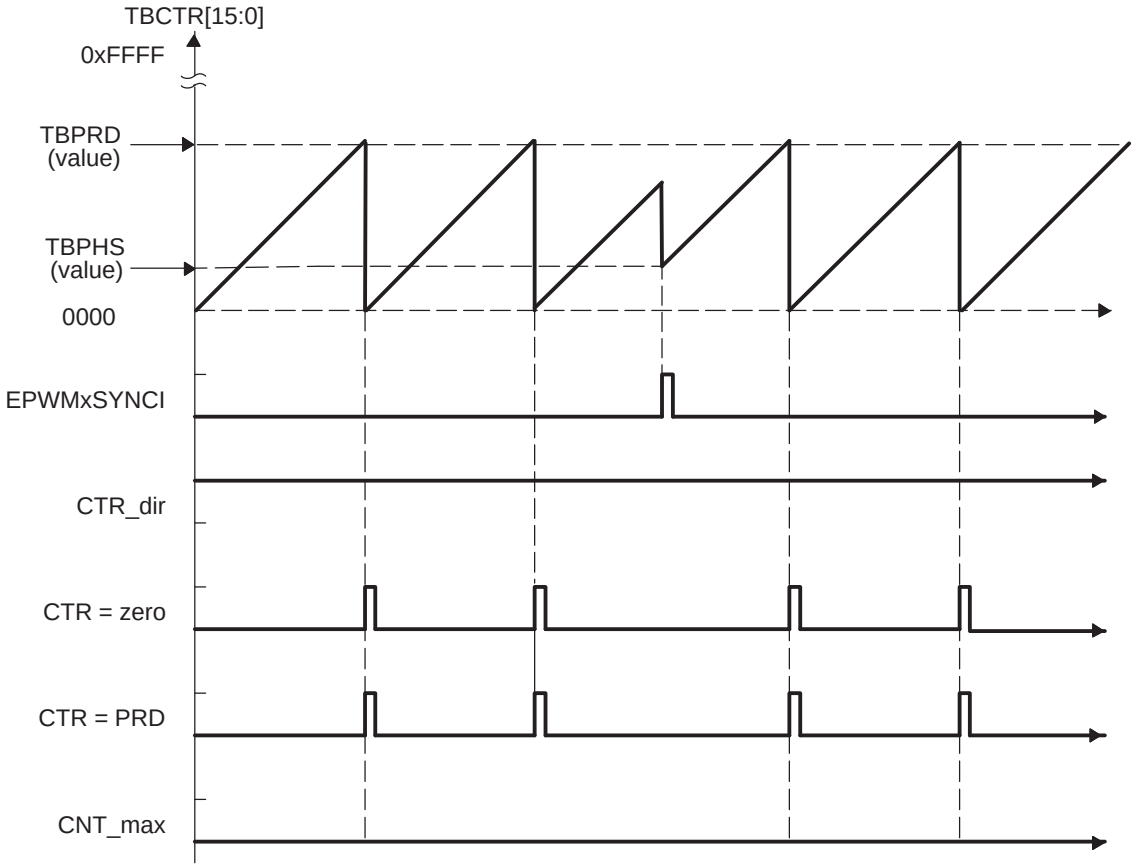


**Figure 18-9. Time-Base Down-Count Mode Waveforms**

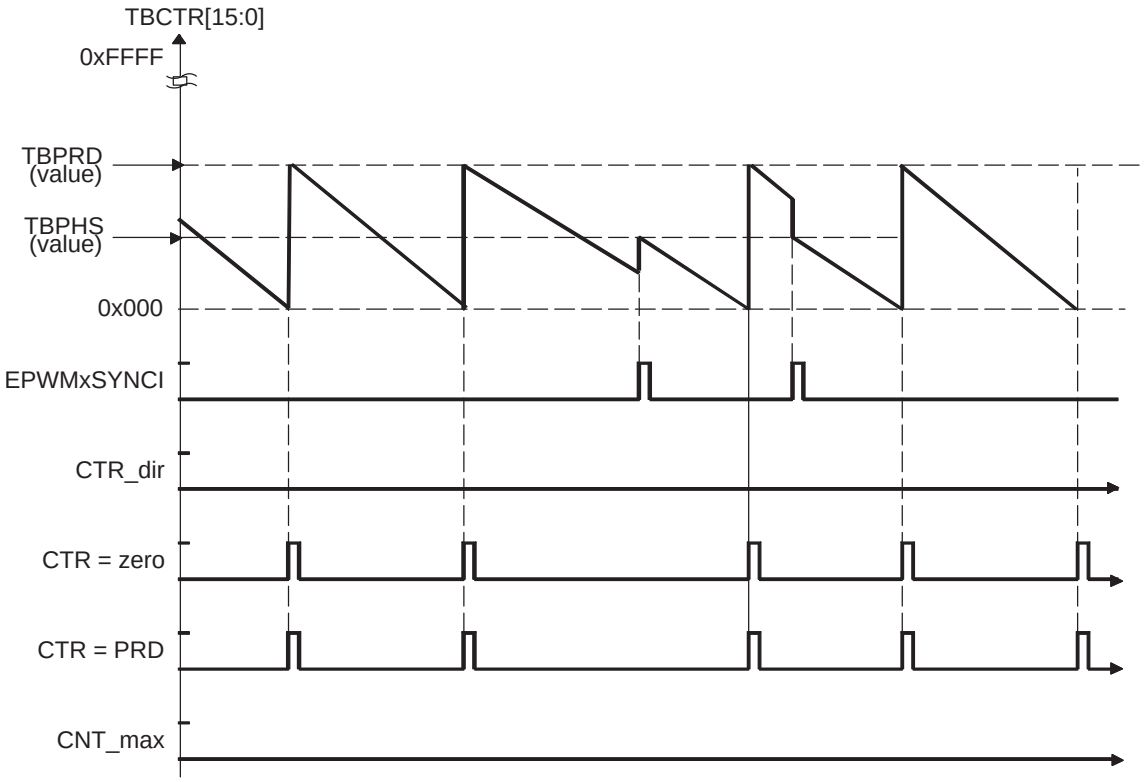



**Figure 18-10. Time-Base Up-Down-Count Waveforms, TBCTL[PHSDIR = 0] Count Down On Synchronization Event**

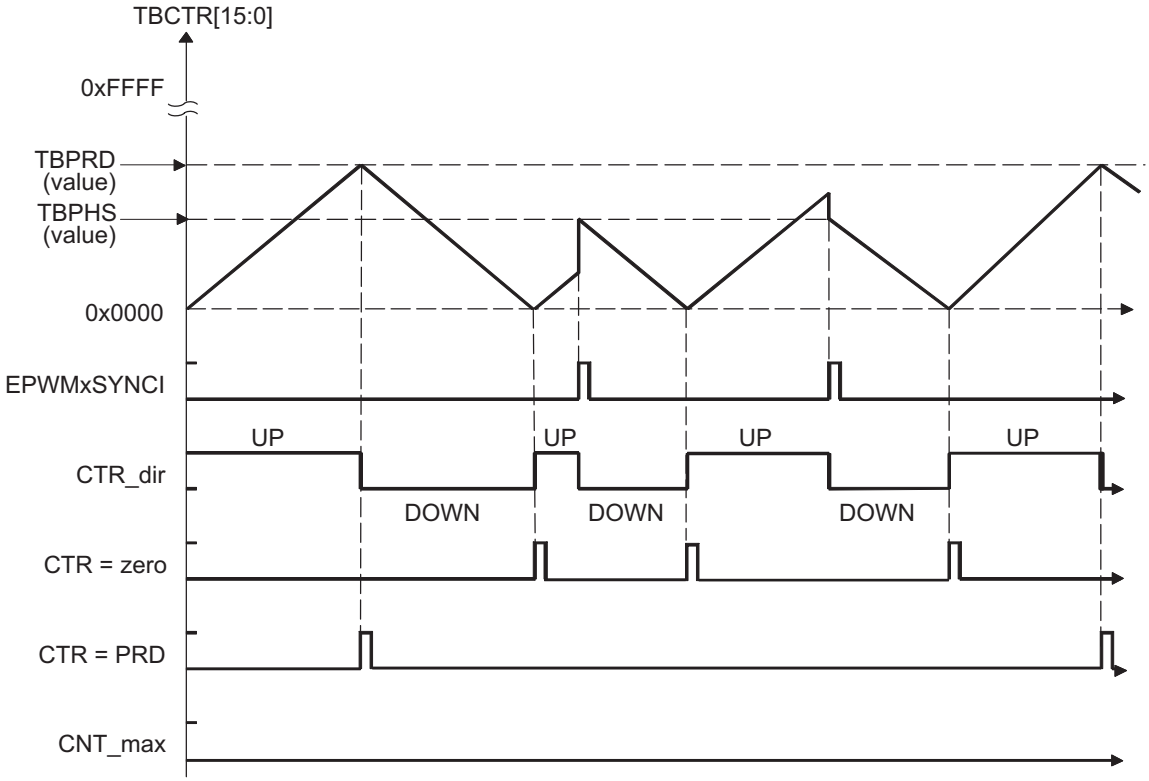
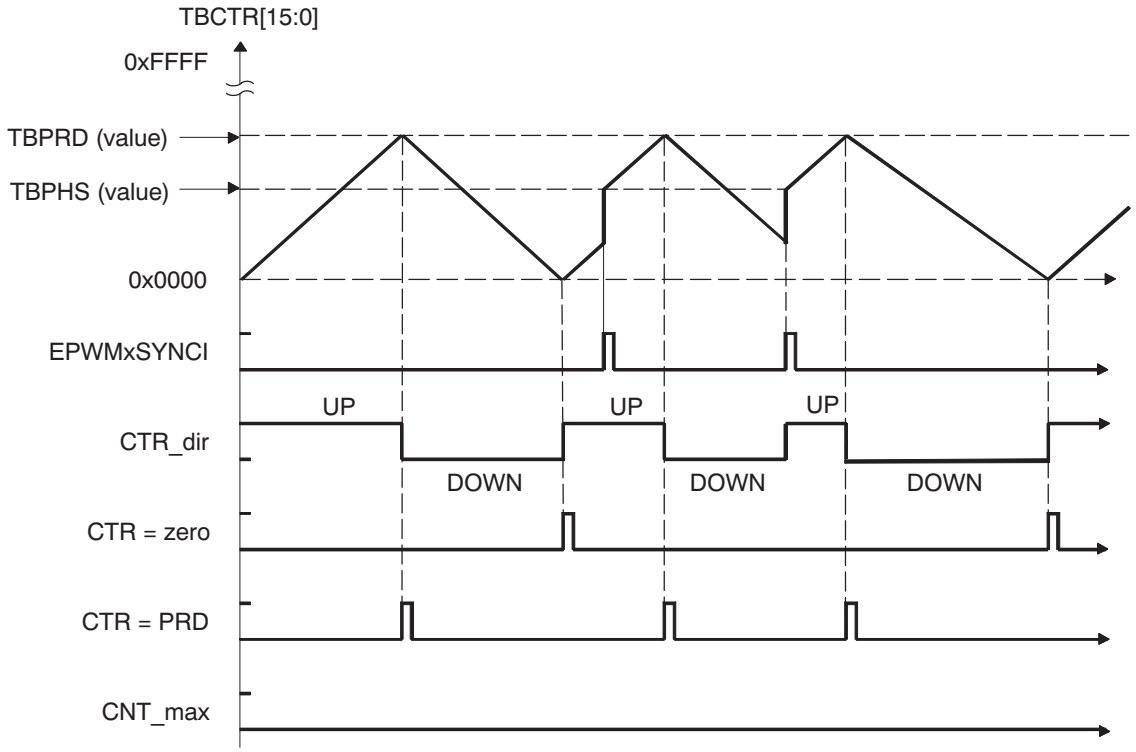


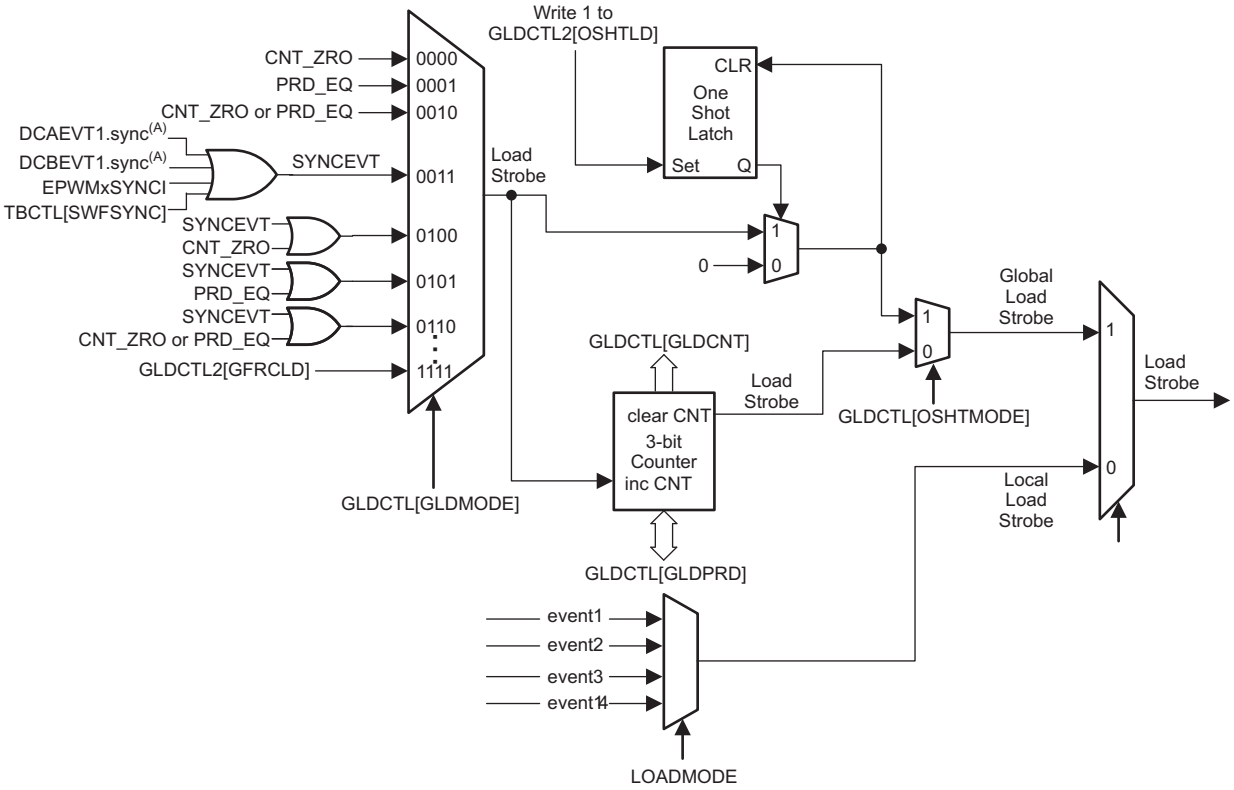
**Figure 18-11. Time-Base Up-Down Count Waveforms, TBCTL[PHSDIR = 1] Count Up On Synchronization Event**




### 18.3.2.7 Global Reload

Figure 18-12 illustrates the signals and registers associated with the global reload feature.

**Figure 18-12. Global Reload: Signals and Registers**




When this feature is enabled, the transfer of contents from the shadow register to the active register, for all registers that have this mode enabled, occurs at the same event as defined by the configuration bits in Global Shadow to Active Load Control Register (GLDCTL[GLDMODE]). When GLDCTL[GLD] = ’1’, shadow to active load event selection bits for individual shadowed registers are ignored and global load mode takes effect for the corresponding registers enabled by GLDCFG[REGx].

When GLDCTL[GLD] = ’1’ and GLDCFG[REGx] = ‘0’ global load mode does not affect the corresponding register (REGx). Shadow to active load event selection bits for individual shadowed registers decide how the transfer of contents from shadow register to active register takes place.

#### *18.3.2.7.1 Global Reload Pulse Pre-Scalar*

This feature provides the capability to choose shadow to active transfers to happen once in ‘N’ occurrences of selected global reload pulse (GLDCTL[GLDMODE]). This pre-scale functionality is not available for registers that cannot or are not configured to use the global load mechanism (that is, GLDCTL[GLD] = ’0’ or GLDCFG[REGx] = ‘0’).

#### *18.3.2.7.2 One-Shot Reload Mode*

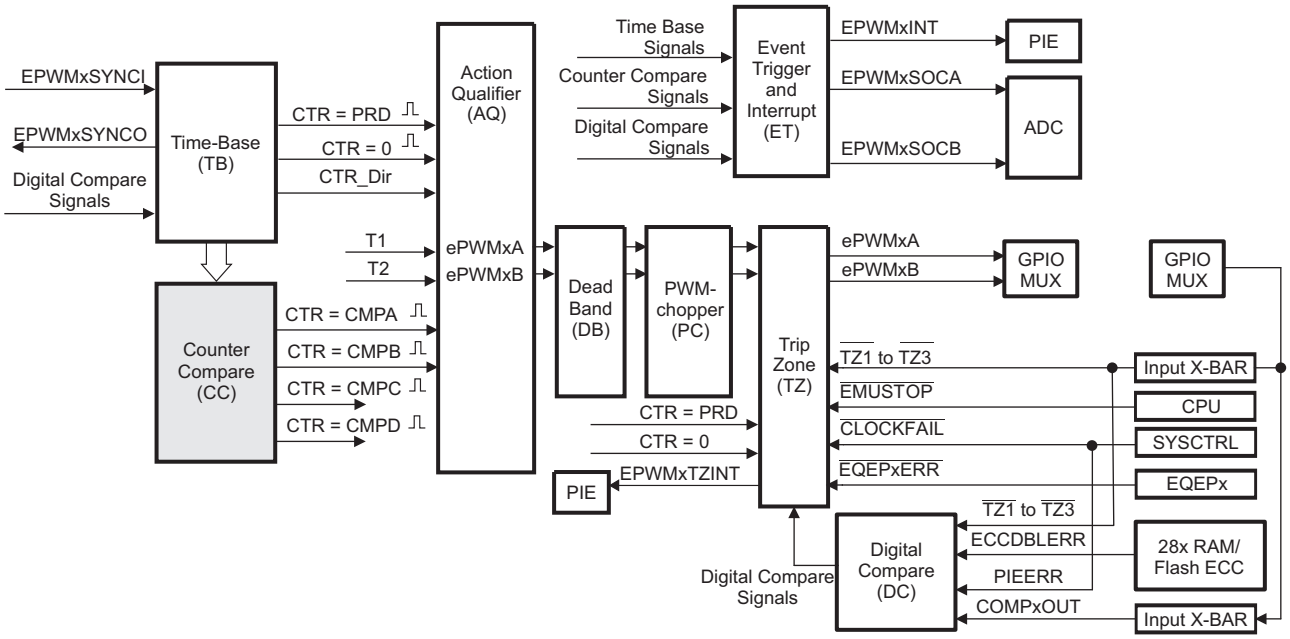
This feature allows users to cause the shadow register to active register transfers to occur once. When GLDCTL2[OSHTLD] = ‘1’ the shadow to active register transfer, for registers that are configured to use the global load mechanism, takes place on the event selected by GLDCTL[GLDMODE].

Software force loading of contents from shadow register to active register is possible by using GLDCTL2[GFRCLD]. The GLDCTL2 register can also be linked across multiple PWM modules by using EPWMXLINK[GLDCTL2LINK]. This, along with the one-shot reload mode feature discussed above, provides a method to correctly update multiple PWM registers in one or more PWM modules at certain PWM events or, if desired, in the same clock cycle. This is very useful in variable frequency applications and/or multi-phase interleaved applications.

### 18.3.3 Counter-Compare (CC) Submodule

Figure 18-13 illustrates the counter-compare submodule within the ePWM.

**Figure 18-13. Counter-Compare Submodule**




#### 18.3.3.1 Purpose of the Counter-Compare Submodule

The counter-compare submodule takes as input the time-base counter value. This value is continuously compared to the counter-compare A (CMPA) counter-compare B (CMPB) counter-compare C (CMPC) and counter-compare D (CMPD )registers. When the time-base counter is equal to one of the compare registers, the counter-compare unit generates an appropriate event.

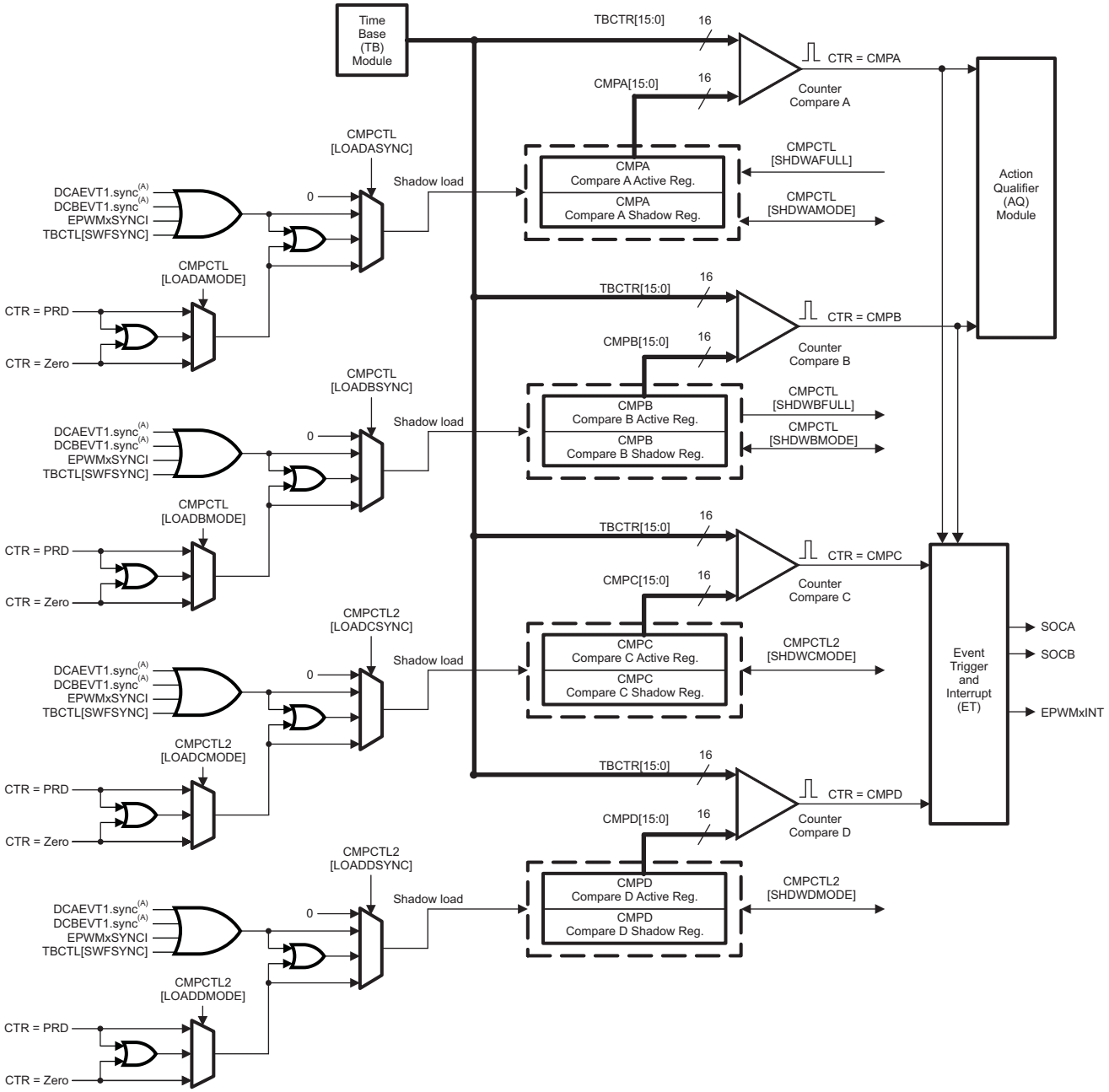
The counter-compare:

- Generates events based on programmable time stamps using the CMPA, CMPB, CMPC and CMPD registers
  - CTR = CMPA: Time-base counter equals counter-compare A register (TBCTR = CMPA)
  - CTR = CMPB: Time-base counter equals counter-compare B register (TBCTR = CMPB)
  - CTR = CMPC: Time-base counter equals counter-compare C register (TBCTR = CMPC)
  - CTR = CMPD: Time-base counter equals counter-compare D register (TBCTR = CMPD)
- Controls the PWM duty cycle if the action-qualifier submodule is configured appropriately using counter-compare A (CMPA) & counter-compare B (CMPB)
- Shadows new compare values to prevent corruption or glitches during the active PWM cycle

#### 18.3.3.2 Controlling and Monitoring the Counter-Compare Submodule

The counter-compare submodule operation is shown in Figure 18-14.

**Figure 18-14. Detailed View of the Counter-Compare Submodule**




A These events are generated by the type 4 ePWM digital compare (DC) submodule based on the levels of the TRIPIN inputs(for example, CMPSSx and $\overline{TZ}$ signals).

### 18.3.3.3 Operational Highlights for the Counter-Compare Submodule

The counter-compare submodule is responsible for generating events which can be used in the action-qualifier and/or event-trigger submodules. There are four independent compare events described below:

1. CTR = CMPA: Time-base counter equal to counter-compare A register (TBCTR = CMPA).
2. CTR = CMPB: Time-base counter equal to counter-compare B register (TBCTR = CMPB).
3. CTR = CMPC: Time-base counter equal to counter-compare C register (TBCTR = CMPC). This event can be used to generate an event in the event trigger submodule only.
4. CTR = CMPD: Time-base counter equal to counter-compare D register (TBCTR = CMPD). This event

can be used to generate an event in the event trigger submodule only

For up-count or down-count mode, each event occurs only once per cycle. For up-down count mode each event occurs twice per cycle if the compare value is between 0x00-TBPRD and once per cycle if the compare value is equal to 0x00 or equal to TBPRD. These events are fed into the action-qualifier submodule where they are qualified by the counter direction and converted into actions if enabled. Refer to Section 18.3.4.1 for more details.

The counter-compare registers CMPA and CMPB each have an associated shadow register. Shadowing provides a way to keep updates to the registers synchronized with the hardware. When shadowing is used, updates to the active registers only occur at strategic points. This prevents corruption or spurious operation due to the register being asynchronously modified by software. The memory address of the active register and the shadow register is identical. Which register is written to or read from is determined by the CMPCTL[SHDWAMODE] and CMPCTL[SHDWBMODE] bits. These bits enable and disable the CMPA shadow register and CMPB shadow register respectively. The behavior of the two load modes is described below:

**Shadow Mode:**

The shadow mode for the CMPA is enabled by clearing the CMPCTL[SHDWAMODE] bit and the shadow register for CMPB is enabled by clearing the CMPCTL[SHDWBMODE] bit. Shadow mode is enabled by default for both CMPA and CMPB.

If the shadow register is enabled then the content of the shadow register is transferred to the active register on one of the following events as specified by the CMPCTL[LOADAMODE] CMPCTL[LOADBMODE] CMPCTL[LOADASYNC] & CMPCTL[LOADBSYNC] register bits:

- CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD).
- CTR = Zero: Time-base counter equal to zero (TBCTR = 0x00)
- Both CTR = PRD and CTR = Zero
- SYNC event caused by DCAEVT1 or DCBEVT1 or 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.

**Immediate Mode:**

If immediate load mode is selected (that is, CMPCTL[SHDWAMODE] = 1 or CMPCTL[SHDWBMODE] = 1), then a read from or a write to the register will go directly to the active register.

**Additional Comparators**

The counter-compare submodule on ePWMs type 2 and later are responsible for generating two additional independent compare events based on two compare registers, which is fed to Event Trigger submodule :

1. CTR = CMPC: Time-base counter equal to counter-compare C register (TBCTR = CMPC).
2. CTR = CMPD: Time-base counter equal to counter-compare D register (TBCTR = CMPD).

The counter-compare registers CMPC and CMPD each have an associated shadow register. By default this register is shadowed. The memory address of the active register and the shadow register is identical. The value in the active CMPC and CMPD register is compared to the time-base counter (TBCTR). When the values are equal, the counter compare module generates a “time-base counter equal to counter compare C or counter compare D ” event respectively. Shadowing of this register is enabled and disabled by the CMPCTL2[SHDWCMODE] and CMPCTL2[SHDWDMODE] bit. These bits enable and disable the CMPA shadow register and CMPB shadow register respectively. The behavior of the two load modes is described below:

**Shadow Mode:**

The shadow mode for the CMPC is enabled by clearing the CMPCTL2[SHDWCMODE] bit and the shadow register for CMPD is enabled by clearing the CMPCTL2[SHDWDMODE] bit. Shadow mode is enabled by default for both CMPC and CMPD.

If the shadow register is enabled then the content of the shadow register is transferred to the active register on one of the following events as specified by the CMPCTL2[LOADCMODE] CMPCTL2[LOADDMODE] CMPCTL2[LOADCSYNC] & CMPCTL2[LOADDSYNC] register bits:

- CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD).
- CTR = Zero: Time-base counter equal to zero (TBCTR = 0x00)
- Both CTR = PRD and CTR = Zero
- SYNC event caused by DCAEVT1 or DCBEVT1 or 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 will go directly to the active register.

**Global Reload Support**

The global reload control mechanism can also be used for all counter-compare registers by configuring the appropriate bits in the global load configuration register (GLDCFG). When the global reload mode is selected the transfer of contents from shadow register to active register, for all registers that have this mode enabled, occurs at the same event as defined by the configuration bits in the Global Shadow to Active Load Control Register (GLDCTL). The global reload control mechanism is explained in Section 18.3.2.7.

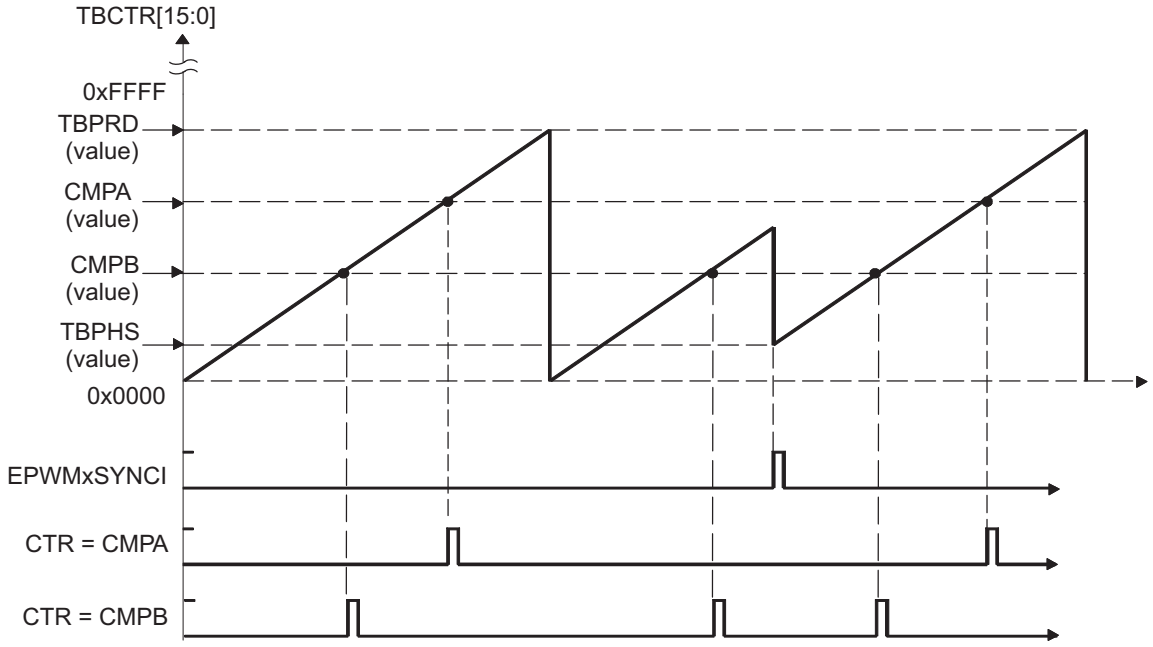
### 18.3.3.4 Count Mode Timing Waveforms

The counter-compare module can generate compare events in all three count modes:

- Up-count mode: used to generate an asymmetrical PWM waveform.
- Down-count mode: used to generate an asymmetrical PWM waveform.
- Up-down-count mode: used to generate a symmetrical PWM waveform.

To best illustrate the operation of the first three modes, the timing diagrams in Figure 18-15 through Figure 18-18 show when events are generated and how the EPWMxSYNCI signal interacts.

**Figure 18-15. Counter-Compare Event Waveforms in Up-Count Mode**




NOTE: 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 18-16. Counter-Compare Events in Down-Count Mode**

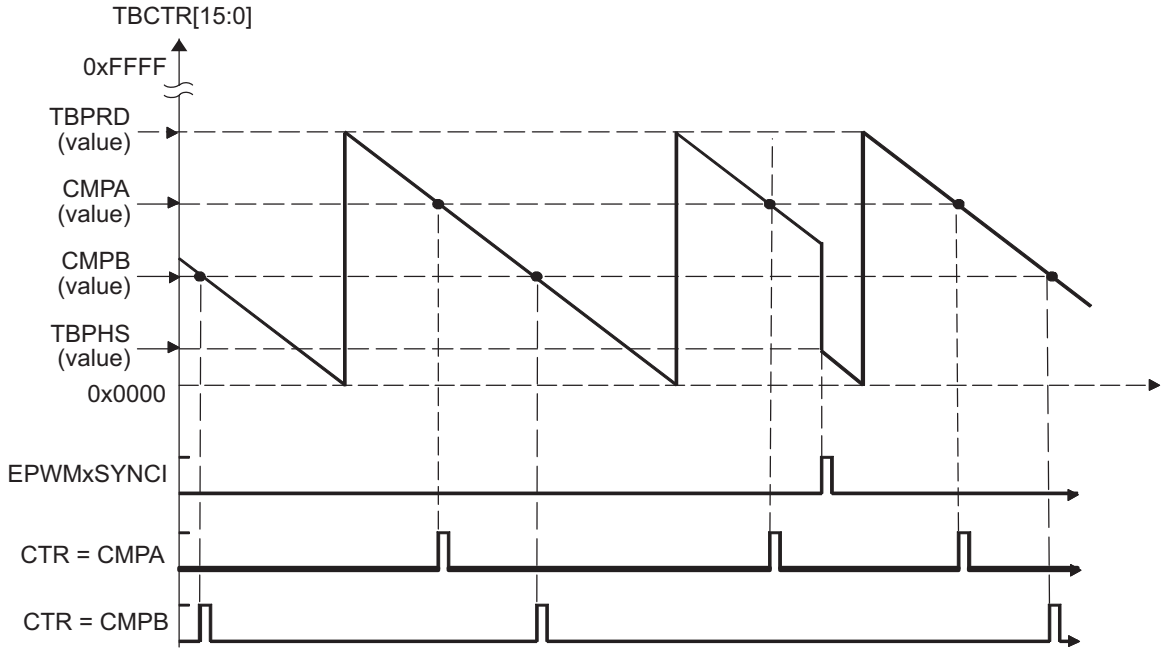


**Figure 18-17. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 0] Count Down On Synchronization Event**

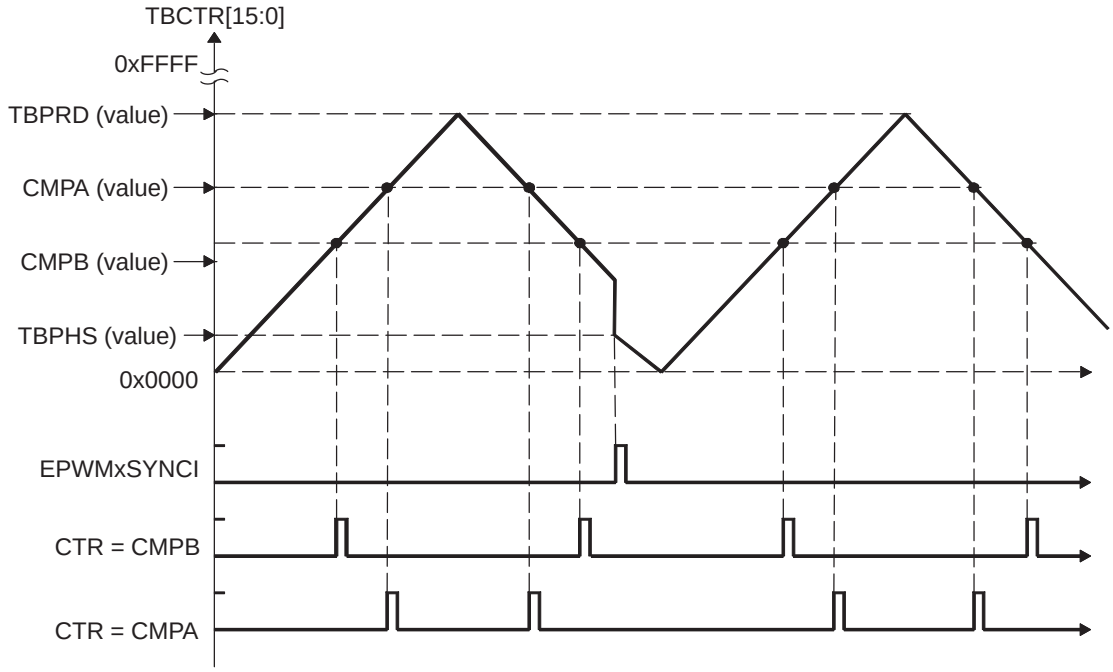
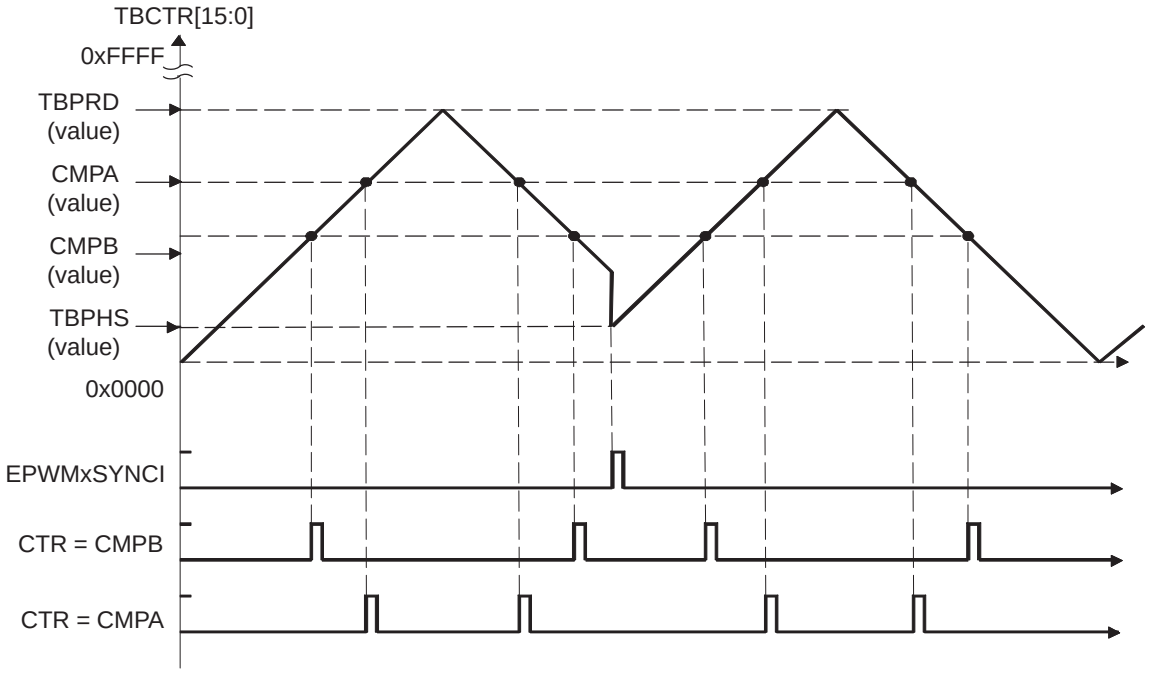


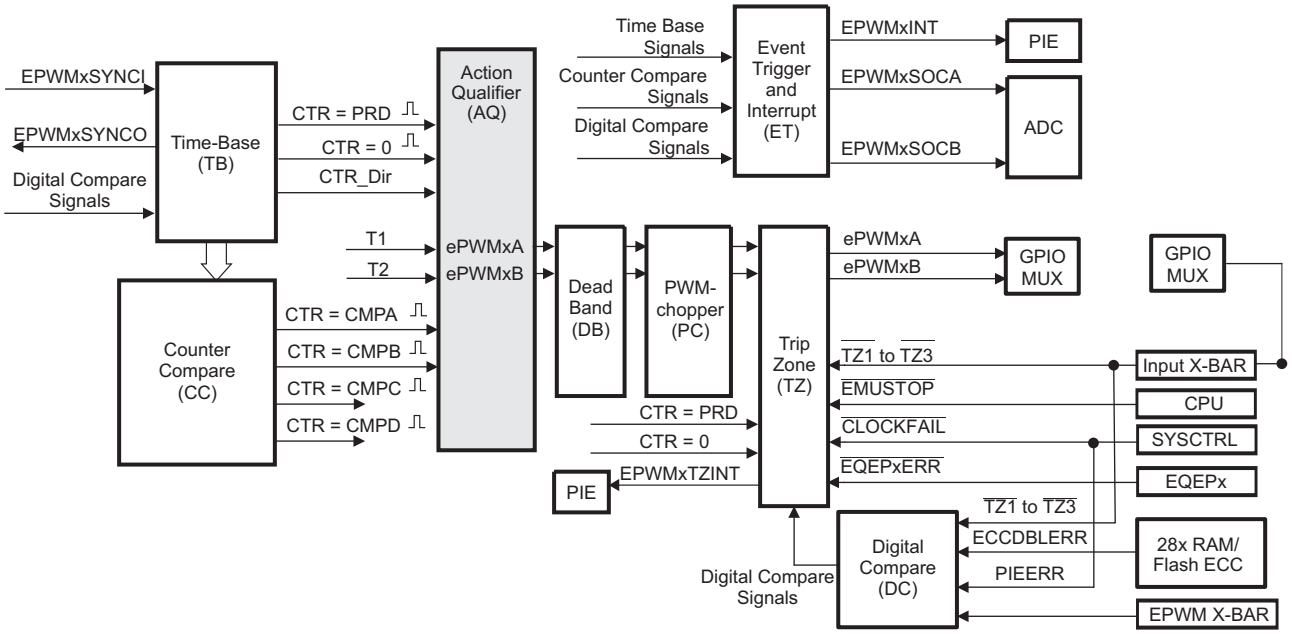

**Figure 18-18. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 1] Count Up On Synchronization Event**




### 18.3.4 Action-Qualifier (AQ) Submodule

Figure 18-19 shows the action-qualifier (AQ) submodule in the ePWM system.

**Figure 18-19. Action-Qualifier Submodule**




The action-qualifier submodule has the most important role in waveform construction and PWM generation. It decides which events are converted into various action types, thereby producing the required switched waveforms at the EPWMxA and EPWMxB outputs.

### 18.3.4.1 Purpose of the Action-Qualifier Submodule

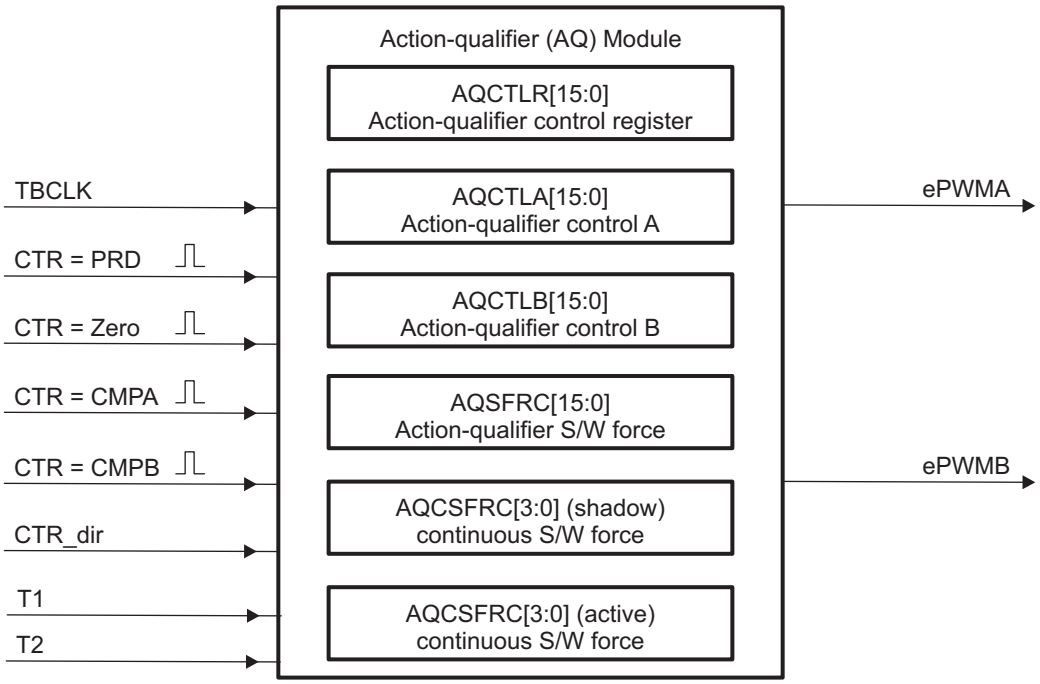
The action-qualifier submodule is responsible for the following:

- Qualifying and generating actions (set, clear, toggle) based on the following events:
  - CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD).
  - CTR = Zero: Time-base counter equal to zero (TBCTR = 0x00)
  - CTR = CMPA: Time-base counter equal to the counter-compare A register (TBCTR = CMPA)
  - CTR = CMPB: Time-base counter equal to the counter-compare B register (TBCTR = CMPB)
- T1, T2 events: Trigger events based on comparator, trip or syncin events
- Managing priority when these events occur concurrently
- Providing independent control of events when the time-base counter is increasing and when it is decreasing

### 18.3.4.2 Action-Qualifier Submodule Control and Status Register Definitions

The action-qualifier submodule operation is shown in Figure 18-20 rolled and monitored via the registers in Section 18.6.

**Figure 18-20. Action-Qualifier Submodule Inputs and Outputs**




For convenience, the possible input events are summarized again in Table 18-3.

**Table 18-3. Action-Qualifier Submodule Possible Input Events**

| Signal | Description | Registers Compared |
|---|---|---|
| CTR = PRD | Time-base counter equal to the period value | TBCTR = TBPRD |
| CTR = Zero | Time-base counter equal to zero | TBCTR = 0x00 |
| CTR = CMPA | Time-base counter equal to the counter-compare A | TBCTR = CMPA |
| CTR = CMPB | Time-base counter equal to the counter-compare B | TBCTR = CMPB |
| T1 event | Based on comparator, trip or syncin events | None |
| T2 event | Based on comparator, trip or syncin events | None |
| Software forced event | Asynchronous event initiated by software | |

The software forced action is a useful asynchronous event. This control is handled by the AQSFRC and AQCSFRC registers.

The action-qualifier submodule controls how the two outputs EPWMxA and EPWMxB behave when a particular event occurs. The event inputs to the action-qualifier submodule are further qualified by the counter direction (up or down). This allows for independent action on outputs on both the count-up and count-down phases.

The possible actions imposed on outputs EPWMxA and EPWMxB are:

- **Set High:**

  Set output EPWMxA or EPWMxB to a high level.
- **Clear Low:**

  Set output EPWMxA or EPWMxB to a low level.
- **Toggle:**

  If EPWMxA or EPWMxB is currently pulled high, then pull the output low. If EPWMxA or EPWMxB is currently pulled low, then pull the output high.
- **Do Nothing:**

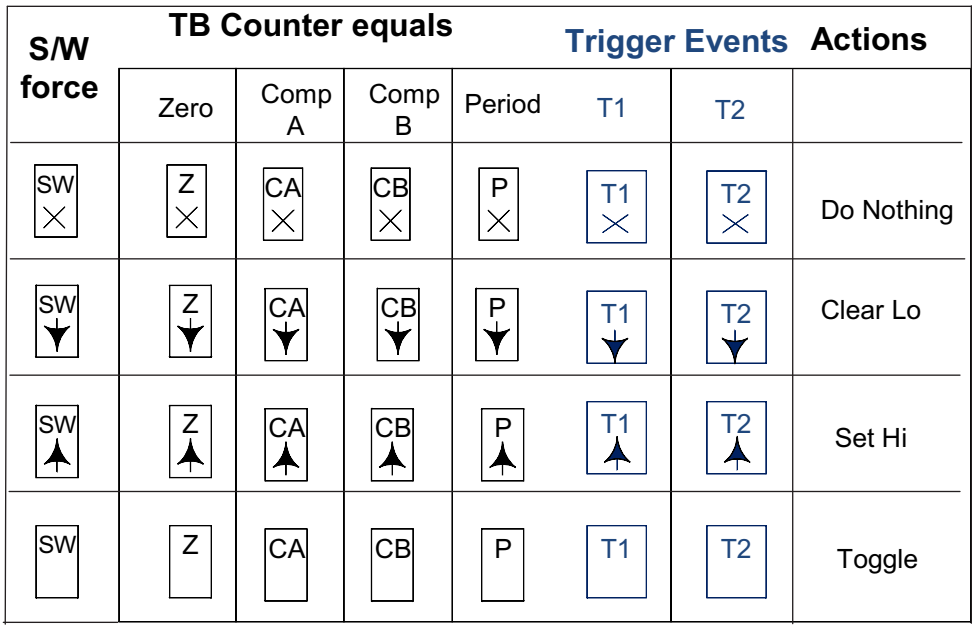
  Keep outputs EPWMxA and EPWMxB at same level as currently set. Although the "Do Nothing" option

prevents an event from causing an action on the EPWMxA and EPWMxB outputs, this event can still trigger interrupts and ADC start of conversion. See the Event-trigger Submodule description in Section 18.3.8 for details.

Actions are specified independently for either output (EPWMxA or EPWMxB). Any or all events can be configured to generate actions on a given output. For example, both CTR = CMPA and CTR = CMPB can operate on output EPWMxA. All qualifier actions are configured via the control registers found at the end of this section.

For clarity, the drawings in this document use a set of symbolic actions. These symbols are summarized in Figure 18-21. Each symbol represents an action as a marker in time. Some actions are fixed in time (zero and period) while the CMPA and CMPB actions are moveable and their time positions are programmed via the counter-compare A and B registers, respectively. To turn off or disable an action, use the "Do Nothing option"; it is the default at reset.

**Figure 18-21. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs**




The Action Qualifier Trigger Event Source Selection register (AQTSRCSEL) is used to select the source for T1 and T2 events. T1/T2 selection and configuration of a trip/digital-compare event in Action Qualifier submodule is independent of the configuration of that event in the Trip-Zone submodule. A particular trip event may or may not be configured to cause trip action in the Trip Zone submodule, but the same event can be used by the Action Qualifier to generate T1/T2 for controlling PWM generation.

### 18.3.4.3 Action-Qualifier Event Priority

It is possible for the ePWM action qualifier to receive more than one event at the same time. In this case events are assigned a priority by the hardware. The general rule is events occurring later in time have a higher priority and software forced events always have the highest priority. The event priority levels for up-down-count mode are shown in Table 18-4. 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 18-4. Action-Qualifier Event Priority for Up-Down-Count Mode**

| Priority Level | Event If TBCTR is Incrementing TBCTR = Zero up to TBCTR = TBPRD | Event If TBCTR is Decrementing TBCTR = TBPRD down to TBCTR = 1 |
|---|---|---|
| 1 (Highest) | Software forced event | Software forced event |
| 2 | T1 on up-count (T1U) | T1 on down-count (T1D) |
| 3 | T2 on up-count (T2U) | T2 on down-count (T2D) |
| 4 | Counter equals CMPB on up-count (CBU) | Counter equals CMPB on down-count (CBD) |
| 5 | Counter equals CMPA on up-count (CAU) | Counter equals CMPA on down-count (CAD) |
| 6 | Counter equals zero | Counter equals period (TBPRD) |
| 7 | T1 on down-count (T1D) | T1 on up-count (T1U) |
| 8 | T2 on down-count (T2D) | T2 on up-count (T2U) |
| 9 | Counter equals CMPB on down-count (CBD) | Counter equals CMPB on up-count (CBU) |
| 10 (Lowest) | Counter equals CMPA on down-count (CAD) | Counter equals CMPA on up-count (CBU) |

Table 18-5 shows the action-qualifier priority for up-count mode. In this case, the counter direction is always defined as up and thus down-count events will never be taken.

**Table 18-5. Action-Qualifier Event Priority for Up-Count Mode**

| Priority Level | Event |
|---|---|
| 1 (Highest) | Software forced event |
| 2 | Counter equal to period (TBPRD) |
| 3 | T1 on up-count (T1U) |
| 4 | T2 on up-count (T2U) |
| 5 | Counter equal to CMPB on up-count (CBU) |
| 6 | Counter equal to CMPA on up-count (CAU) |
| 7 (Lowest) | Counter equal to Zero |

Table 18-6 shows the action-qualifier priority for down-count mode. In this case, the counter direction is always defined as down and thus up-count events will never be taken.

**Table 18-6. Action-Qualifier Event Priority for Down-Count Mode**

| Priority Level | Event |
|---|---|
| 1 (Highest) | Software forced event |
| 2 | Counter equal to Zero |
| 3 | T1 on down-count (T1D) |
| 4 | T2 on down-count (T2D) |
| 3 | Counter equal to CMPB on down-count (CBD) |
| 4 | Counter equal to CMPA on down-count (CAD) |
| 5 (Lowest) | Counter equal to period (TBPRD) |

It is possible to set the compare value greater than the period. In this case the action will take place as shown in Table 18-7.

**Table 18-7. Behavior if CMPA/CMPB is Greater than the Period**

| Counter Mode | Compare on Up-Count Event CAD/CBD | Compare on Down-Count Event CAD/CBD |
|---|---|---|
| Up-Count Mode | If CMPA/CMPB ≤ TBPRD period, then the event occurs on a compare match (TBCTR=CMPA or CMPB). | Never occurs. |
| | If CMPA/CMPB > TBPRD, then the event will not occur. | |
| Down-Count Mode | Never occurs. | If CMPA/CMPB < TBPRD, the event will occur on a compare match (TBCTR=CMPA or CMPB). |
| | | If CMPA/CMPB ≥ TBPRD, the event will occur on a period match (TBCTR=TBPRD). |
| Up-Down-Count Mode | If CMPA/CMPB < TBPRD and the counter is incrementing, the event occurs on a compare match (TBCTR=CMPA or CMPB). | If CMPA/CMPB < TBPRD and the counter is decrementing, the event occurs on a compare match (TBCTR=CMPA or CMPB). |
| | If CMPA/CMPB is ≥ TBPRD, the event will occur on a period match (TBCTR = TBPRD). | If CMPA/CMPB ≥ TBPRD, the event occurs on a period match (TBCTR=TBPRD). |

#### 18.3.4.4 AQCTLA and AQCTLB Shadow Mode Operations

To enable Action Qualifier mode changes which must occur at the end of a period even when the phase changes, shadowing of the AQCTLA and AQCTLB registers has been added on ePWMs type 2 and later. Additionally, shadow to active load on SYNC of these registers is supported as well. Shadowing of this register is enabled and disabled by the AQCTLR[SHDWAQAMODE] and AQCTLR[SHDWAQBMODE] bits. These bits enable and disable the AQCTLA shadow register and AQCTLB shadow register, respectively. The behavior of the two load modes is described below:

**Shadow Mode:**

The shadow mode for the AQCTLA is enabled by setting the AQCTLR[SHDWAQAMODE] bit, and the shadow register for AQCTLB is enabled by setting the AQCTLR[SHDWAQBMODE] bit. Shadow mode is disabled by default for both AQCTLA and AQCTLB

If the shadow register is enabled, then the content of the shadow register is transferred to the active register on one of the following events as specified by the AQCTLR[LDAQAMODE] AQCTLR[LDAQBMODE] AQCTLR[LDAQASYNC] & AQCTLR[LDAQBSYNC] register bits:

- CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD).
- CTR = Zero: Time-base counter equal to zero (TBCTR = 0x00)
- Both CTR = PRD and CTR = Zero
- SYNC event caused by DCAEVT1 or DCBEVT1 or EPWMxSYNCI or TBCTL[SWFSYNC]
- Both SYNC event or a selection made by LDAQAMODE/LDAQBMODE

**Global Reload Support**

Global reload control mechanism can also be used for AQCTLA:AQCTLA2, AQCTLB:AQCTLB2 and AQCSFRC registers by configuring the appropriate bits in the global load configuration register (GLDCFG). When global reload mode is selected, the transfer of contents from shadow register to active register for all registers that have this mode enabled, occurs at the same event as defined by the configuration bits in the Global Shadow to Active Load Control Register (GLDCTL). The global reload control mechanism is explained in Section 18.3.2.7.

**Immediate Load Mode:**

If immediate load mode is selected (that is, AQCTLR[SHDWAQAMODE] = 0 or AQCTLR[SHDWAQBMODE] = 0), then a read from or a write to the register will go directly to the active register. See Figure 18-22 and Figure 18-23.

**NOTE:** Shadow to Active Load of Action Qualifier Output A/B Control Register [AQCTLA & AQCTLB] on CMPA = 0 or CMPB = 0 boundary

If the Counter-Compare A Register (CMPA) or Counter-Compare B Register (CMPB) is set to a value of 0 and the action qualifier action on AQCTLA and AQCTLB is configured to occur in the same instant as a shadow to active load (that is, CMPA=0 and AQCTLA shadow to active load on TBCTR=0 using AQCTLR register LDAQAMODE and LDAQAMODE bits), then both events enter contention and it is recommended to use a Non-Zero Counter-Compare when using Shadow to Active Load of Action Qualifier Output A/B Control Register on TBCTR = 0 boundary.

**Figure 18-22. AQCTLR[SHDWAQAMODE]**

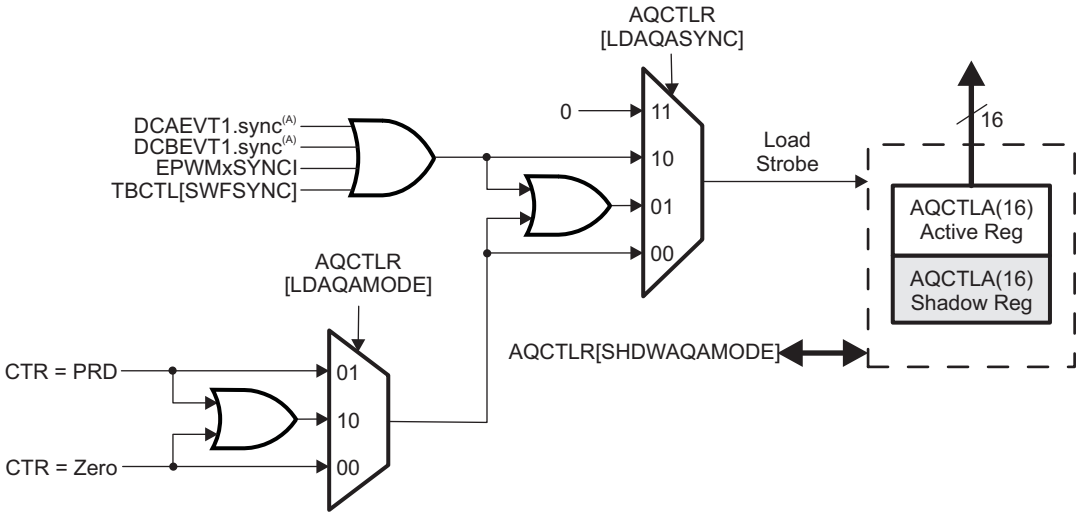
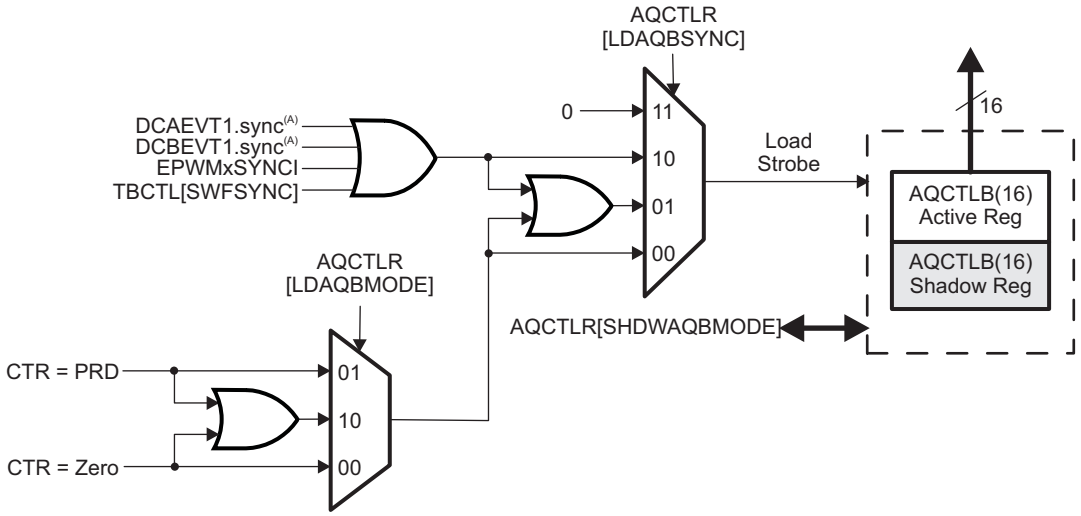



**Figure 18-23. AQCTLR[SHDWAQBMODE]**



### 18.3.4.5 Waveforms for Common Configurations

**NOTE:** The waveforms in this document show the behavior of the ePWMs for a static compare register value. In a running system, the active compare registers (CMPA and CMPB) are typically updated from their respective shadow registers once every period. The user specifies when the update will take place; either when the time-base counter reaches zero or when the time-base counter reaches period. There are some cases when the action based on the new value can be delayed by one period or the action based on the old value can take effect for an extra period. Some PWM configurations avoid this situation. These include, but are not limited to, the following:

**Use up-down-count mode to generate a symmetric PWM:**

- If you load CMPA/CMPB on zero, then use CMPA/CMPB values greater than or equal to 1.
- If you load CMPA/CMPB on period, then use CMPA/CMPB values less than or equal to TBPRD-1.

  This means there will always be a pulse of at least one TBCLK cycle in a PWM period which, when very short, tend to be ignored by the system.

**Use up-down-count mode to generate an asymmetric PWM:**

- To achieve 50%-0% asymmetric PWM use the following configuration: Load CMPA/CMPB on period and use the period action to clear the PWM and a compare-up action to set the PWM. Modulate the compare value from 0 to TBPRD to achieve 50%-0% PWM duty.

**When using up-count mode to generate an asymmetric PWM:**

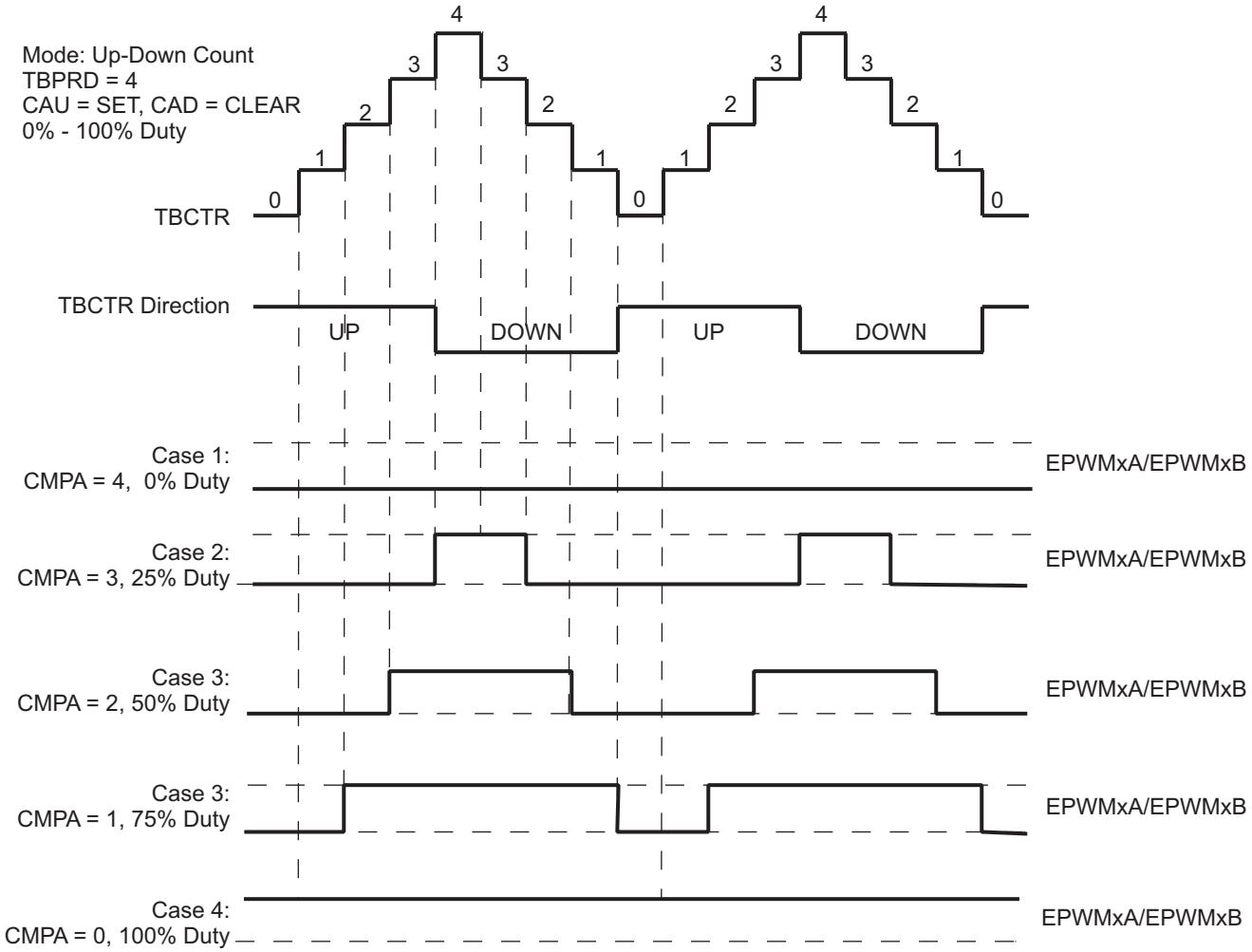
- To achieve 0-100% asymmetric PWM use the following configuration: Load CMPA/CMPB on TBPRD. Use the Zero action to set the PWM and a compare-up action to clear the PWM. Modulate the compare value from 0 to TBPRD+1 to achieve 0-100% PWM duty.

See the *Using Enhanced Pulse Width Modulator (ePWM) Module for 0-100% Duty Cycle Control* Application Report (literature number SPRAAI1)

Figure 18-24 shows how a symmetric PWM waveform can be generated using the up-down-count mode of the TBCTR. In this mode 0%-100% DC modulation is achieved by using equal compare matches on the up count and down count portions of the waveform. In the example shown, CMPA is used to make the comparison. When the counter is incrementing the CMPA match will pull the PWM output high. Likewise, when the counter is decrementing the compare match will pull the PWM signal low. When CMPA = 0, the PWM signal is low for the entire period giving the 0% duty waveform. When CMPA = TBPRD, the PWM signal is high achieving 100% duty.

When using this configuration in practice, if you load CMPA/CMPB on zero, then use CMPA/CMPB values greater than or equal to 1. If you load CMPA/CMPB on period, then use CMPA/CMPB values less than or equal to TBPRD-1. This means there will always be a pulse of at least one TBCLK cycle in a PWM period which, when very short, tend to be ignored by the system.

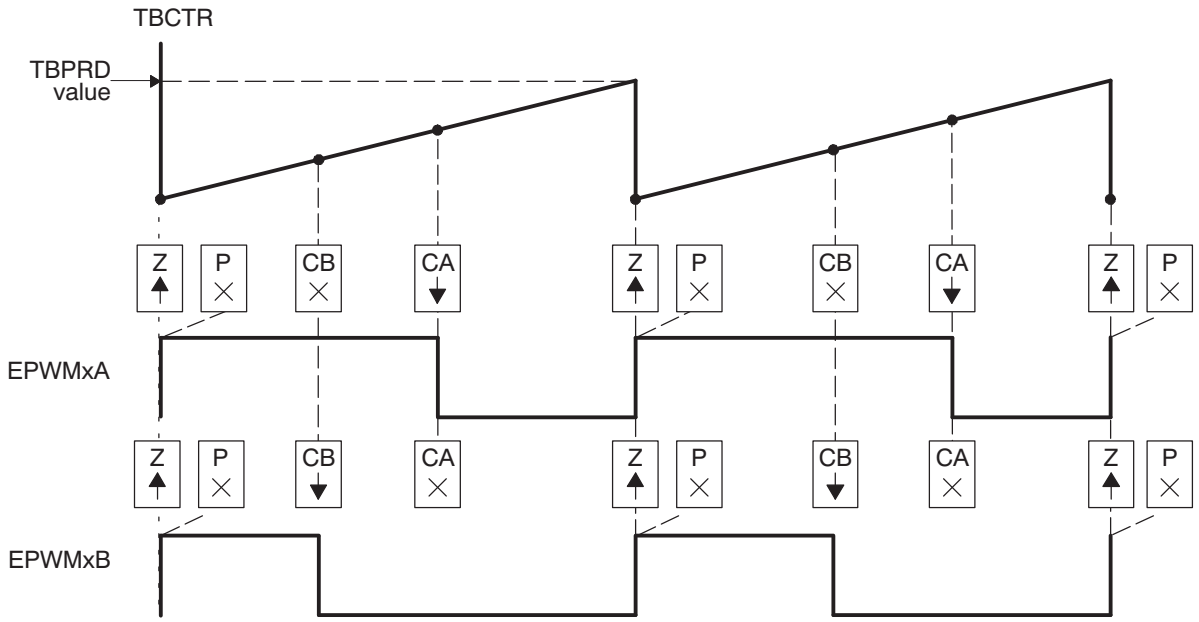
**Figure 18-24. Up-Down-Count Mode Symmetrical Waveform**




The PWM waveforms in Figure 18-25 through Figure 18-30 show some common action-qualifier configurations. Some conventions used in the figures and examples are as follows:

- TBPRD, CMPA, and CMPB refer to the value written in their respective registers. The active register, not the shadow register, is used by the hardware.
- CMPx, refers to either CMPA or CMPB.
- EPWMxA and EPWMxB refer to the output signals from ePWMx
- Up-Down means Count-up-and-down mode, Up means up-count mode and Dwn means down-count mode
- Sym = Symmetric, Asym = Asymmetric

**Figure 18-25. Up, Single Edge Asymmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB—Active High**




A PWM period = (TBPRD + 1 ) × $T_{TBCLK}$

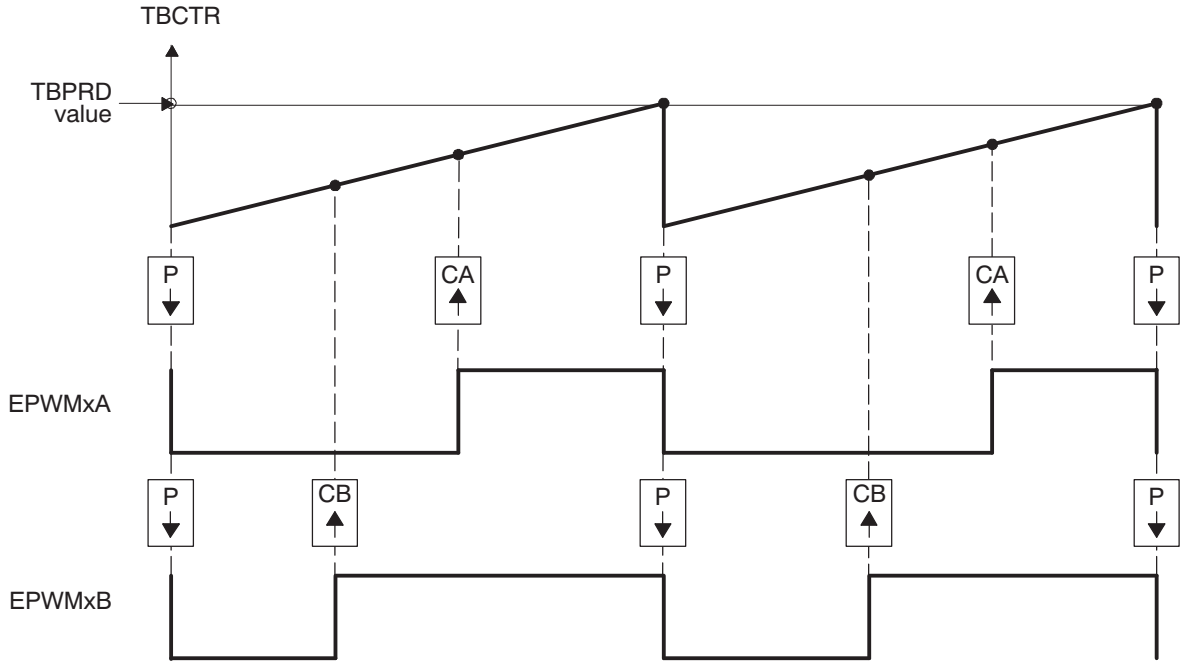
B Duty modulation for EPWMxA is set by CMPA, and is active high (that is, high time duty proportional to CMPA).

C Duty modulation for EPWMxB is set by CMPB and is active high (that is, high time duty proportional to CMPB).

D The "Do Nothing" actions ( X ) are shown for completeness, but will not be shown on subsequent diagrams.

E Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCTR wraps from period to 0000.

**Figure 18-26. Up, Single Edge Asymmetric Waveform With Independent Modulation on EPWMxA and EPWMxB—Active Low**



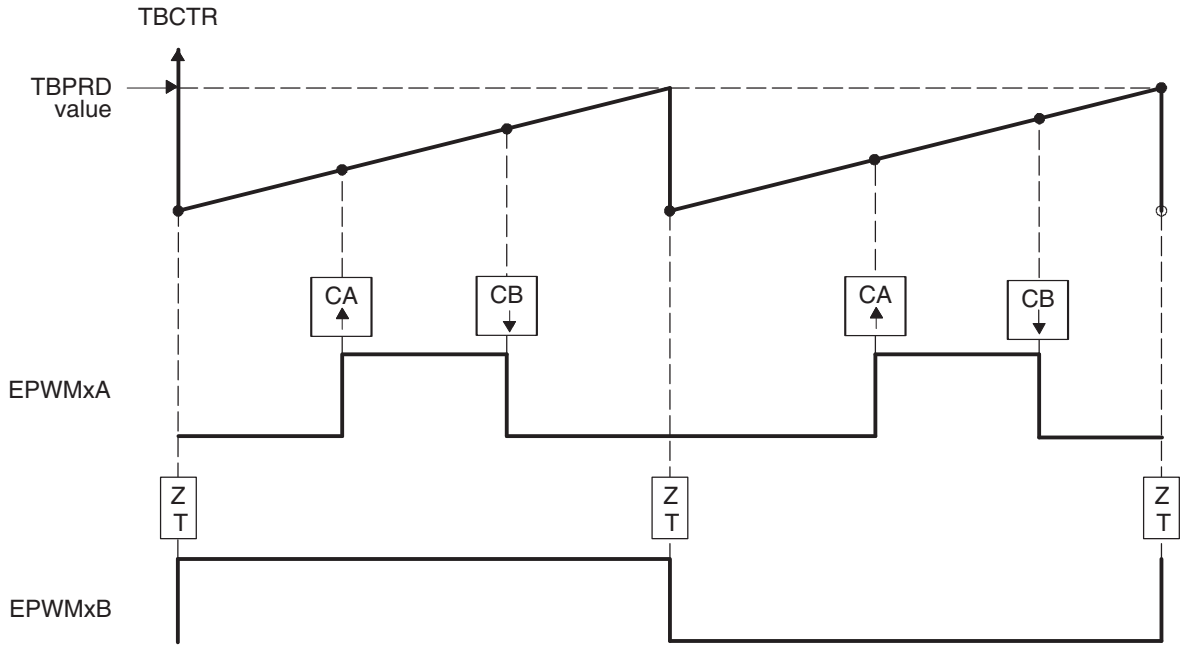

A PWM period = (TBPRD + 1 ) × $T_{TBCLK}$

B Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).

C Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).

D Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCTR wraps from period to 0000.

**Figure 18-27. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA**

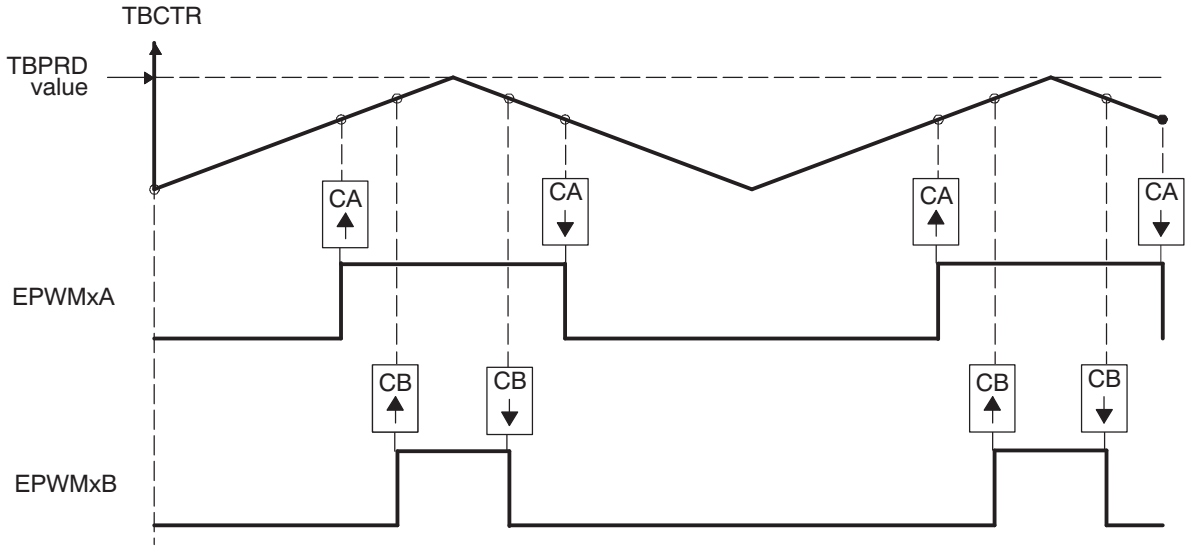



A PWM frequency = 1/( (TBPRD + 1 ) × $T_{TBCLK}$ )

B Pulse can be placed anywhere within the PWM cycle (0000 - TBPRD)

C High time duty proportional to (CMPB - CMPA)

**Figure 18-28. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active Low**



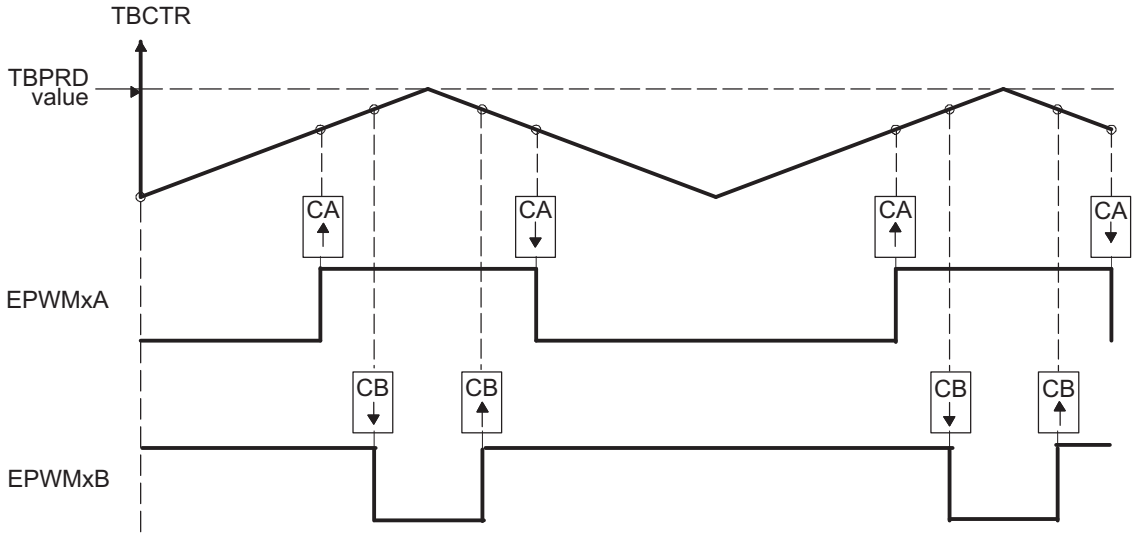

A PWM period = 2 x TBPRD × $T_{TBCLK}$

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 18-29. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Complementary**




A PWM period = 2 × TBPRD × $T_{TBCLK}$

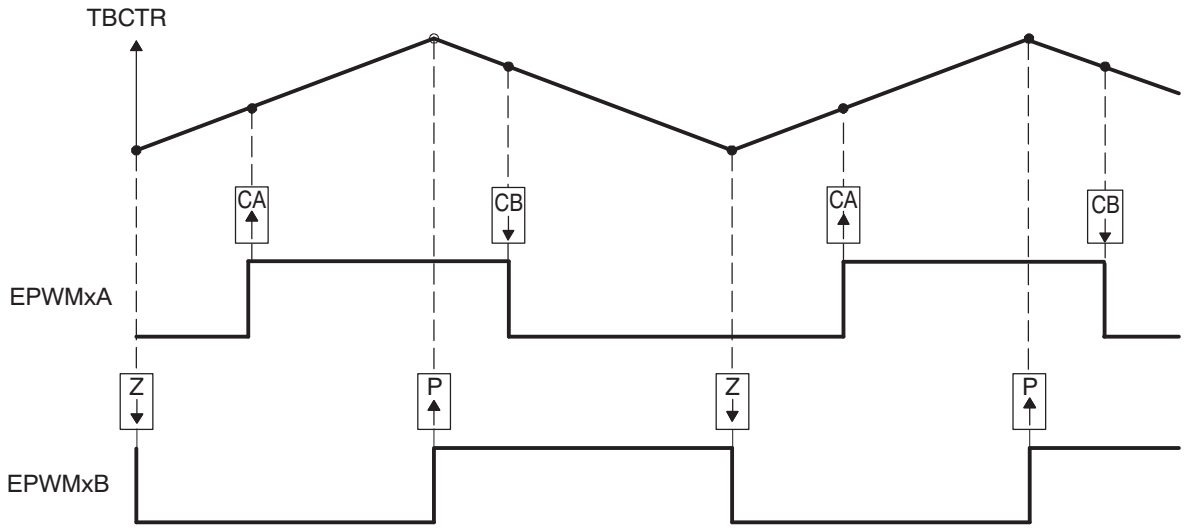
B Duty modulation for EPWMxA is set by CMPA, and is active low, that is, low time duty proportional to CMPA.

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 18-30. Up-Down-Count, Dual Edge Asymmetric Waveform, With Independent Modulation on EPWMxA—Active Low**




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.

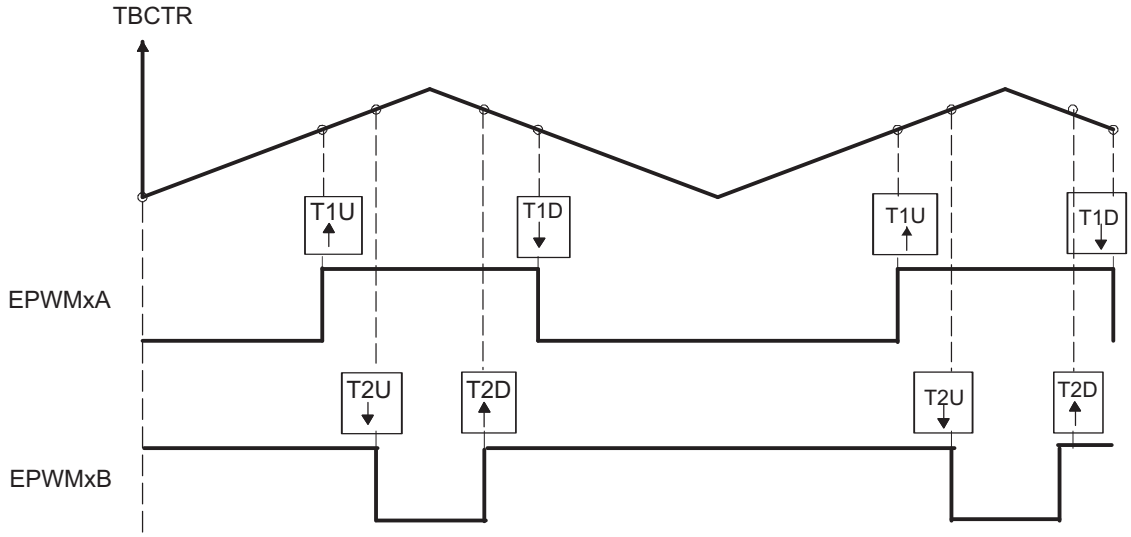
C Duty modulation for EPWMxA is set by CMPA and CMPB.

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 18-31. Up-Down-Count, PWM Waveform Generation Utilizing T1 and T2 Events**




A PWM period = 2 × TBPRD × TTBCLK

B Independent T1 event actions when counter is counting up and when it is counting down are used to generate EPWMxA output.

C Independent T2 event actions when counter is counting up and when it is counting down are used to generate EPWMxB output.

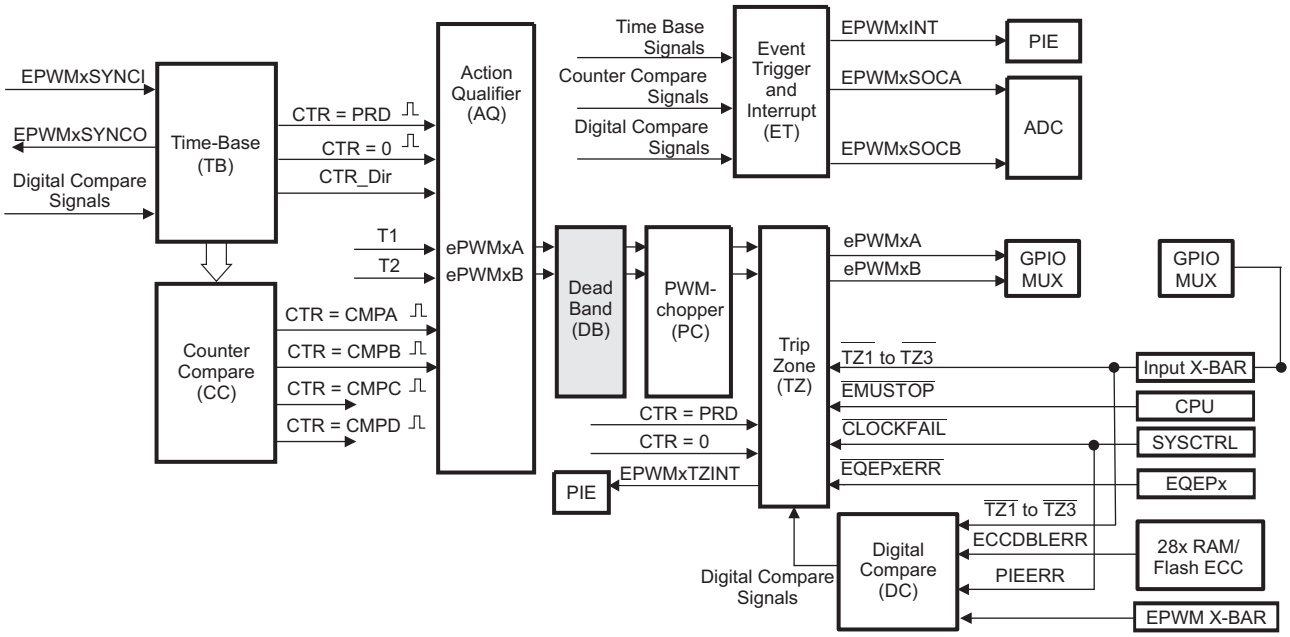
D TZ1 is selected as the source for T1.

E TZ2 is selected as the source for T2.

### 18.3.5 Dead-Band Generator (DB) Submodule

Figure 18-32 illustrates the dead-band submodule within the ePWM module.

**Figure 18-32. Dead_Band Submodule**




#### 18.3.5.1 Purpose of the Dead-Band Submodule

The action-qualifier (AQ) module section discussed how it is possible to generate the required dead band by having full control over edge placement using both the CMPA and CMPB resources of the ePWM module. However, if the more classical edge delay-based dead band with polarity control is required, then the dead-band submodule described here should be used.

The key functions of the dead-band module are:

- Generating appropriate signal pairs (EPWMxA and EPWMxB) with dead-band relationship from a single EPWMxA input
- Programming signal pairs for:
  - Active high (AH)
  - Active low (AL)
  - Active high complementary (AHC)
  - Active low complementary (ALC)
- Adding programmable delay to rising edges (RED)
- Adding programmable delay to falling edges (FED)
- Can be totally bypassed from the signal path (note dotted lines in diagram)

#### 18.3.5.2 Dead-band Submodule Additional Operating Modes

On type 1 ePWM RED could appear on one channel output and FED could appear on the other channel output.

The following list shows the distinct difference between type 1 and type 4 modules with respect to dead-band operating modes:

- By adding S6, S7, and S8 in Figure 18-33, RED and FED can appear on both the A-channel and B-channel outputs. Additionally, both RED and FED together can be applied to either the A-channel or B-channel outputs to allow B-channel phase shifting with respect to the A-channel.
  **Note:** Phase shifting B-channel with respect to the A-channel using the dead-band submodule

additional operating modes has limitations with respect to the choice of RED and FED delay with respect to the operating duty cycle of the ePWMxA and ePWMxB outputs.

- The dead-band counters have also been increased to 14 bits
- Dead-band and dead-band High-resolution registers are now shadowed.
- High-resolution dead-band RED and FED have been enabled using the DBREDHR and DBFEDHR registers

**NOTE:** The PWM chopper will not be enabled when high-resolution dead band is enabled.

**NOTE:** High-resolution dead-band RED and FED requires Half-Cycle clocking mode (DBCTL[HALFCYCLE] = 1).

Cannot have both RED and FED together applied to both ePWMxA and ePWMxB. RED and FED together can be applied only to either OutA OR OutB.

**NOTE:** Phase shifting B-channel with respect to the A-channel: When PWMxB is derived from PWMxA using the DEDB_MODE bit and by delaying rising edge and falling edge by the phase shift amount. When the duty cycle value on PWMxA is less than this phase shift amount, PWMxA's falling edge has precedence over the delayed rising edge for PWMxB. It is recommended to make sure the duty cycle value of the current waveform fed to the dead-band module is greater than the required phase shift amount.

**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] & DBCTL [LOADFEDMODE] register bits:

- CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD).
- CTR = Zero: Time-base counter equal to zero (TBCTR = 0x00)
- Both CTR = PRD and CTR = Zero

The DBCTL register can be shadowed. The shadow mode for DBCTL is enabled by setting the DBCTL2[SHDWDBCTLMODE] bit. If the shadow register is enabled then the content of the shadow register is transferred to the active register on one of the following events as specified by the DBCTL2[LOADDBCTLMODE] register bit:

- CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD)
- CTR = Zero: Time-base counter equal to zero (TBCTR = 0x00)
- Both CTR = PRD and CTR = Zero

**Global Reload Support**

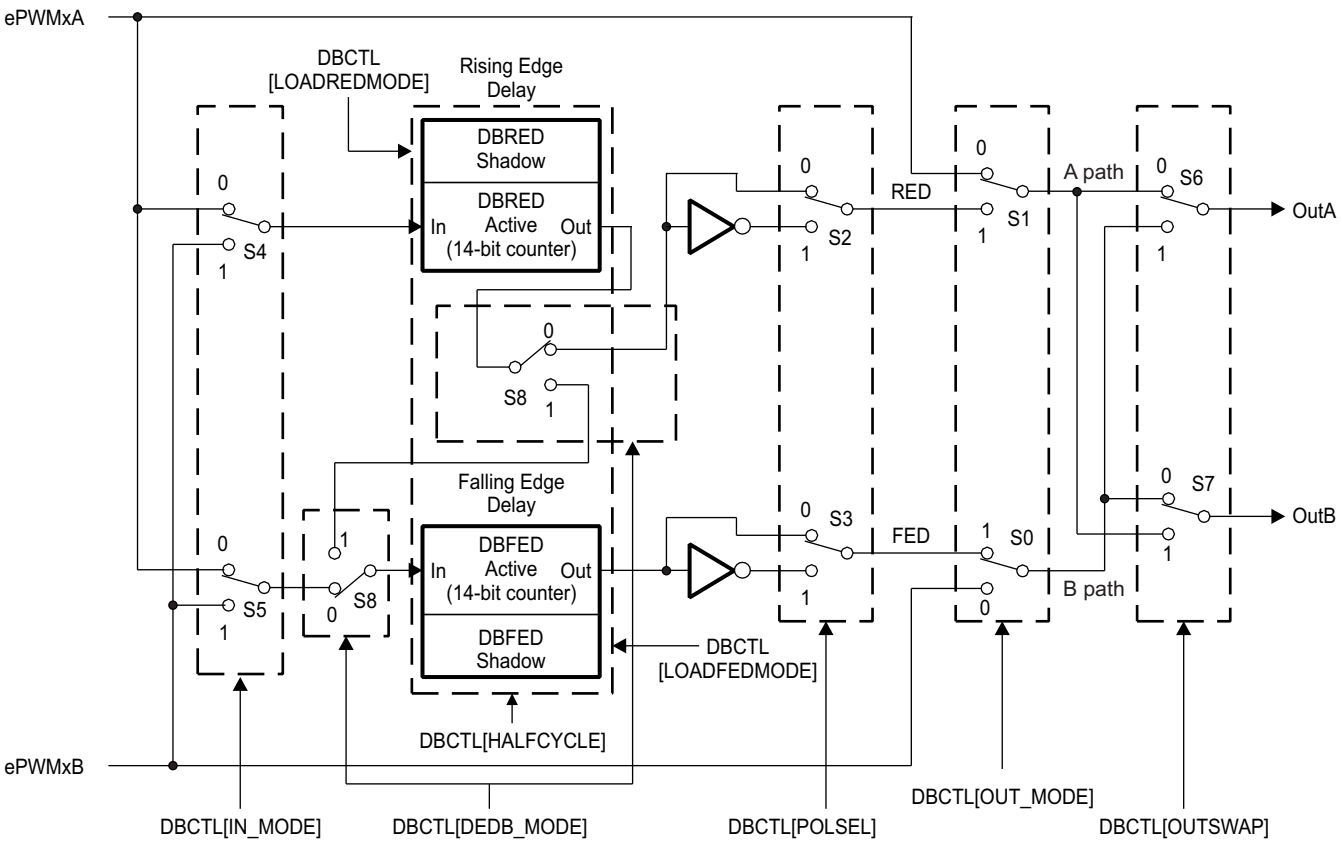
Global reload control mechanism can also be used for DBRED:DBREDHR, DBFED:DBFEDHR and DBCTL registers by configuring the appropriate bits in the global load configuration register (GLDCFG). When global reload mode is selected the transfer of contents from shadow register to active register, for all registers that have this mode enabled, occurs at the same event as defined by the configuration bits in the Global Shadow to Active Load Control Register (GLDCTL). The Global reload control mechanism is explained in Section 18.3.2.7.

**NOTE:** When DBRED/DBFED active is loaded with a new shadow value while DB counters are counting, the new DBRED / DBFED value only affects the NEXT PWMx edge and not the current edge.

### 18.3.5.3 Operational Highlights for the Dead-Band Submodule

The configuration options for the dead-band submodule are shown in Figure 18-33.

**Figure 18-33. Configuration Options for the Dead-Band Submodule**




Although all combinations are supported, not all are typical usage modes. Table 18-8 documents some classical dead-band configurations. These modes assume that the DBCTL[IN_MODE] is configured such that EPWMxA In is the source for both falling-edge and rising-edge delay. Enhanced, or non-traditional modes can be achieved by changing the input signal source. The modes shown in Table 18-8 fall into the following categories:

- **Mode 1: Bypass both falling-edge delay (FED) and rising-edge delay (RED)**

  Allows you to fully disable the dead-band submodule from the PWM signal path.
- **Mode 2-5: Classical Dead-Band Polarity Settings:**

  These represent typical polarity configurations that should address all the active high/low modes required by available industry power switch gate drivers. The waveforms for these typical cases are shown in Figure 18-34. Note that to generate equivalent waveforms to Figure 18-34, configure the action-qualifier submodule to generate the signal as shown for EPWMxA.
- **Mode 6: Bypass rising-edge-delay and Mode 7: Bypass falling-edge-delay**

  Finally the last two entries in Table 18-8 show combinations where either the falling-edge-delay (FED) or rising-edge-delay (RED) blocks are bypassed.

**Table 18-8. Classical Dead-Band Operating Modes**

| Mode | Mode Description | DBCTL[POLSEL] | | DBCTL[OUT_MODE] | |
|---|---|---|---|---|---|
| | | S3 | S2 | S1 | S0 |
| 1 | EPWMxA and EPWMxB Passed Through (No Delay) | X | X | 0 | 0 |
| 2 | Active High Complementary (AHC) | 1 | 0 | 1 | 1 |
| 3 | Active Low Complementary (ALC) | 0 | 1 | 1 | 1 |

**Table 18-8. Classical Dead-Band Operating Modes (continued)**

| Mode | Mode Description | DBCTL[POLSEL] | | DBCTL[OUT_MODE] | |
|---|---|---|---|---|---|
| | | S3 | S2 | S1 | S0 |
| 4 | Active High (AH) | 0 | 0 | 1 | 1 |
| 5 | Active Low (AL) | 1 | 1 | 1 | 1 |
| 6 | EPWMxA Out = EPWMxA In (No Delay)<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 |

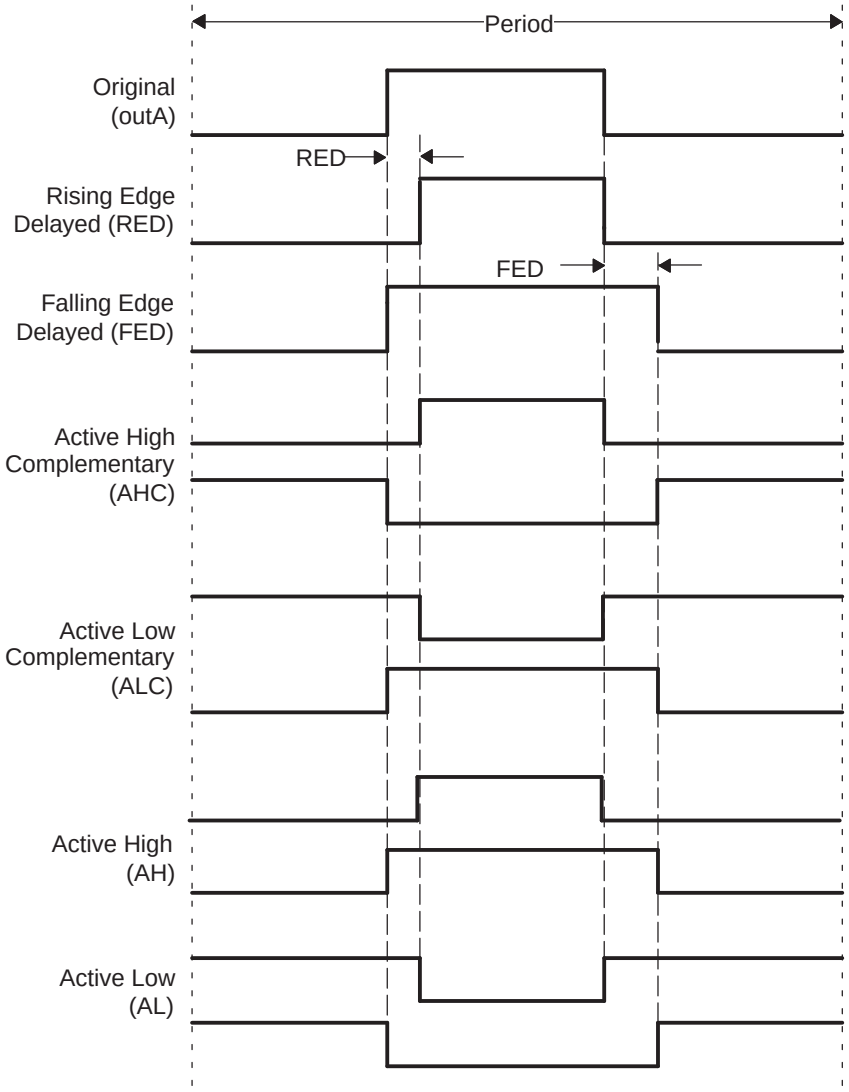
**Table 18-9. Additional Dead-Band Operating Modes**

| Mode Description | DBCTL[DEDB-MODE] | DBCTL[OUTSWAP] | |
|---|---|---|---|
| | S8 | S6 | S7 |
| EPWMxA and EPWMxB signals are as defined by OUT-MODE bits. | 0 | 0 | 0 |
| EPWMxA = A-path as defined by OUT-MODE bits.<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, user should always either set OUT_MODE bits such that Apath = InA ***or*** OUTSWAP bits such that EPWMxA=Bpath. Otherwise, EPWMxA will be invalid.

Figure 18-34 shows waveforms for typical cases where 0% < duty < 100%.

**Figure 18-34. 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 their value represents the number of time-base clock, TBCLK, periods by which a signal edge is delayed. For example, the formula to calculate falling-edge-delay and rising-edge-delay is:

$FED = DBFED \times T_{TBCLK}$

$RED = DBRED \times T_{TBCLK}$

Where $T_{TBCLK}$ is the period of TBCLK, the prescaled version of EPWMCLK.

For convenience, delay values for various TBCLK options are shown in Table 18-10. The ePWM input clock frequency that these delay values been computed by is 100 MHz.

**Table 18-10. Dead-Band Delay Values in µS as a Function of DBFED and DBRED**

| Dead-Band Value | Dead-Band Delay in µS | | |
|---|---|---|---|
| DBFED, DBRED | TBCLK = EPWMCLK/1 | TBCLK = EPWMCLK /2 | TBCLK = EPWMCLK/4 |
| 1 | 0.01 µS | 0.02 µS | 0.04 µS |
| 5 | 0.05 µS | 0.10µS | 0.20 µS |
| 10 | 0.10 µS | 0.20 µS | 0.40 µS |
| 100 | 1.00 µS | 2.00 µS | 4.00 µS |
| 200 | 2.00 µS | 4.00 µS | 8.00 µS |
| 400 | 4.00 µS | 8.00 µS | 16.00 µS |
| 500 | 5.00 µS | 10.00 µS | 20.00 µS |
| 600 | 6.00 µS | 12.00 µS | 24.00 µS |
| 700 | 7.00 µS | 14.00 µS | 28.00 µS |
| 800 | 8.00 µS | 16.00 µS | 32.00 µS |
| 900 | 9.00 µS | 18.00 µS | 36.00 µS |
| 1000 | 10.00 µS | 20.00 µS | 40.00 µS |

When half-cycle clocking is enabled, the formula to calculate the falling-edge-delay and rising-edge-delay becomes:

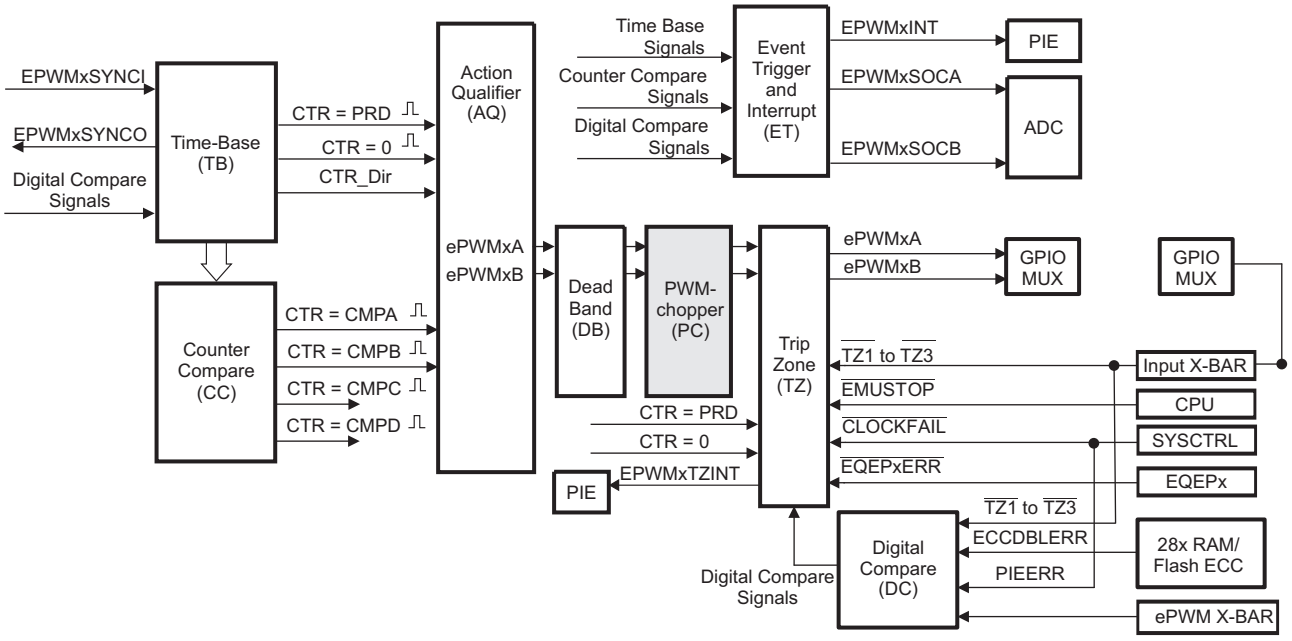
$FED = DBFED \times T_{TBCLK}/2$

$RED = DBRED \times T_{TBCLK}/2$

### 18.3.6 PWM Chopper (PC) Submodule

Figure 18-35 illustrates the PWM chopper (PC) submodule within the ePWM module.

**Figure 18-35. PWM Chopper Submodule**




The PWM chopper submodule allows a high-frequency carrier signal to modulate the PWM waveform generated by the action-qualifier and dead-band submodules. This capability is important if you need pulse transformer-based gate drivers to control the power switching elements.

#### 18.3.6.1 Purpose of the PWM Chopper Submodule

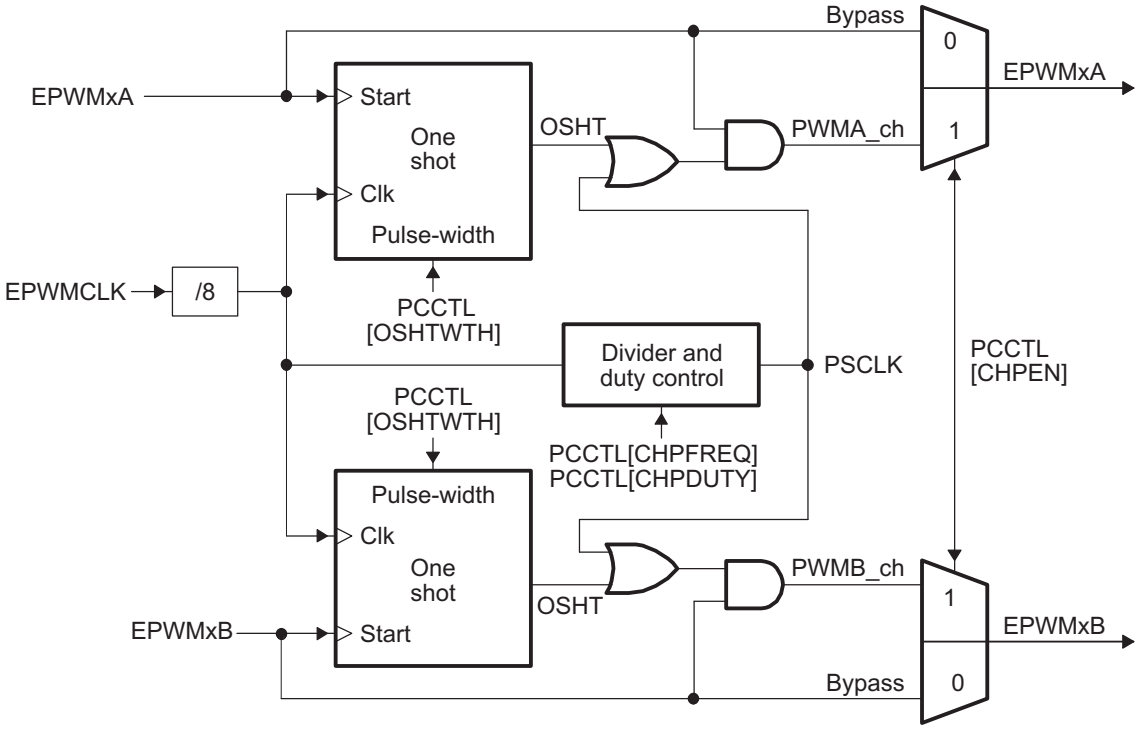
The key functions of the PWM chopper submodule are:

- Programmable chopping (carrier) frequency
- Programmable pulse width of first pulse
- Programmable duty cycle of second and subsequent pulses
- Can be fully bypassed if not required

#### 18.3.6.2 Operational Highlights for the PWM Chopper Submodule

Figure 18-36 shows the operational details of the PWM chopper submodule. The carrier clock is derived from EPWMCLK. Its frequency and duty cycle are controlled via the CHPFREQ and CHPDUTY bits in the PCCTL register. The one-shot block is a feature that provides a high energy first pulse to ensure hard and fast power switch turn on, while the subsequent pulses sustain pulses, ensuring the power switch remains on. The one-shot width is programmed via the OSHTWTH bits. The PWM chopper submodule can be fully disabled (bypassed) via the CHPEN bit.

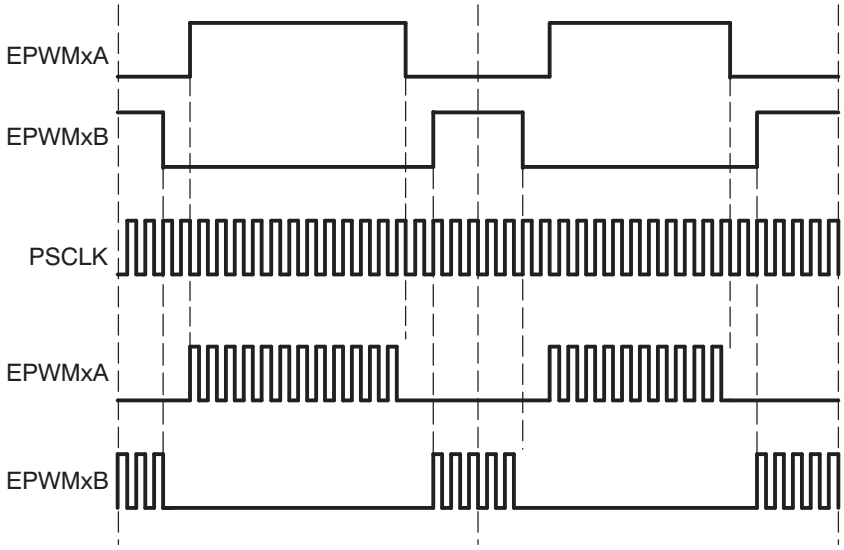
**Figure 18-36. PWM Chopper Submodule Operational Details**




### 18.3.6.3 Waveforms

Figure 18-37 shows simplified waveforms of the chopping action only; one-shot and duty-cycle control are not shown. Details of the one-shot and duty-cycle control are discussed in the following sections.

**Figure 18-37. Simple PWM Chopper Submodule Waveforms Showing Chopping Action Only**



#### 18.3.6.3.1 One-Shot Pulse

The width of the first pulse can be programmed to any of 16 possible pulse width values. The width or period of the first pulse is given by:

$T_{1stpulse} = T_{EPWMCLK} \times 8 \times OSHTWTH$

Where $T_{EPWMCLK}$ is the period of the system clock (EPWMCLK) and OSHTWTH is the four control bits (value from 1 to 16)

Figure 18-38 shows the first and subsequent sustaining pulses and Table 18-11 gives the possible pulse width values for a EPWMCLK = 80 MHz.

**Figure 18-38. PWM Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses**

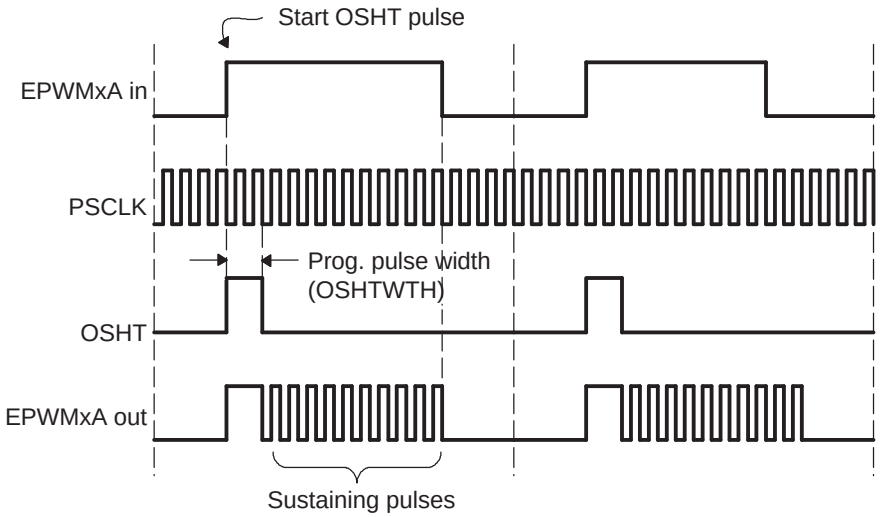



**Table 18-11. Possible Pulse Width Values for EPWMCLK = 80 MHz**

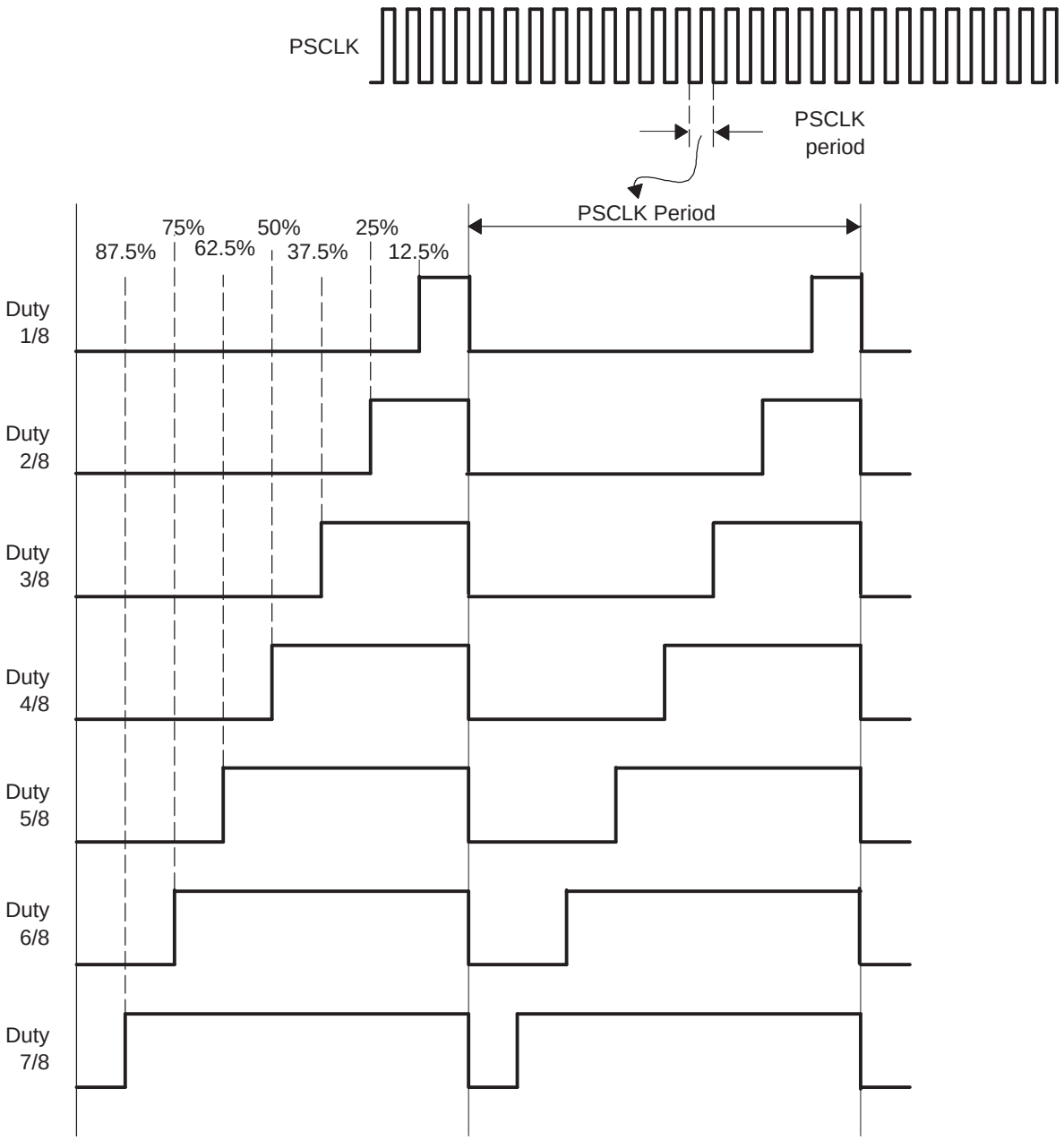
| OSHTWTHz (hex) | Pulse Width (nS) |
|---|---|
| 0 | 100 |
| 1 | 200 |
| 2 | 300 |
| 3 | 400 |
| 4 | 500 |
| 5 | 600 |
| 6 | 700 |
| 7 | 800 |
| 8 | 900 |
| 9 | 1000 |
| A | 1100 |
| B | 1200 |
| C | 1300 |
| D | 1400 |
| E | 1500 |
| F | 1600 |

#### 18.3.6.3.2 Duty Cycle Control

Pulse transformer-based gate drive designs need to comprehend the magnetic properties or characteristics of the transformer and associated circuitry. Saturation is one such consideration. To assist the gate drive designer, the duty cycles of the second and subsequent pulses have been made programmable. These sustaining pulses ensure the correct drive strength and polarity is maintained on the power switch gate during the on period, and hence a programmable duty cycle allows a design to be tuned or optimized via software control.

Figure 18-39 shows the duty cycle control that is possible by programming the CHPDUTY bits. One of seven possible duty ratios can be selected ranging from 12.5% to 87.5%.

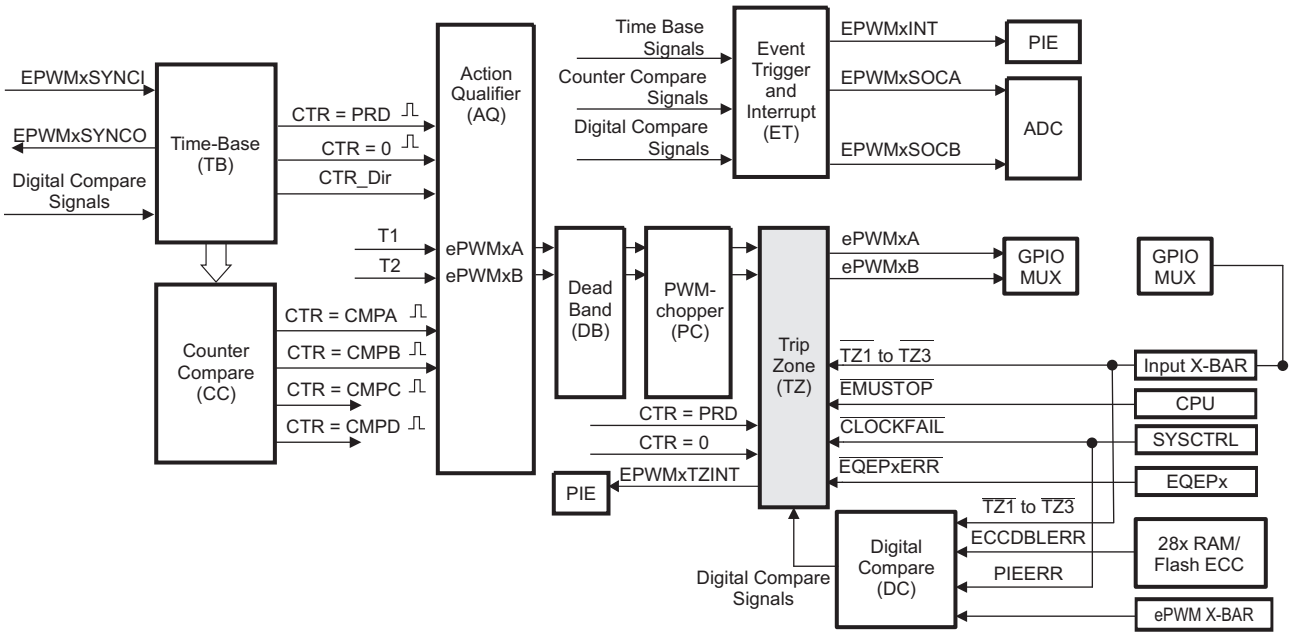
**Figure 18-39. PWM Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses**



### 18.3.7 Trip-Zone (TZ) Submodule

Figure 18-40 shows how the trip-zone (TZ) submodule fits within the ePWM module.

**Figure 18-40. Trip-Zone Submodule**




Each ePWM module is connected to six $\overline{\text{TZn}}$ signals ($\overline{\text{TZ1}}$ to $\overline{\text{TZ6}}$). $\overline{\text{TZ1}}$ to $\overline{\text{TZ3}}$ are sourced from the GPIO mux. $\overline{\text{TZ4}}$ is sourced from an inverted EQEPxERR signal on those devices with an EQEP module. $\overline{\text{TZ5}}$ is connected to the system clock fail logic, and $\overline{\text{TZ6}}$ is sourced from the EMUSTOP output from the CPU. These signals indicate external fault or trip conditions, and the ePWM outputs can be programmed to respond accordingly when faults occur.

#### 18.3.7.1 Purpose of the Trip-Zone Submodule

The key functions of the trip-zone submodule are:

- Trip inputs $\overline{\text{TZ1}}$ to $\overline{\text{TZ6}}$ can be flexibly mapped to any ePWM module.
- Upon a fault condition, outputs EPWMxA and EPWMxB can be forced to one of the following:
  - High
  - Low
  - High-impedance
  - No action taken
- Support for one-shot trip (OSHT) for major short circuits or over-current conditions.
- Support for cycle-by-cycle tripping (CBC) for current limiting operation.
- Support for digital compare tripping (DC) based on state of on-chip analog comparator module outputs and/or $\overline{\text{TZ1}}$ to $\overline{\text{TZ3}}$ signals.
- Each trip-zone input and digital compare (DC) submodule DCAEVT1/2 or DCBEVT1/2 force event can be allocated to either one-shot or cycle-by-cycle operation.
- Interrupt generation is possible on any trip-zone input.
- Software-forced tripping is also supported.
- The trip-zone submodule can be fully bypassed if it is not required.

#### 18.3.7.2 Operational Highlights for the Trip-Zone Submodule

The following sections describe the operational highlights and configuration options for the trip-zone submodule.

The trip-zone signals $\overline{TZ1}$ to $\overline{TZ6}$ (also collectively referred to as $\overline{TZn}$) are active low input signals. When one of these signals goes low, or when a DCAEVT1/2 or DCBEVT1/2 force happens based on the TZDCSEL register event selection, it indicates that a trip event has occurred. Each ePWM module can be individually configured to ignore or use each of the trip-zone signals or DC events. Which trip-zone signals or DC events are used by a particular ePWM module is determined by the TZSEL register for that specific ePWM module. The trip-zone signals may or may not be synchronized to the EPWM clock (EPWMCLK) and digitally filtered within the GPIO MUX block. A minimum of 3*TBCLK low pulse width on $\overline{TZn}$ inputs is sufficient to trigger a fault condition on the ePWM module. If the pulse width is less than this, the trip condition may not be latched by CBC or OST latches. The asynchronous trip makes sure that if clocks are missing for any reason, the outputs can still be tripped by a valid event present on $\overline{TZn}$ inputs . The GPIOs or peripherals must be appropriately configured. For more information, see the device-specific version of the *System Control and Interrupts* chapter.

Each $\overline{TZn}$ input can be individually configured to provide either a cycle-by-cycle or one-shot trip event for an ePWM module. DCAEVT1 and DCBEVT1 events can be configured to directly trip an ePWM module or provide a one-shot trip event to the module. Likewise, DCAEVT2 and DCBEVT2 events can also be configured to directly trip an ePWM module or provide a cycle-by-cycle trip event to the module. This configuration is determined by the TZSEL[DCAEVT1/2], TZSEL[DCBEVT1/2], TZSEL[CBCn], and TZSEL[OSHTn] control bits (where n corresponds to the trip input) respectively.

- **Cycle-by-Cycle (CBC):**

  When a cycle-by-cycle trip event occurs, the action specified in the TZCTL[TZA] and TZCTL[TZB] bits is carried out immediately on the EPWMxA and/or EPWMxB outputs. Table 18-12 lists some of the possible actions. Independent actions can be specified based on the occurrence of the event while the counter is counting up and/or while it is counting down by appropriately configuring bits in the TZCTL2 register. Actions specified in the TZCTL2 register take effect only when the ETZE bit in TZCTL2 is set.

  Additionally, when a cycle-by-cycle trip event occurs, the cycle-by-cycle trip event flag (TZFLG[CBC]) is set and a EPWMx_TZINT interrupt is generated if it is enabled in the TZEINT register and PIE peripheral. A corresponding flag for the event that caused the CBC event is also set in register TZCBCFLG.

  If the CBC interrupt is enabled via the TZEINT register, and DCAEVT2 or DCBEVT2 are selected as CBC trip sources via the TZSEL register, it is not necessary to also enable the DCAEVT2 or DCBEVT2 interrupts in the TZEINT register, as the DC events trigger interrupts through the CBC mechanism.

  The specified condition on the inputs is automatically cleared based on the selection made with TZCLR[CBCPULSE] if the trip event is no longer present. Therefore, in this mode, the trip event is cleared or reset every PWM cycle. The TZFLG[CBC] and TZCBCFLG flag bits will remain set until they are manually cleared by writing to the TZCLR[CBC] and TZCBCCLR flag bits. If the cycle-by-cycle trip event is still present when the TZFLG[CBC] and/or TZCBCFLG register bits are cleared, then these bits will again be immediately set..

- **One-Shot (OSHT):**

  When a one-shot trip event occurs, the action specified in the TZCTL[TZA] and TZCTL[TZB] bits is carried out immediately on the EPWMxA and/or EPWMxB output. Table 18-12 lists some of the possible actions. Independent actions can be specified based on the occurrence of the event while the counter is counting up and/or while it is counting down by appropriately configuring bits in TZCTL2 register. Actions specified in TZCTL2 register take effect only when ETZE bit in TZCTL2 is set.

  Additionally, when a one-shot trip event occurs, the one-shot trip event flag (TZFLG[OST]) is set and a EPWMx_TZINT interrupt is generated if it is enabled in the TZEINT register and PIE peripheral. A corresponding flag for the event that caused the OST event is also set in register TZOSTFLG. The one-shot trip condition must be cleared manually by writing to the TZCLR[OST] bit. If desired, TZOSTFLG register bit should be cleared by manually writing to the corresponding bit in the TZOSTCLR register.

  If the one-shot interrupt is enabled via the TZEINT register, and DCAEVT1 or DCBEVT1 are selected as OSHT trip sources via the TZSEL register, it is not necessary to also enable the DCAEVT1 or DCBEVT1 interrupts in the TZEINT register, as the DC events trigger interrupts through the OSHT mechanism.

- **Digital Compare Events (DCAEVT1/2 and DCBEVT1/2):**

  A digital compare DCAEVT1/2 or DCBEVT1/2 event is generated based on a combination of the

DCAH/DCAL and DCBH/DCBL signals as selected by the TZDCSEL register. The signals which source the DCAH/DCAL and DCBH/DCBL signals are selected via the DCTRIPSEL register and can be either trip zone input pins or analog comparator CMPSSx signals. For more information on the digital compare submodule signals, see Section 18.3.9.

When a digital compare event occurs, the action specified in the TZCTL[DCAEVT1/2] and TZCTL[DCBEVT1/2] bits is carried out immediately on the EPWMxA and/or EPWMxB output. Table 18-12 lists the possible actions. Independent actions can be specified based on the occurrence of the event while the counter is counting up and/or while it is counting down by appropriately configuring bits in TZCTLDCA and TZCTLDCB register. Actions specified in TZCTLDCA and TZCTLDCB registers take effect only when ETZE bit in TZCTL2 is set.

In addition, the relevant DC trip event flag (TZFLG[DCAEVT1/2] / TZFLG[DCBEVT1/2]) is set and a EPWMx_TZINT interrupt is generated if it is enabled in the TZEINT register and PIE peripheral.

The specified condition on the pins is automatically cleared when the DC trip event is no longer present. The TZFLG[DCAEVT1/2] or TZFLG[DCBEVT1/2] flag bit will remain set until it is manually cleared by writing to the TZCLR[DCAEVT1/2] or TZCLR[DCBEVT1/2] bit. If the DC trip event is still present when the TZFLG[DCAEVT1/2] or TZFLG[DCBEVT1/2] flag is cleared, then it will again be immediately set.

The action taken when a trip event occurs can be configured individually for each of the ePWM output pins by way of the TZCTL, TZCTL2, TZCTLDCA,, and TZCTLDCB register bit fields. Some of the possible actions, shown in Table 18-12, can be taken on a trip event.

**Table 18-12. Possible Actions On a Trip Event**

| TZCTL Register bit-field Settings | EPWMxA and/or EPWMxB | Comment |
|---|---|---|
| 0,0 | High-Impedance | Tripped |
| 0,1 | Force to High State | Tripped |
| 1,0 | Force to Low State | Tripped |
| 1,1 | No Change | Do Nothing.<br>No change is made to the output. |

***Example 18-1. Trip-Zone Configurations***

**Scenario A:**
A one-shot trip event on $\overline{TZ1}$ pulls both EPWM1A, EPWM1B low and also forces EPWM2A and EPWM2B high.

- Configure the ePWM1 registers as follows:
  - TZSEL[OSHT1] = 1: enables $\overline{TZ1}$ as a one-shot event source for ePWM1
  - TZCTL[TZA] = 2: EPWM1A will be forced low on a trip event.
  - TZCTL[TZB] = 2: EPWM1B will be forced low on a trip event.
- Configure the ePWM2 registers as follows:
  - TZSEL[OSHT1] = 1: enables $\overline{TZ1}$ as a one-shot event source for ePWM2
  - TZCTL[TZA] = 1: EPWM2A will be forced high on a trip event.
  - TZCTL[TZB] = 1: EPWM2B will be forced high on a trip event.

**Scenario B:**
A cycle-by-cycle event on $\overline{TZ5}$ pulls both EPWM1A, EPWM1B low.
A one-shot event on $\overline{TZ1}$ or $\overline{TZ6}$ puts EPWM2A into a high impedance state.

- Configure the ePWM1 registers as follows:
  - TZSEL[CBC5] = 1: enables $\overline{TZ5}$ as a one-shot event source for ePWM1
  - TZCTL[TZA] = 2: EPWM1A will be forced low on a trip event.
  - TZCTL[TZB] = 2: EPWM1B will be forced low on a trip event.

***Example 18-1. Trip-Zone Configurations (continued)***

- Configure the ePWM2 registers as follows:
  - TZSEL[OSHT1] = 1: enables $\overline{TZ1}$ as a one-shot event source for ePWM2
  - TZSEL[OSHT6] = 1: enables $\overline{TZ6}$ as a one-shot event source for ePWM2
  - TZCTL[TZA] = 0: EPWM2A will be put into a high-impedance state on a trip event.
  - TZCTL[TZB] = 3: EPWM2B will ignore the trip event.

#### 18.3.7.3 Generating Trip Event Interrupts

Figure 18-41 and Figure 18-42 illustrate the trip-zone submodule control and interrupt logic, respectively. DCAEVT1/2 and DCBEVT1/2 signals are described in further detail in Section 18.3.9.

**Figure 18-41. Trip-Zone Submodule Mode Control Logic**

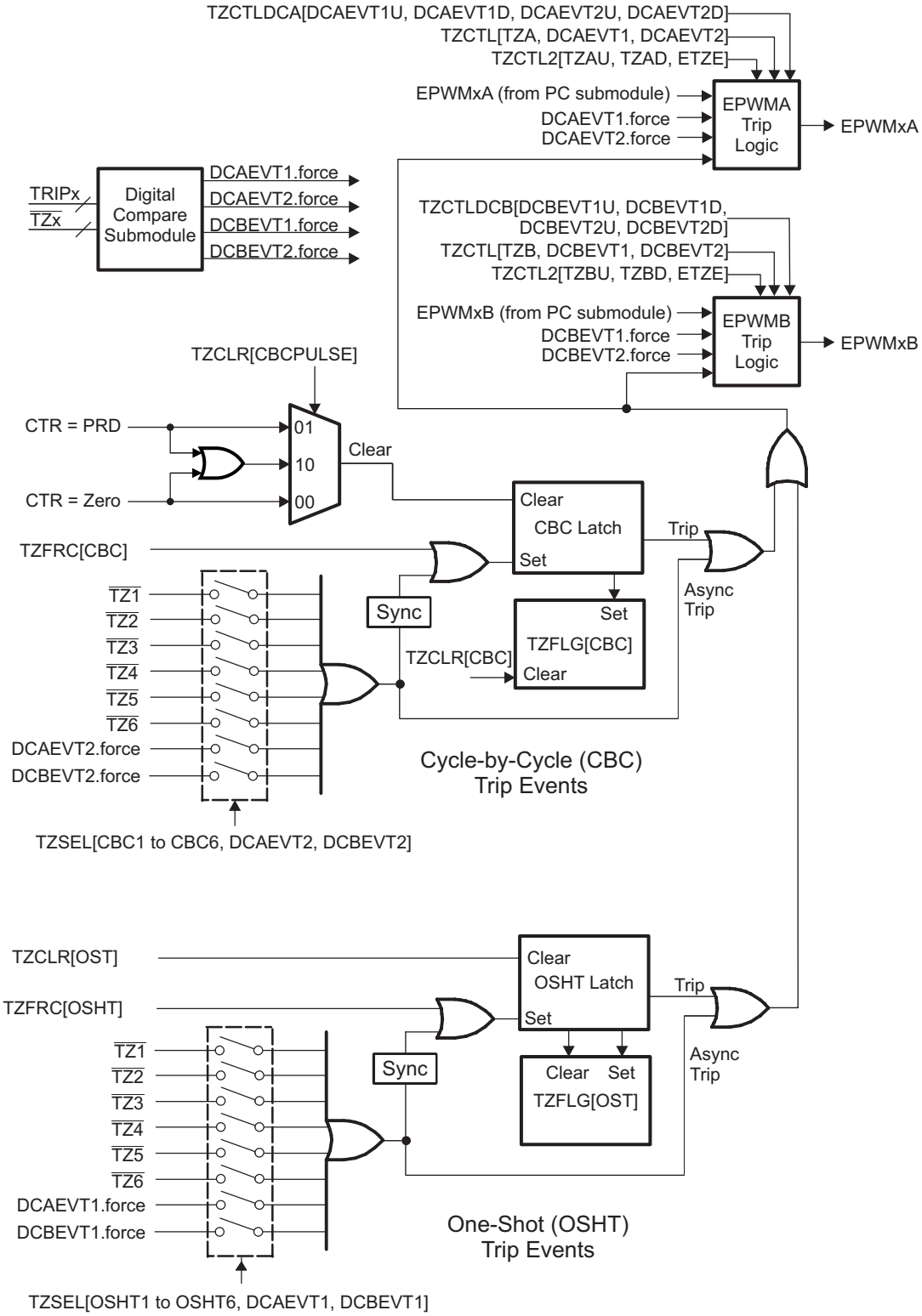
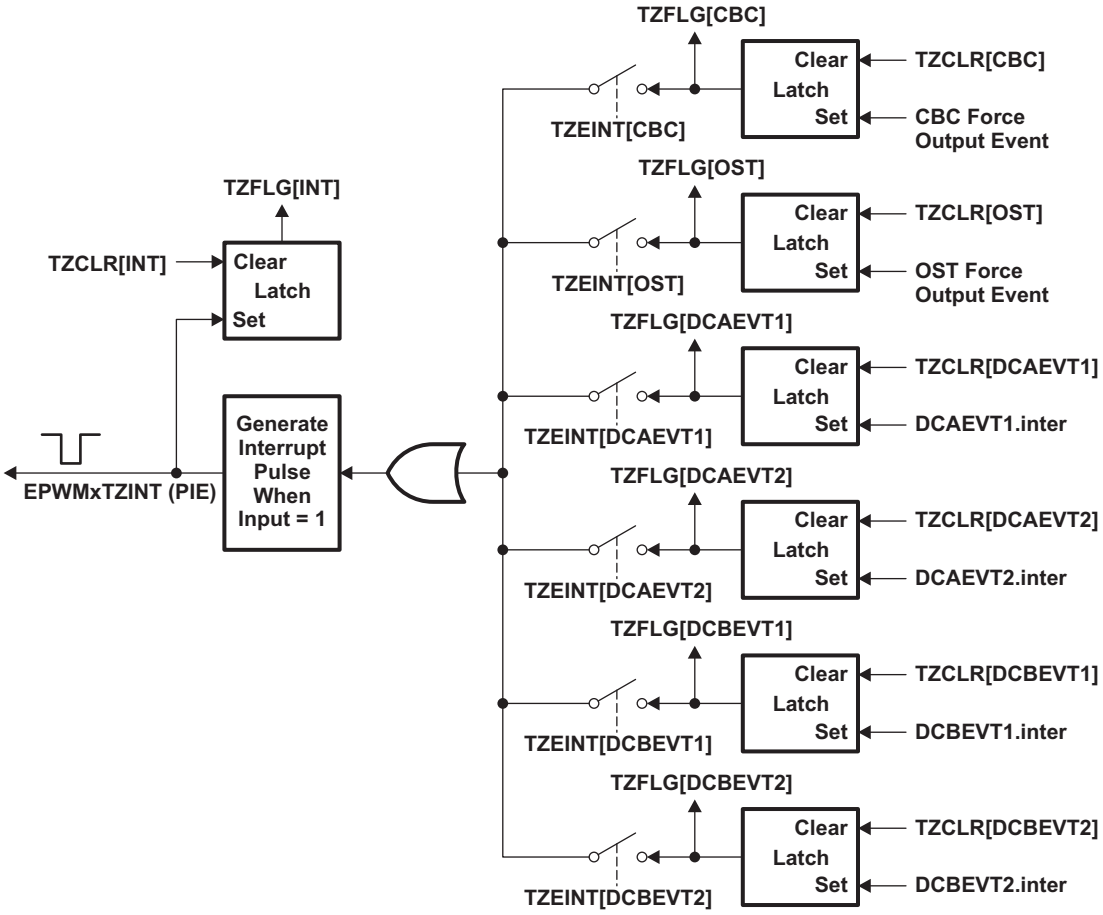


**Figure 18-42. Trip-Zone Submodule Interrupt Logic**



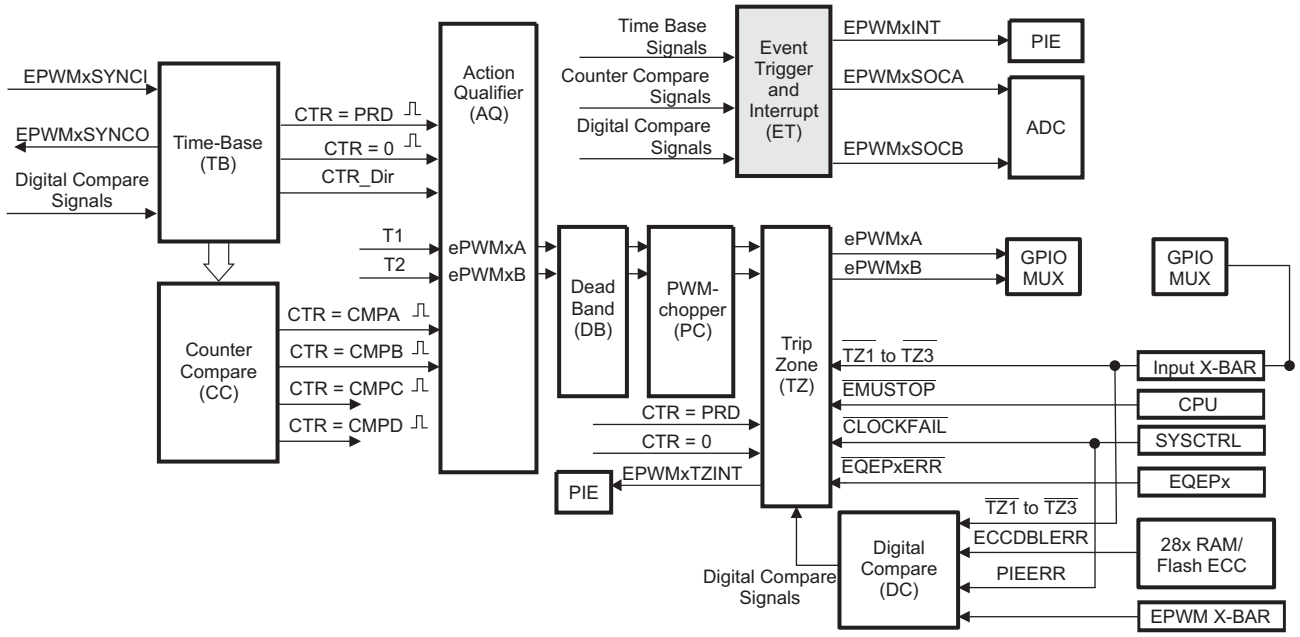

### 18.3.8 Event-Trigger (ET) Submodule

The key functions of the event-trigger submodule are:

- Receives event inputs generated by the time-base, counter-compare, and digital-compare submodules
- Uses the time-base direction information for up/down event qualification
- Uses prescaling logic to issue interrupt requests and ADC start of conversion at:
  - Every event
  - Every second event
  - Up to every fifteenth event
- Provides full visibility of event generation via event counters and flags
- Allows software forcing of Interrupts and ADC start of conversion

The event-trigger submodule manages the events generated by the time-base submodule, the counter-compare submodule, and the digital-compare submodule to generate an interrupt to the CPU and/or a start of conversion pulse to the ADC when a selected event occurs. Figure 18-43 illustrates where the event-trigger submodule fits within the ePWM system.

**Figure 18-43. Event-Trigger Submodule**

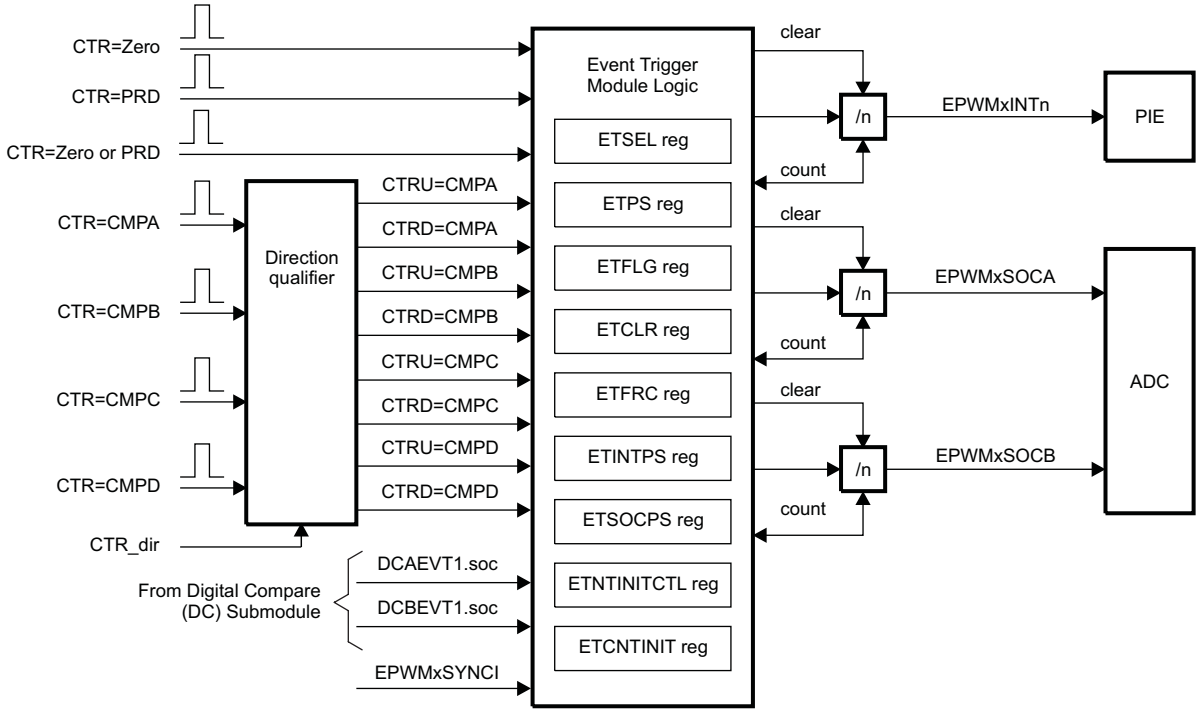



### 18.3.8.1 Operational Overview of the ePWM Type 4 Event-Trigger Submodule

The event-trigger submodule monitors various event conditions (shown as inputs on the left side of Figure 18-44) and can be configured to prescale these events before issuing an Interrupt request or an ADC start of conversion. The event-trigger prescaling logic can issue Interrupt requests and ADC start of conversion at:

- Every event
- Every second event
- Up to Every fifteenth event

**Figure 18-44. Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs**




- ETSEL - This selects which of the possible events will trigger an interrupt or start an ADC conversion.
- ETPS - This programs the event prescaling options mentioned above.
- ETFLG - These are flag bits indicating status of the selected and prescaled events.
- ETCLR - These bits allow you to clear the flag bits in the ETFLG register via software.
- ETFRC - These bits allow software forcing of an event. Useful for debugging or software intervention.
- ETINTPS - This programs the interrupt event prescaling options, supporting count and period up to 15 events.
- ETSOCPS - This programs the SOC event prescaling options, supporting count and period up to 15 events.
- ETCNTINITCTL - These bits enable ETCNTINIT initialization via SYNC event OR via software force.
- ETCNTINIT - These bits allow you to initialize INT/SOCA/SOCB counters on SYNC events (or software force) with user programmed value.

A more detailed look at how the various register bits interact with the Interrupt and ADC start of conversion logic are shown in Figure 18-45, Figure 18-46, and Figure 18-47.

Figure 18-45 shows the event-trigger's interrupt generation logic. The interrupt-period (ETPS[INTPRD]) bits specify the number of events required to cause an interrupt pulse to be generated. The choices available are:

- Do not generate an interrupt.
- Generate an interrupt on every event
- Generate an interrupt on every second event
- Generate an interrupt on every third event

On ePWM type 4, in order to enable event generation capability up to 15 events the following changes have been made. The selection made on ETPS[INTPSSEL] bit determines whether ETINTPS register, INTCNT2 and INTPRD2 bit fields determine frequency of events (interrupt once every 0-15 events).

Which event can cause an interrupt is configured by the interrupt selection (ETSEL[INTSEL]) and (ETSEL[INTSELCMP]) bits. The event can be one of the following:

- Time-base counter equal to zero (TBCTR = 0x00).
- Time-base counter equal to period (TBCTR = TBPRD).
- Time-base counter equal to zero or period (TBCTR = 0x00 || TBCTR = TBPRD).
- Time-base counter equal to the compare A register (CMPA) when the timer is incrementing.
- Time-base counter equal to the compare A register (CMPA) when the timer is decrementing.
- Time-base counter equal to the compare B register (CMPB) when the timer is incrementing.
- Time-base counter equal to the compare B register (CMPB) when the timer is decrementing.
- Time-base counter equal to the compare C register (CMPC) when the timer is incrementing.
- Time-base counter equal to the compare C register (CMPC) when the timer is decrementing.
- Time-base counter equal to the compare D register (CMPD) when the timer is incrementing.
- Time-base counter equal to the compare D register (CMPD) when the timer is decrementing.

The number of events that have occurred can be read from the interrupt event counter ETPS[INTCNT] or ETINTPS[INTCNT2] register bits based off of the selection made using ETPS[INTPSSEL]. That is, when the specified event occurs the ETPS[INTCNT] or ETINTPS[INTCNT2] bits are incremented until they reach the value specified by ETPS[INTPRD] or ETINTPS[INTPRD2] determined again by the selection made in ETPS[INTPSSEL]. When ETPS[INTCNT] = ETPS[INTPRD] the counter stops counting and its output is set. The counter is only cleared when an interrupt is sent to the PIE.

When ETPS[INTCNT] reaches ETPS[INTPRD] the following behavior will occur [The below behavior is also applicable to ETINTPS[INTCNT2] & ETINTPS[INTPRD2] :

- If interrupts are enabled, ETSEL[INTEN] = 1 and the interrupt flag is clear, ETFLG[INT] = 0, then an interrupt pulse is generated and the interrupt flag is set, ETFLG[INT] = 1, and the event counter is cleared ETPS[INTCNT] = 0. The counter will begin counting events again.
- If interrupts are disabled, ETSEL[INTEN] = 0, or the interrupt flag is set, ETFLG[INT] = 1, the counter stops counting events when it reaches the period value ETPS[INTCNT] = ETPS[INTPRD].
- If interrupts are enabled, but the interrupt flag is already set, then the counter will hold its output high until the ENTFLG[INT] flag is cleared. This allows for one interrupt to be pending while one is serviced.

Writing to the INTPRD bits will automatically clear the counter INTCNT = 0 and the counter output will be reset (so no interrupts are generated). Writing a 1 to the ETFRC[INT] bit will increment the event counter INTCNT. The counter will behave as described above when INTCNT = INTPRD. When INTPRD = 0, the counter is disabled and hence no events will be detected and the ETFRC[INT] bit is also ignored. The same applies to ETINTPS[INTCNT2] & ETINTPS[INTPRD2]

The above definition means that you can generate an interrupt on every event, on every second event, or on every third event if using the INTCNT and INTPRD. You can generate an interrupt on every event up to 15 events if using the INTCNT2 and INTPRD2.

The INTCNT2 value can be initialized with the value from ETCNTINIT[INTINIT] based on the selection made in ETCNTINITCTL[INTINITEN]. When ETCNTINITCTL[INTINITEN] is set, then it enables initialization of INTCNT2 counter with contents of ETCNTINIT[INTINIT] on a SYNC event or software force determined by ETCNTINITCTL[INTINITFRC] .

**Figure 18-45. Event-Trigger Interrupt Generator**

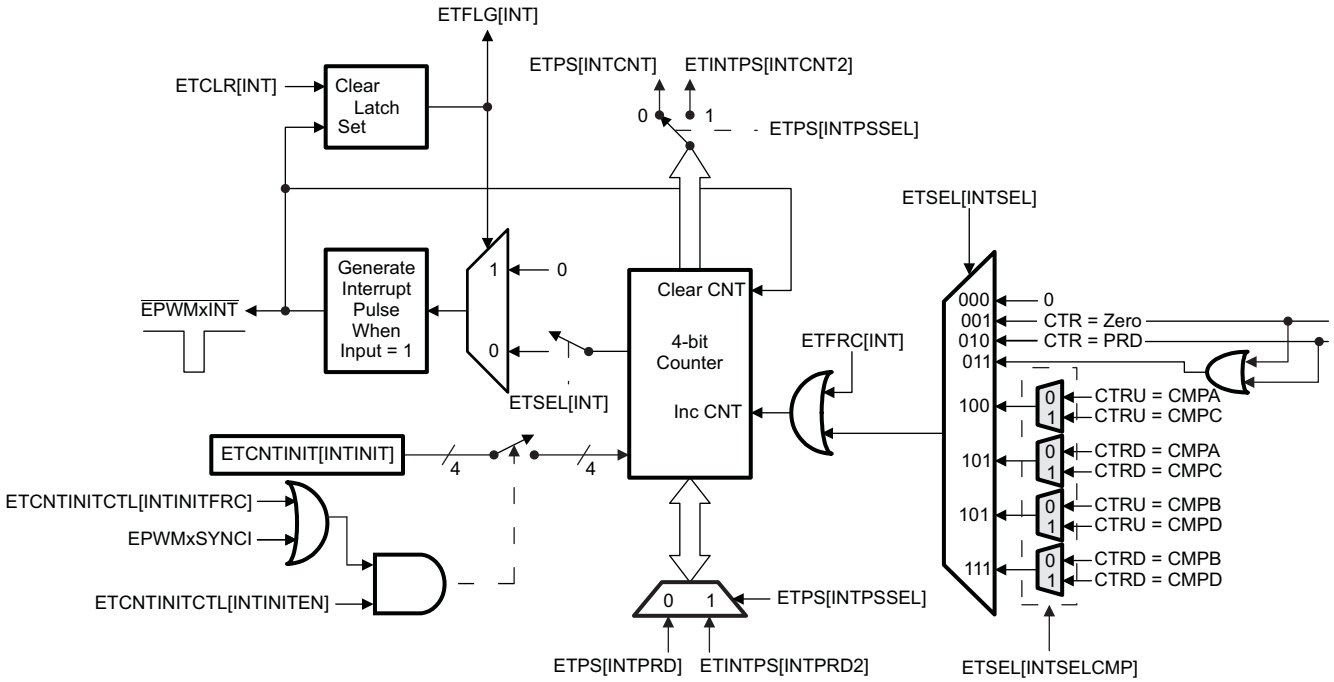
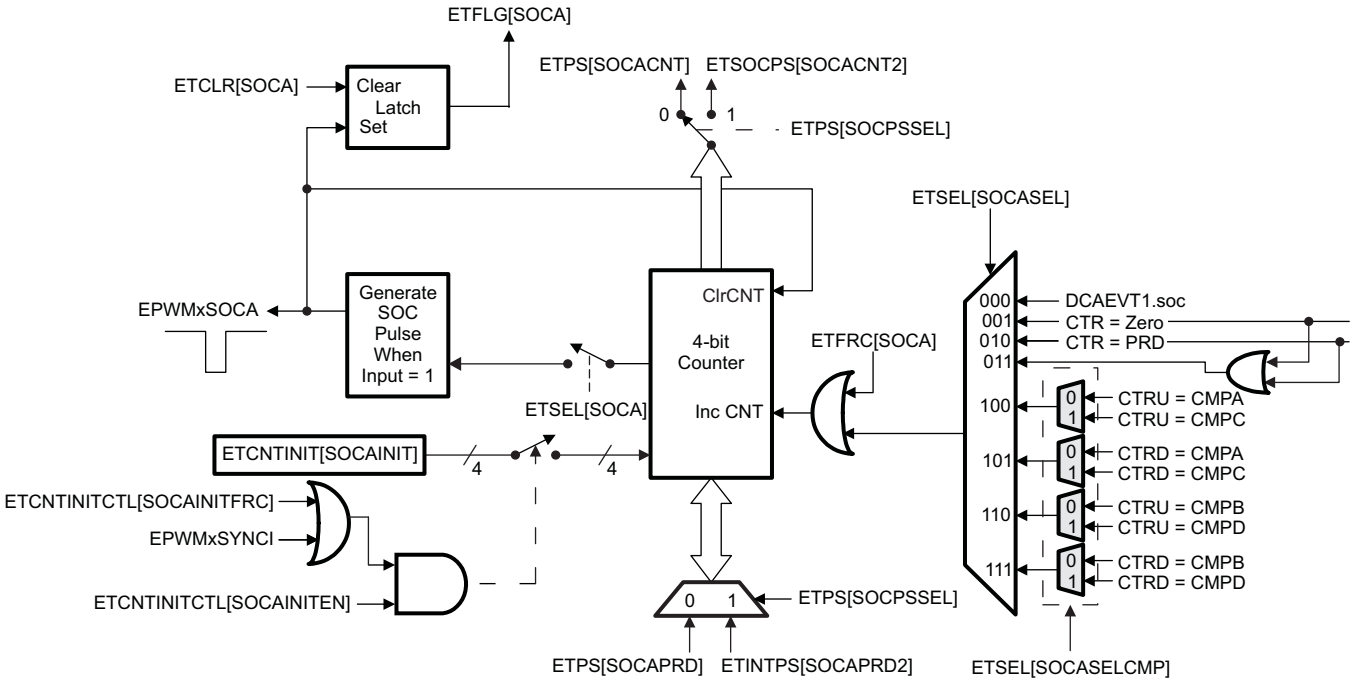



Figure 18-46 shows the operation of the event-trigger's start-of-conversion-A (SOCA) pulse generator. The enhancements include SOCASELCMP and SOCBSELCMP bit fields defined in the ETSEL register enable CMPC and CMPD events respectively to cause a start of conversion. The ETPS[SOCPSSEL] bit field determines whether SOCACNT2 and SOCAPRD2 take control or not. The ETPS[SOCACNT] counter and ETPS[SOCAPRD] period values behave similarly to the interrupt generator except that the pulses are continuously generated. That is, the pulse flag ETFLG[SOCA] is latched when a pulse is generated, but it does not stop further pulse generation. The enable/disable bit ETSEL[SOCAEN] stops pulse generation, but input events can still be counted until the period value is reached as with the interrupt generation logic. The event that will trigger an SOCA and SOCB pulse can be configured separately in the ETSEL[SOCASEL] and ETSEL[SOCBSEL] bits. The possible events are the same events that can be specified for the interrupt generation logic with the addition of the DCAEVT1.soc and DCBEVT1.soc event signals from the digital compare (DC) submodule. The SOCACNT2 initialization scheme is very similar to the interrupt generator with respective enable, value initialize and SYNC or software force options.

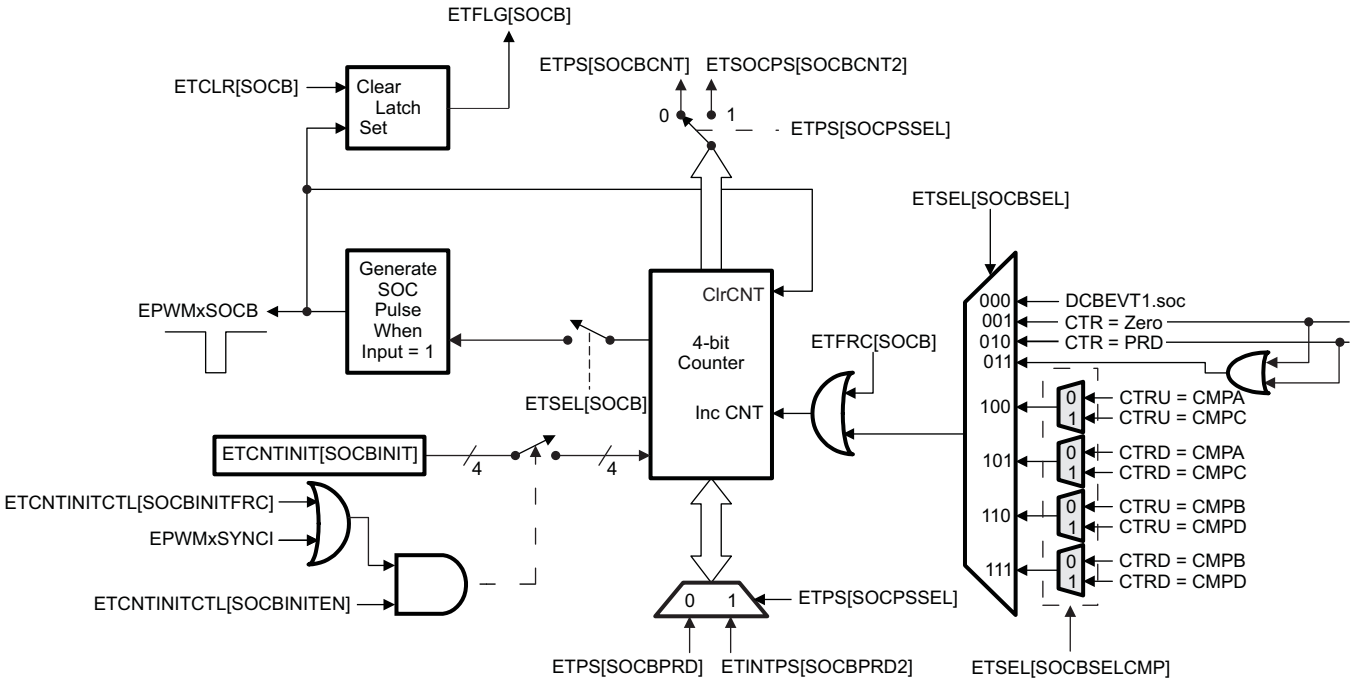
**Figure 18-46. Event-Trigger SOCA Pulse Generator**




A The DCAEVT1.soc signals are signals generated by the Digital compare (DC) submodule in Section 18.3.9.

Figure 18-47 shows the operation of the event-trigger's start-of-conversion-B (SOCB) pulse generator. The event-trigger's SOCB pulse generator operates the same way as the SOCA.

**Figure 18-47. Event-Trigger SOCB Pulse Generator**

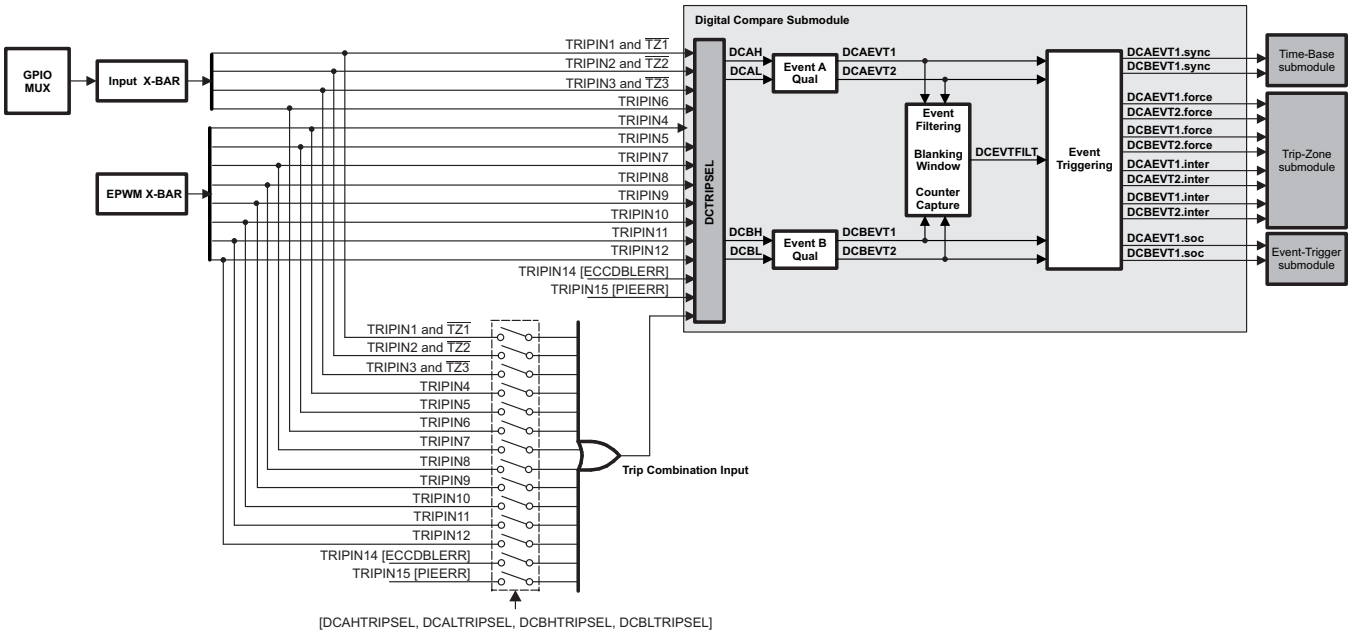



A The DCBEVT1.soc signals are signals generated by the Digital compare (DC) submodule in Section 18.3.9.

### 18.3.9 Digital Compare (DC) Submodule

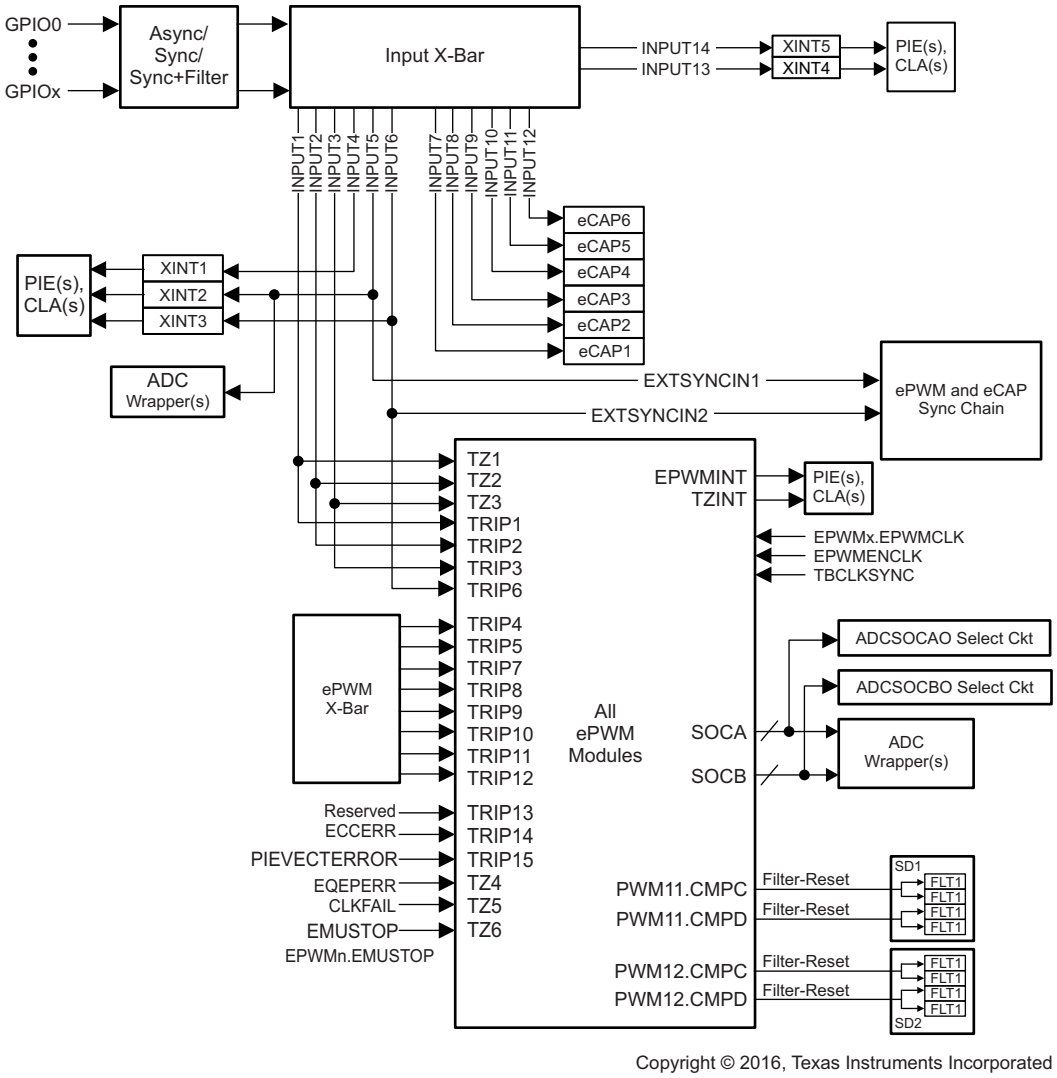
Figure 18-48 illustrates where the digital compare (DC) submodule signals interface to other submodules in the ePWM system.

**Figure 18-48. Digital-Compare Submodule High-Level Block Diagram**




The eCAP input signals are sourced from the Input X-BAR signals as shown in Figure 18-49.

**Figure 18-49. GPIO MUX-to-Trip Input Connectivity**






On this device, any of the GPIO pins can be flexibly mapped to be the trip-zone input and/or trip inputs to the trip-zone submodule and digital compare submodule. The Input X-BAR Input Select (INPUTxSELECT) register defines which GPIO pins gets assigned to be the trip-zone inputs / trip inputs.

The digital compare (DC) submodule compares signals external to the ePWM module (for instance, CMPSSx signals from the analog comparators) to directly generate PWM events/actions which then feed to the event-trigger, trip-zone, and time-base submodules. Additionally, blanking window functionality is supported to filter noise or unwanted pulses from the DC event signals.

**NOTE:** The user is responsible for driving correct state on the selected pin before enabling clock and configuring the trip input for the respective ePWM peripheral to avoid spurious latch of TRIP signal.

### 18.3.9.1 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 logic externally using the GPIO peripheral, internal PIE, 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 which can then either be filtered or fed directly to the trip-zone, event-trigger, and time-base submodules to:
  - generate a trip zone interrupt
  - generate an ADC start of conversion
  - force an event
  - generate a synchronization event for synchronizing the ePWM module TBCTR.
- Event filtering (blanking window logic) can optionally blank the input signal to remove noise.

#### 18.3.9.2 Enhanced Trip Action

In order to allow multiple comparators at a time to affect DCA/BEVTx events and trip actions, there is a OR logic to bring together ALL trip inputs (up to 15) from sources external to the ePWM module and feed into DCAH, DCAL, DCBH, and DCBL as “combinational input” using the DCTRIPSEL register. This is configured by writing appropriate value [Trip Combination input ] to the DCAHCOMPSEL,DCALCOMPSEL, DCBHCOMPSEL, DCBLCOMPSEL bit fields in the DCTRIPSEL register.

The user has an discrete choice for which trip input to put through the combinational logic for Digital Compare A High/Low (DCAH, DCAL) and Digital Compare B High/Low (DCBH, DCBL) signals generation. This is achieved using the selection from DCAHTRIPSEL, DCALTRIPSEL, DCBHTRIPSEL and DCBLTRIPSEL register. The appropriate bit when set indicates that Trip input is chosen for “combinational input” by the DCTRIPSEL register.

Apart from these options user can also make the external trip inputs which feed into the OR gate individually selectable and not go through the “combinational input” by using the DCTRIPSEL register.

#### 18.3.9.3 Using CMPSS to Trip the ePWM on a Cycle-by-Cycle Basis

When using the CMPSS to trip the ePWM on a cycle-by-cycle basis, steps should be taken to prevent an asserted comparator trip state in one PWM cycle from extending into the following cycle. The CMPSS can be used to signal a trip condition to the downstream ePWM modules. For applications like peak current mode control, only one trip event per PWM cycle is expected. Under certain conditions, it is possible for a sustained or late trip event (arriving near the end of a PWM cycle) to carry over into the next PWM cycle if precautions are not taken. If either the CMPSS Digital Filter or the ePWM Digital Compare (DC) submodule is configured to qualify the comparator trip signal, “N” number of clock cycles of qualification will be introduced before the EPWM trip logic can respond to logic changes of the trip signal. Once an ePWM trip condition is qualified, the trip condition will remain active for N clock cycles after the comparator trip signal has de-asserted. If a qualified comparator trip signal remains asserted within N clock cycles prior to the end of a PWM cycle, the trip condition will not be cleared until after the following PWM cycle has started. Thus, the new PWM cycle will detect a trip condition as soon as it begins.

To avoid this undesired trip condition, the user application should take steps to ensure that the qualified trip signal seen by the ePWM trip logic is deasserted prior to the end of each PWM cycle. This can be accomplished through various methods:

- Design the system such that a comparator trip will not be asserted within N clock cycles prior to the end of the PWM cycle.
- Activate blanking of the comparator trip signal via the EPWM event filter at least two clock cycles prior to the PWMSYNC signal and continue blanking for at least N clock cycles into the next PWM cycle.
- If the CMPSS COMPxLATCH path is used, clear the COMPxLATCH at least N clock cycles prior to the end of the PWM cycle. The latch can be cleared by software (via COMPSTSCLR) or by generating an early PWMSYNC signal. The ePWM modules on this device include the ability to generate PWMSYNC upon a CMPC or CMPD match (via HRPCTL) for arbitrary PWMSYNC placement within the PWM cycle.

#### 18.3.9.4 Operation Highlights of the Digital Compare Submodule

The following sections describe the operational highlights and configuration options for the digital compare submodule.

#### 18.3.9.4.1 Digital Compare Events

As illustrated in Section 18.3.9.4.2 earlier in this section, trip zone inputs ($\overline{TZ1}$, $\overline{TZ2}$, and $\overline{TZ3}$) and CMPSSx signals from the analog comparator (COMP) module can be selected via the DCTRIPSEL bits to generate the Digital Compare A High and Low (DCAH/L) and Digital Compare B High and Low (DCBH/L) signals. Then, the configuration of the 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 may or may not get latched by CBC or OST latches.

The DCAEVT1/2 and DCBEVT1/2 events can then be filtered to provide a filtered version of the event signals (DCEVTFILT) or the filtering can be bypassed. Filtering is discussed further in Section 18.3.9.4.2. Either the DCAEVT1/2 and DCBEVT1/2 event signals or the filtered DCEVTFILT event signals can generate a force to the trip zone module, a TZ interrupt, an ADC SOC, or a PWM sync signal.

- **force signal:**

  DCAEVT1/2.force signals force trip zone conditions which either directly influence the output on the EPWMxA pin (via TZCTL, TZCTLDCA, TZCTLDCB register configurations) or, if the DCAEVT1/2 signals are selected as one-shot or cycle-by-cycle trip sources (via the TZSEL register), the DCAEVT1/2.force signals can effect the trip action via the TZCTL or TZCTL2 register configurations. The DCBEVT1/2.force signals behaves similarly, but affect the EPWMxB output pin instead of the EPWMxA output pin.

  The priority of conflicting actions on the TZCTL, TZCTL2, TZCTLDCA and TZCTLDCB registers is as follows (highest priority overrides lower priority):

  Output EPWMxA:

  – TZA (highest) -> DCAEVT1 -> DCAEVT2 (lowest)
  – TZAU (highest) -> DCAEVT1U -> DCAEVT2U (lowest)
  – TZAD (highest) -> DCAEVT1D -> DCAEVT2D (lowest)

  Output EPWMxB:

  – TZB (highest) -> DCBEVT1 -> DCBEVT2 (lowest)
  – TZBU (highest) -> DCBEVT1U -> DCBEVT2U (lowest)
  – TZBD (highest) -> DCBEVT1D -> DCBEVT2D (lowest)

- **interrupt signal:**

  DCAEVT1/2.interrupt signals generate trip zone interrupts to the PIE. To enable the interrupt, the user must set the DCAEVT1, DCAEVT2, DCBEVT1, or DCBEVT2 bits in the TZEINT register. Once one of these events occurs, an EPWMxTZINT interrupt is triggered, and the corresponding bit in the TZCLR register must be set in order to clear the interrupt.

- **soc signal:**

  The DCAEVT1.soc signal interfaces with the event-trigger submodule and can be selected as an event which generates an ADC start-of-conversion-A (SOCA) pulse via the ETSEL[SOCASEL] bit. Likewise, the DCBEVT1.soc signal can be selected as an event which generates an ADC start-of-conversion-B (SOCB) pulse via the ETSEL[SOCBSEL] bit.

- **sync signal:**

  The DCAEVT1.sync and DCBEVT1.sync events are ORed with the EPWMxSYNCI input signal and the TBCTL[SWFSYNC] signal to generate a synchronization pulse to the time-base counter.

The diagrams below show how the DCAEVT1, DCAEVT2 or DCEVTFLT signals are processed to generate the digital compare A event force, interrupt, soc and sync signals.

**Figure 18-50. DCAEVT1 Event Triggering**

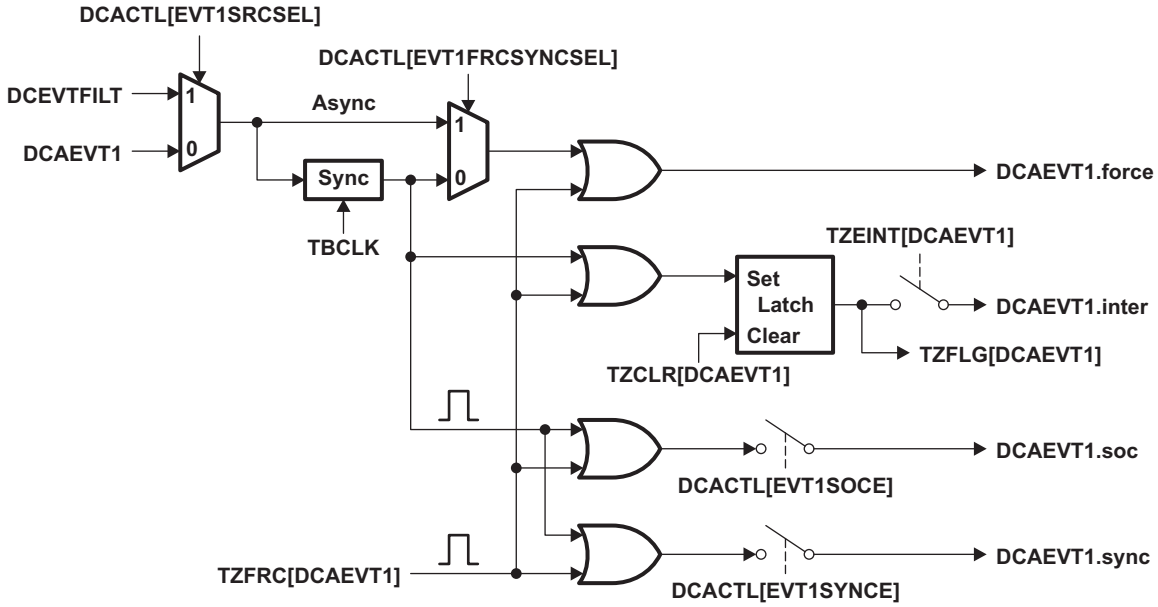



**Figure 18-51. DCAEVT2 Event Triggering**

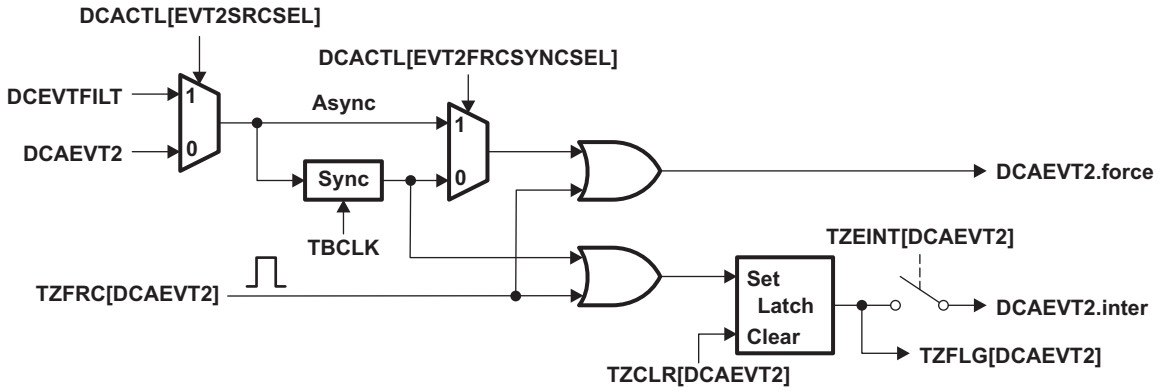


Figure 18-52 and Figure 18-53 show how the DCBEVT1, DCBEVT2 or DCEVTFLT signals are processed to generate the digital compare B event force, interrupt, soc and sync signals.

**Figure 18-52. DCBEVT1 Event Triggering**

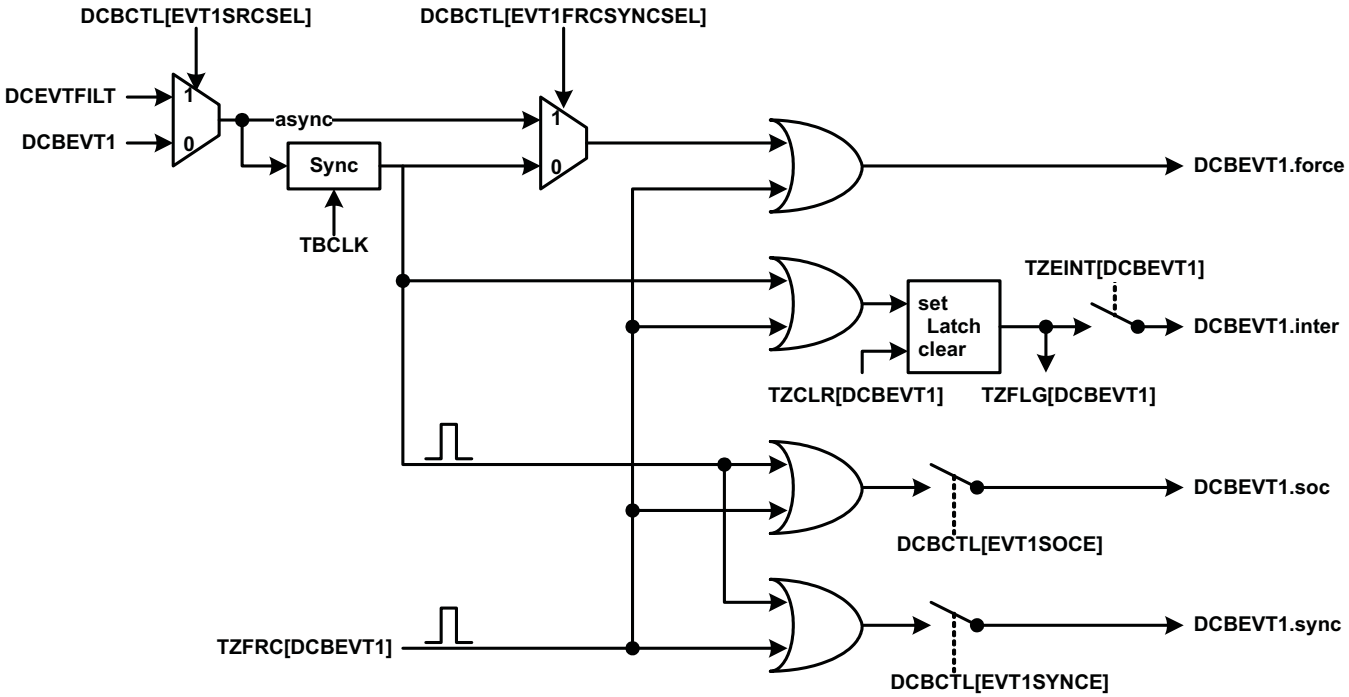
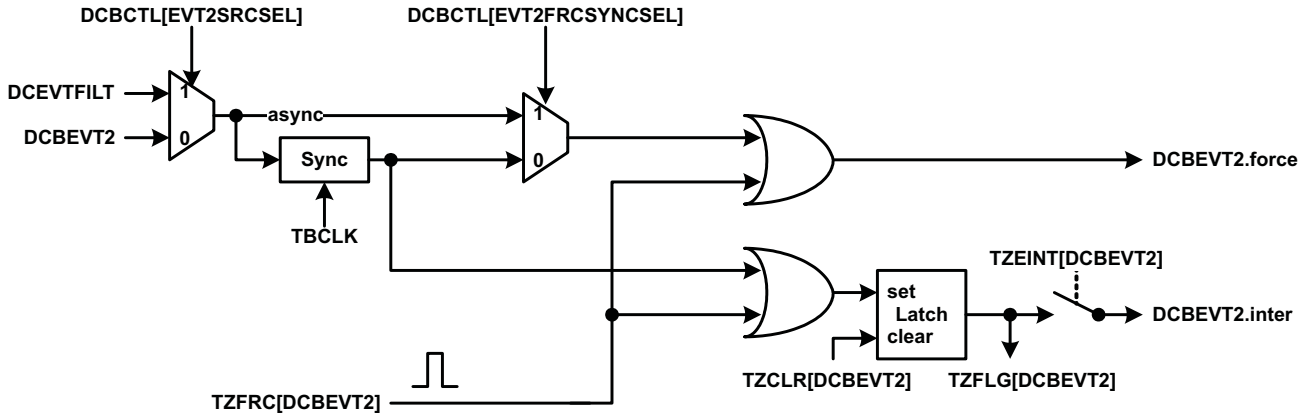


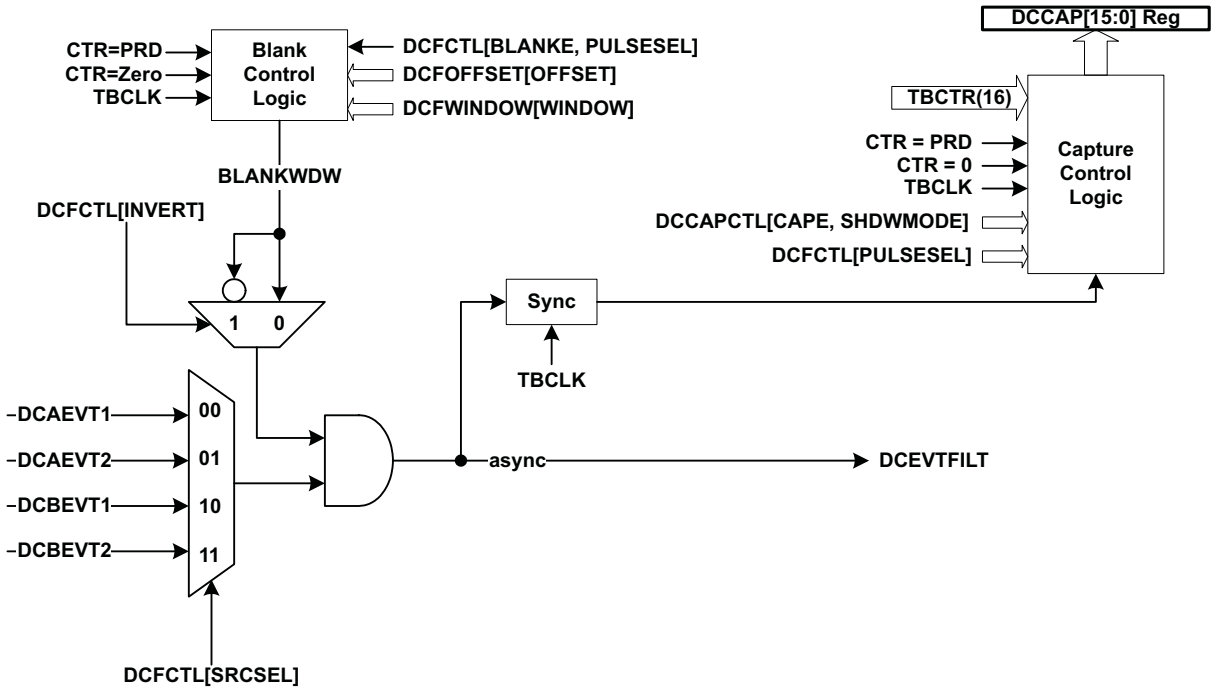

**Figure 18-53. DCBEVT2 Event Triggering**




### 18.3.9.4.2 Event Filtering

The DCAEVT1/2 and DCBEVT1/2 events can be filtered via event filtering logic to remove noise by optionally blanking events for a certain period of time. This is useful for cases where the analog comparator outputs may be selected to trigger DCAEVT1/2 and DCBEVT1/2 events, and the blanking logic is used to filter out potential noise on the signal prior to tripping the PWM outputs or generating an interrupt or ADC start-of-conversion. The event filtering can also capture the TBCTR value of the trip event. Figure 18-54 shows the details of the event filtering logic.

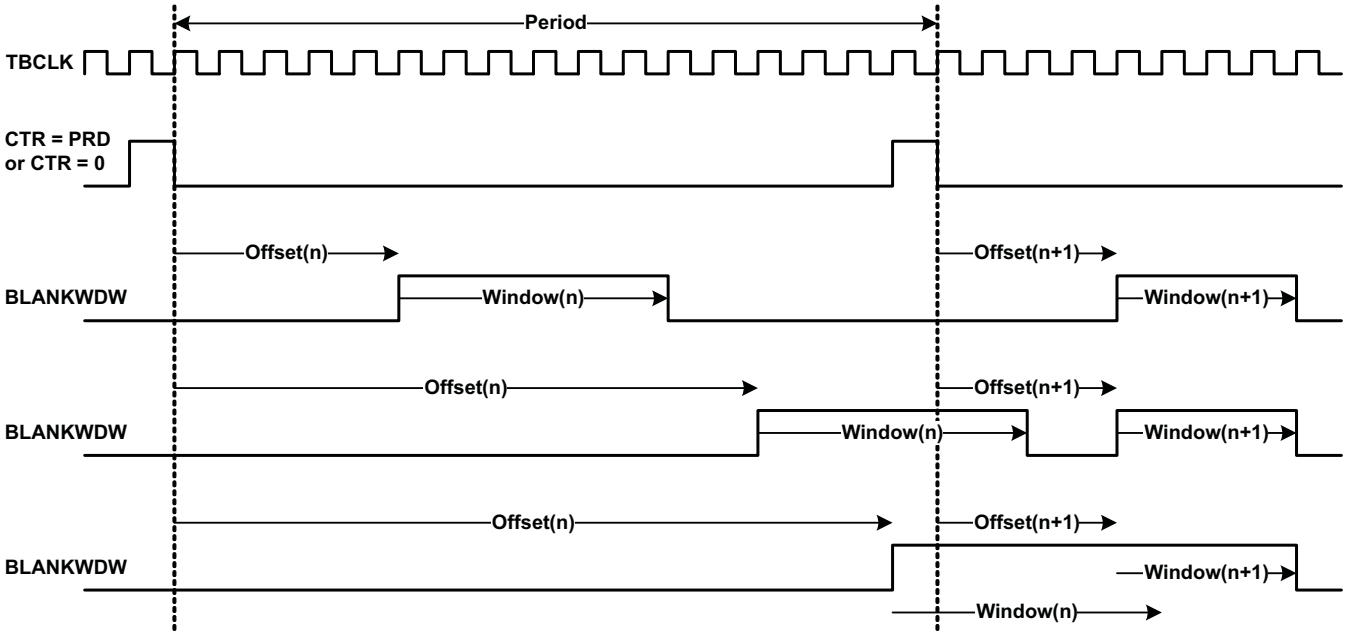
**Figure 18-54. Event Filtering**




If the blanking logic is enabled, one of the digital compare events – DCAEVT1, DCAEVT2, DCBEVT1, DCBEVT2 – is selected for filtering. The blanking window, which filters out all event occurrences on the signal while it is active, will be aligned to either a CTR = PRD pulse or a CTR = 0 pulse or both CTR = PRD and CTR = 0 (configured by the DCFCTL[PULSESEL] bits). An offset value in TBCLK counts is programmed into the DCFOFFSET register, which determines at what point after the CTR = PRD or CTR = 0 pulse the blanking window starts. The duration of the blanking window, in number of TBCLK counts after the offset counter expires, is written to the DCFWINDOW register by the application. During the blanking window, all events are ignored. Before and after the blanking window ends, events can generate soc, sync, interrupt, and force signals as before.

The diagram below illustrates several timing conditions for the offset and blanking window within an ePWM period. Notice that if the blanking window crosses the CTR = 0 or CTR = PRD boundary, the next window still starts at the same offset value after the CTR = 0 or CTR = PRD pulse.

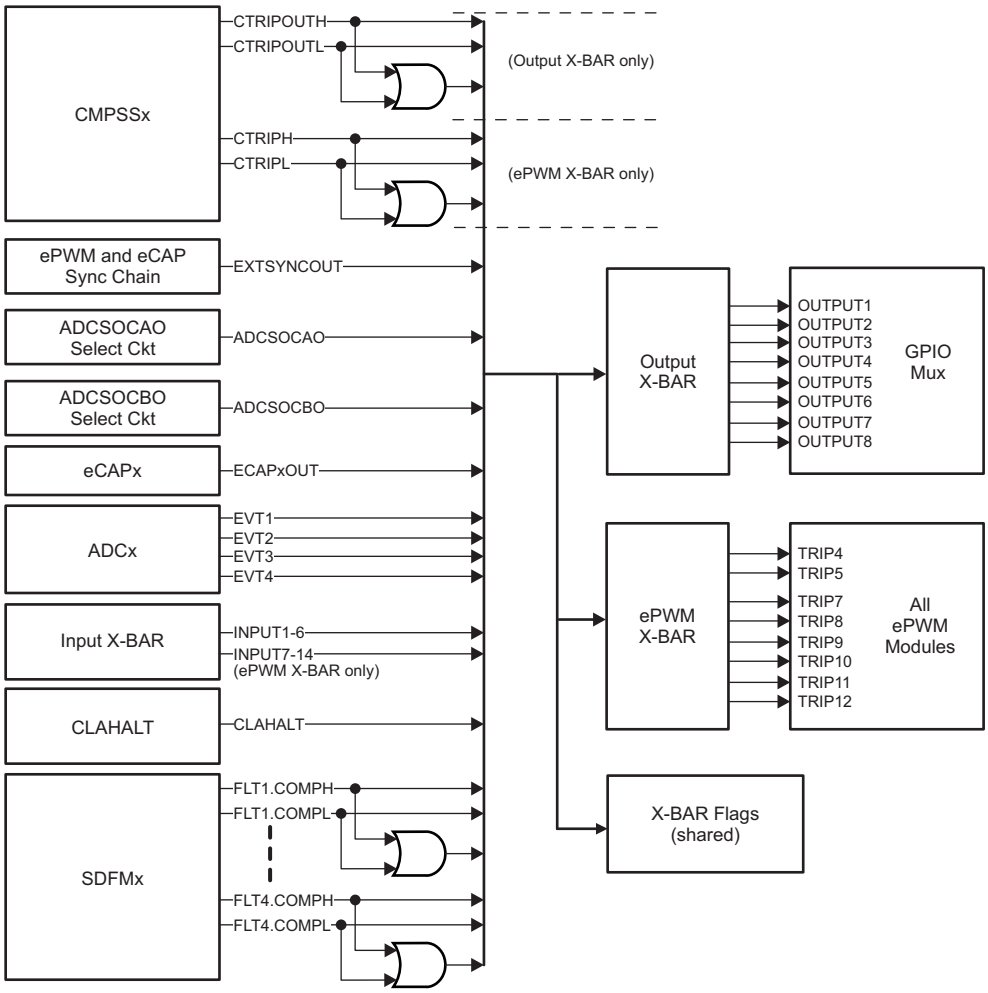
**Figure 18-55. Blanking Window Timing Diagram**




### 18.3.10 EPWM X-BAR

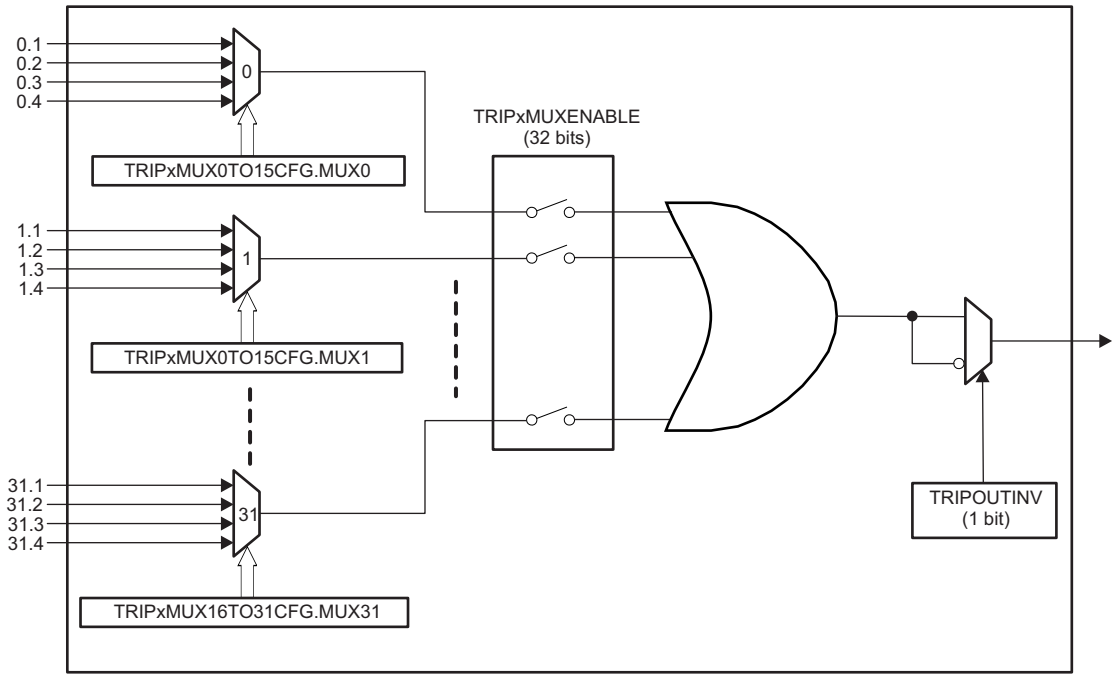
Figure 18-56 shows the architecture of the EPWM X-Bar. This module enables selection of various trigger sources into any of the eight dedicated ETWPM trips inputs, namely the TRIP4, TRIP5, TRIP7, TRIP8, TRIP9, TRIP10, TRIP11 and TRIP12.

**Figure 18-56. EPWM X-BAR**




The EPWM X-BAR has eight outputs which are routed to each EPWM module. Figure 18-57 represents the architecture of a single output but it is identical to the architecture of all of the other outputs:

**Figure 18-57. EPWM Architecture - Single Output**




First, determine the signal(s) which should be passed to the EPWM by referencing the EPWM X-Bar Input Table. You may select up to one signal per mux (31 total muxes) for each TRIPx output. Select the inputs to each mux via the TRIPxMUX0TO15CFG and TRIPxMUX16TO31CFG registers. In order to pass any signal through to the EPWM, you must also enable the mux in the TRIPxMUXENABLE register. All muxes which are enabled will be logically OR'd before being passed on to the respective TRIPx signal on the EPWM. You may also optionally invert the signal via the TRIPOUTINV register.

**Table 18-13. ePWM X-BAR Mux Configuration Table**

| Mux | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 0 | CMPSS1.CTRIPH | CMPSS1.CTRIPH_OR_CTRIPL | ADCAEVT1 | ECAP1OUT |
| 1 | CMPSS1.CTRIPL | INPUTXBAR1 | | ADCCEVT1 |
| 2 | CMPSS2.CTRIPH | CMPSS2.CTRIPH_OR_CTRIPL | ADCAEVT2 | ECAP2OUT |
| 3 | CMPSS2.CTRIPL | INPUTXBAR2 | | ADCCEVT2 |
| 4 | CMPSS3.CTRIPH | CMPSS3.CTRIPH_OR_CTRIPL | ADCAEVT3 | ECAP3OUT |
| 5 | CMPSS3.CTRIPL | INPUTXBAR3 | | ADCCEVT3 |
| 6 | CMPSS4.CTRIPH | CMPSS4.CTRIPH_OR_CTRIPL | ADCAEVT4 | ECAP4OUT |
| 7 | CMPSS4.CTRIPL | INPUTXBAR4 | | ADCCEVT4 |
| 8 | CMPSS5.CTRIPH | CMPSS5.CTRIPH_OR_CTRIPL | ADCBEVT1 | ECAP5OUT |
| 9 | CMPSS5.CTRIPL | INPUTXBAR5 | | |
| 10 | CMPSS6.CTRIPH | CMPSS6.CTRIPH_OR_CTRIPL | ADCBEVT2 | ECAP6OUT |
| 11 | CMPSS6.CTRIPL | INPUTXBAR6 | | |
| 12 | CMPSS7.CTRIPH | CMPSS7.CTRIPH_OR_CTRIPL | ADCBEVT3 | ECAP7OUT |
| 13 | CMPSS7.CTRIPL | ADCSOCAO | | |
| 14 | | | ADCBEVT4 | EXTSYNCOUT |
| 15 | | ADCSOCBO | | |
| 16 | SD1FLT1.COMPH | SD1FLT1.COMPH_OR_COMPL | | |
| 17 | SD1FLT1.COMPL | INPUT7 | | CLAHALT |
| 18 | SD1FLT2.COMPH | SD1FLT2.COMPH_OR_COMPL | | |
| 19 | SD1FLT2.COMPL | INPUT8 | | |
| 20 | SD1FLT3.COMPH | SD1FLT3.COMPH_OR_COMPL | | |
| 21 | SD1FLT3.COMPL | INPUT9 | | |
| 22 | SD1FLT4.COMPH | SD1FLT4.COMPH_OR_COMPL | | |
| 23 | | INPUT10 | | |
| 24 | | | | |
| 25 | | INPUT11 | | |
| 26 | | | | |
| 27 | | INPUT12 | | |
| 28 | | | | |
| 29 | | INPUT13 | | |
| 30 | | | | |
| 31 | | INPUT14 | | |

**Note:** All unused and reserved positions are tied to ‘0’.

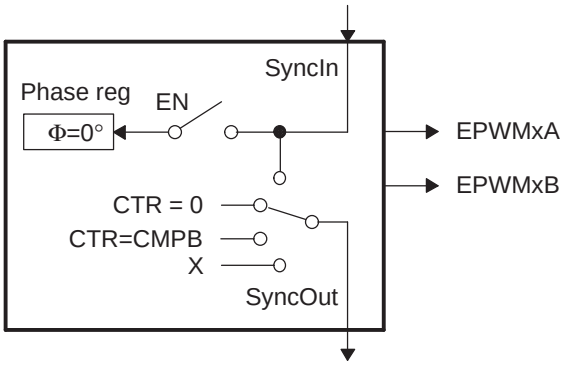
## 18.4 Applications to Power Topologies

An ePWM module has all the local resources necessary to operate completely as a standalone module or to operate in synchronization with other identical ePWM modules.

### 18.4.1 Overview of Multiple Modules

Previously in this chapter, all discussions have described the operation of a single module. To facilitate the understanding of multiple modules working together in a system, the ePWM module described in reference is represented by the more simplified block diagram shown in Figure 18-58. This simplified ePWM block shows only the key resources needed to explain how a multiswitch power topology is controlled with multiple ePWM modules working together.

**Figure 18-58. Simplified ePWM Module**



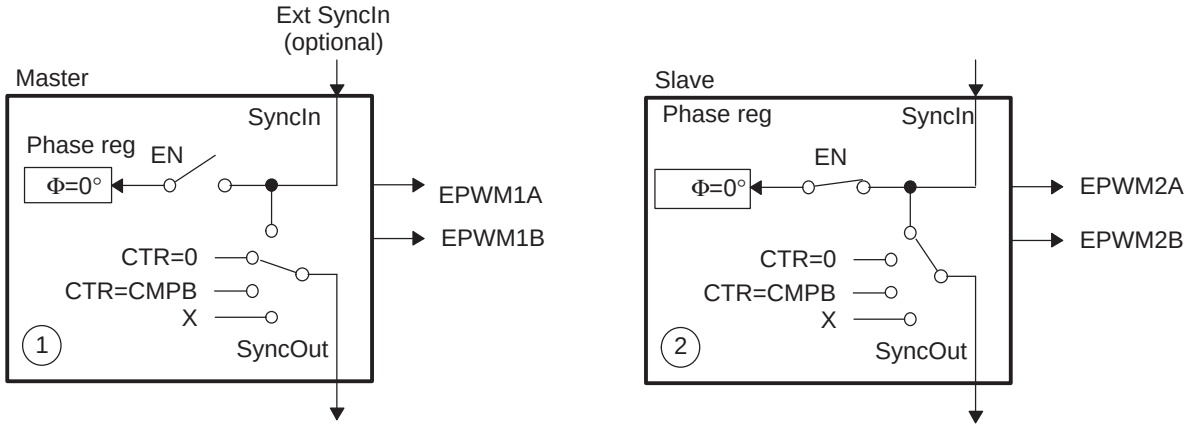

### 18.4.2 Key Configuration Capabilities

The key configuration choices available to each module are as follows:

- Options for SyncIn
  - Load own counter with phase register on an incoming sync strobe—enable (EN) switch closed
  - Do nothing or ignore incoming sync strobe—enable switch open
  - Sync flow-through - SyncOut connected to SyncIn
  - Master mode, provides a sync at PWM boundaries—SyncOut connected to CTR = PRD
  - Master mode, provides a sync at any programmable point in time—SyncOut connected to CTR = CMPB
  - Module is in standalone mode and provides No sync to other modules—SyncOut connected to X (disabled)
- Options for SyncOut
  - Sync flow-through - SyncOut connected to SyncIn
  - Master mode, provides a sync at PWM boundaries—SyncOut connected to CTR = PRD
  - Master mode, provides a sync at any programmable point in time—SyncOut connected to CTR = CMPB
  - Module is in standalone mode and provides No sync to other modules—SyncOut connected to X (disabled)

For each choice of SyncOut, a module may also choose to load its own counter with a new phase value on a SyncIn strobe input or choose to ignore it (that is, via the enable switch). Although various combinations are possible, the two most common—master module and slave module modes—are shown in Figure 18-59.

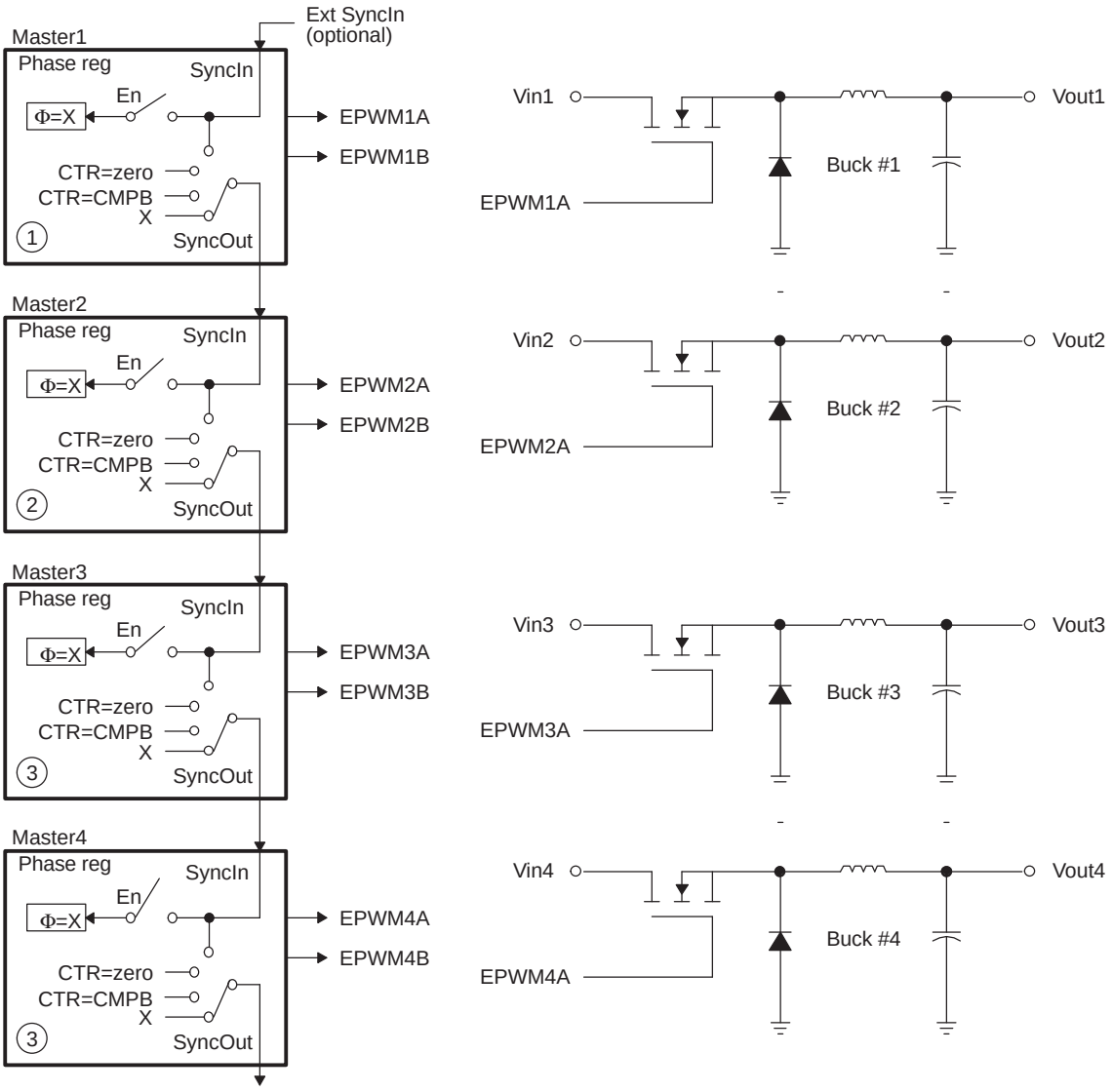
**Figure 18-59. EPWM1 Configured as a Typical Master, EPWM2 Configured as a Slave**




### 18.4.3 Controlling Multiple Buck Converters With Independent Frequencies

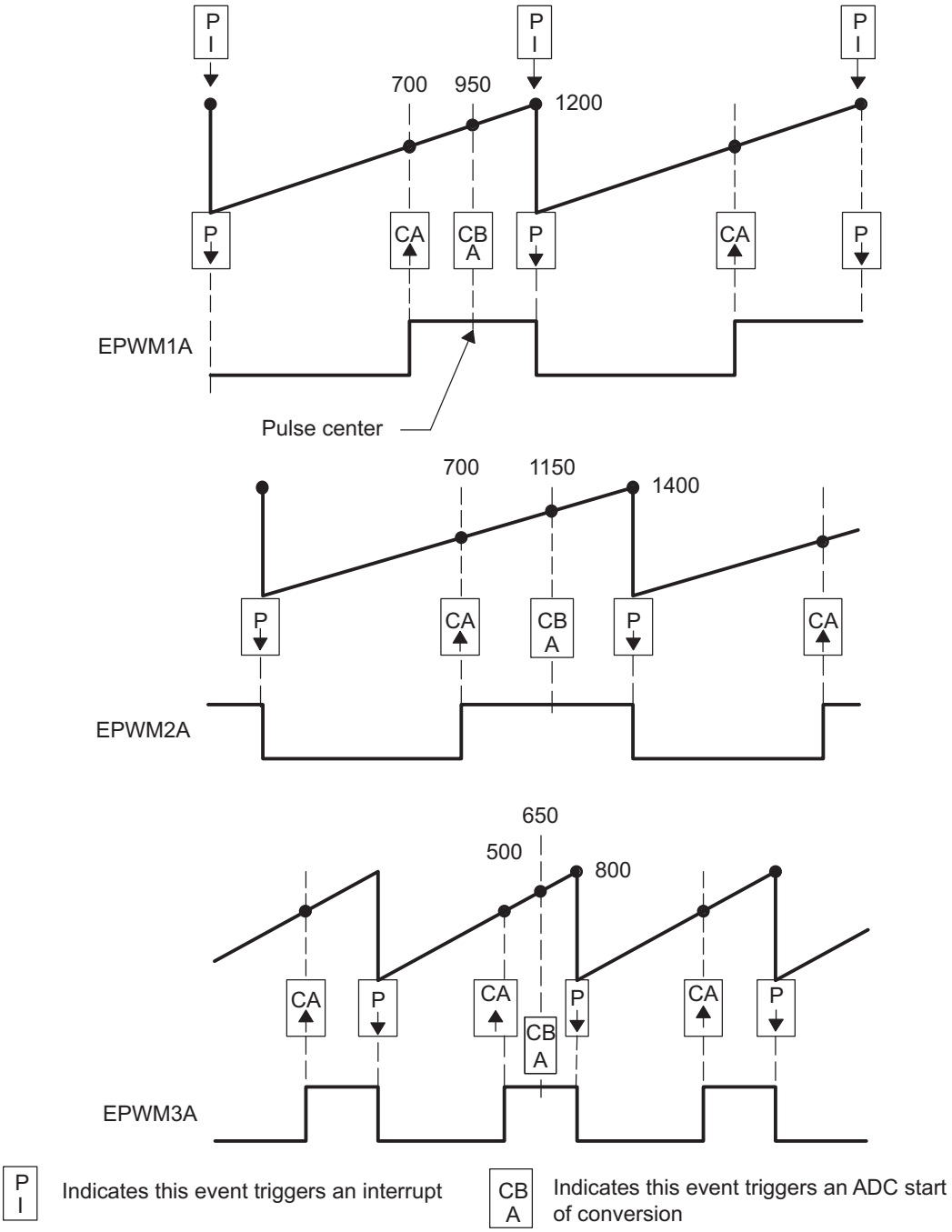
One of the simplest power converter topologies is the buck. A single ePWM module configured as a master can control two buck stages with the same PWM frequency. If independent frequency control is required for each buck converter, then one ePWM module must be allocated for each converter stage. Figure 18-60 shows four buck stages, each running at independent frequencies. In this case, all four ePWM modules are configured as Masters and no synchronization is used. Figure 18-61 shows the waveforms generated by the setup shown in Figure 18-60; note that only three waveforms are shown, although there are four stages.

**Figure 18-60. Control of Four Buck Stages. Here $F_{PWM1} \neq F_{PWM2} \neq F_{PWM3} \neq F_{PWM4}$**




NOTE: φ = X indicates value in phase register is a "don't care"

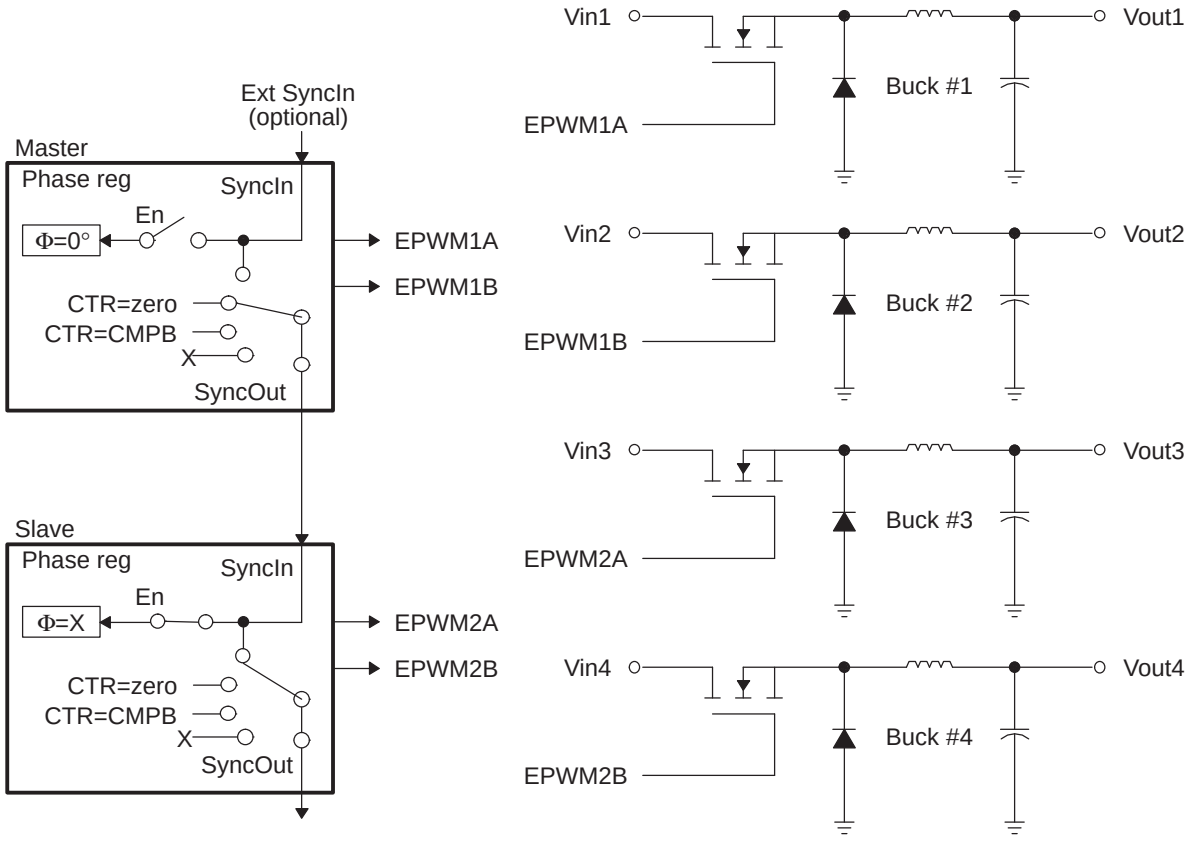
**Figure 18-61. Buck Waveforms for Figure 18-60 (Note: Only three bucks shown here)**




### 18.4.4 Controlling Multiple Buck Converters With Same Frequencies

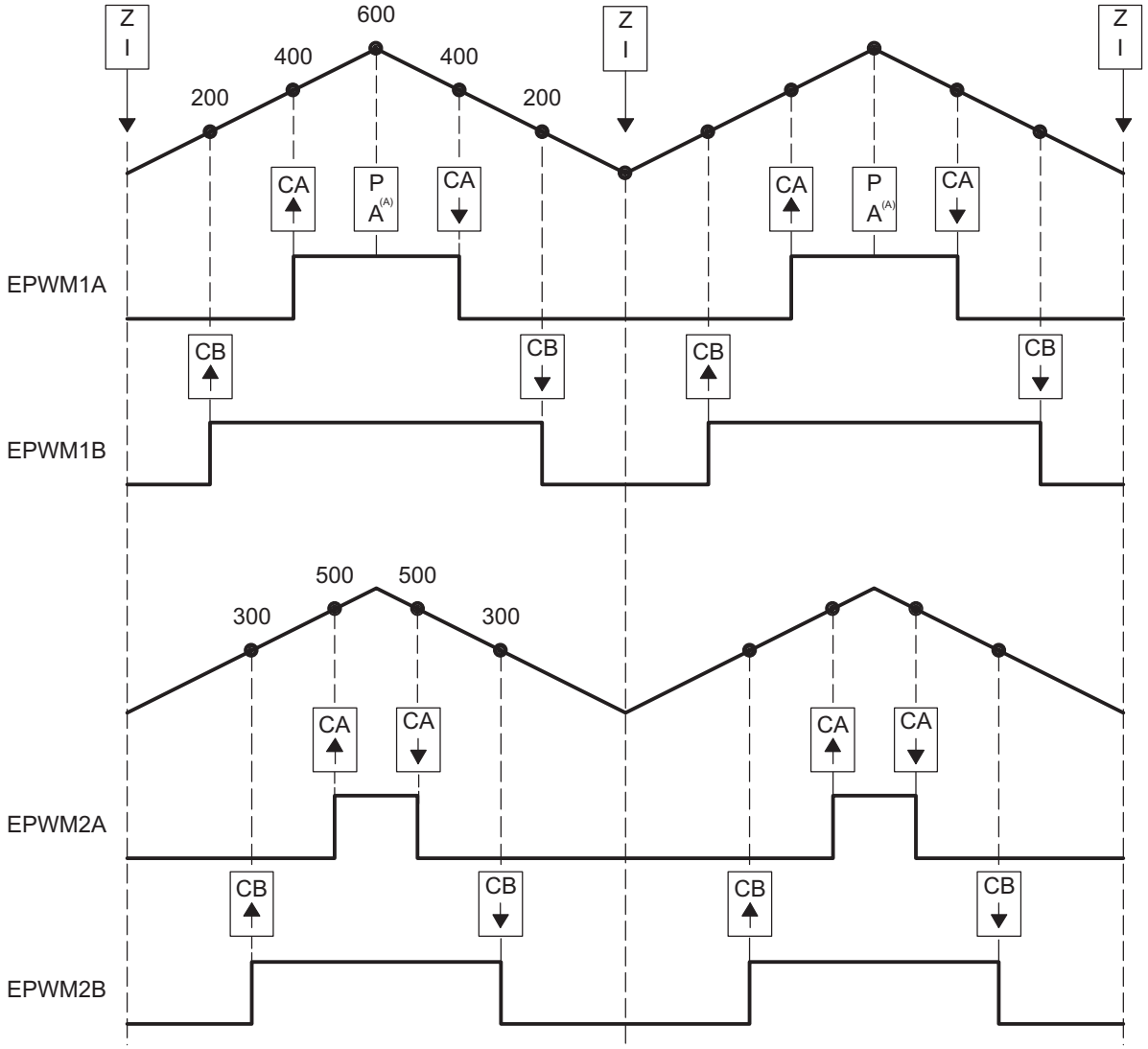
If synchronization is a requirement, ePWM module 2 can be configured as a slave and can operate at integer multiple (N) frequencies of module 1. The sync signal from master to slave ensures these modules remain locked. Figure 18-62 shows such a configuration; Figure 18-63 shows the waveforms generated by the configuration.

**Figure 18-62. Control of Four Buck Stages. (Note: $F_{PWM2}$ = N x $F_{PWM1}$)**




NOTE: φ = X indicates value in phase register is a "don't care"

**Figure 18-63. Buck Waveforms for Figure 18-62 (Note: $F_{PWM2} = F_{PWM1)}$)**




A Starts ADC conversion.

### 18.4.5 Controlling Multiple Half H-Bridge (HHB) Converters

Topologies that require control of multiple switching elements can also be addressed with these same ePWM modules. It is possible to control a Half-H bridge stage with a single ePWM module. This control can be extended to multiple stages. Figure 18-64 shows control of two synchronized Half-H bridge stages where stage 2 can operate at integer multiple (N) frequencies of stage 1. Figure 18-65 shows the waveforms generated by the configuration shown in Figure 18-64.

Module 2 (slave) is configured for Sync flow-through; if required, this configuration allows for a third Half-H bridge to be controlled by PWM module 3 and also, most importantly, to remain in synchronization with master module 1.

**Figure 18-64. Control of Two Half-H Bridge Stages ($F_{PWM2}$ = N x $F_{PWM1}$)**

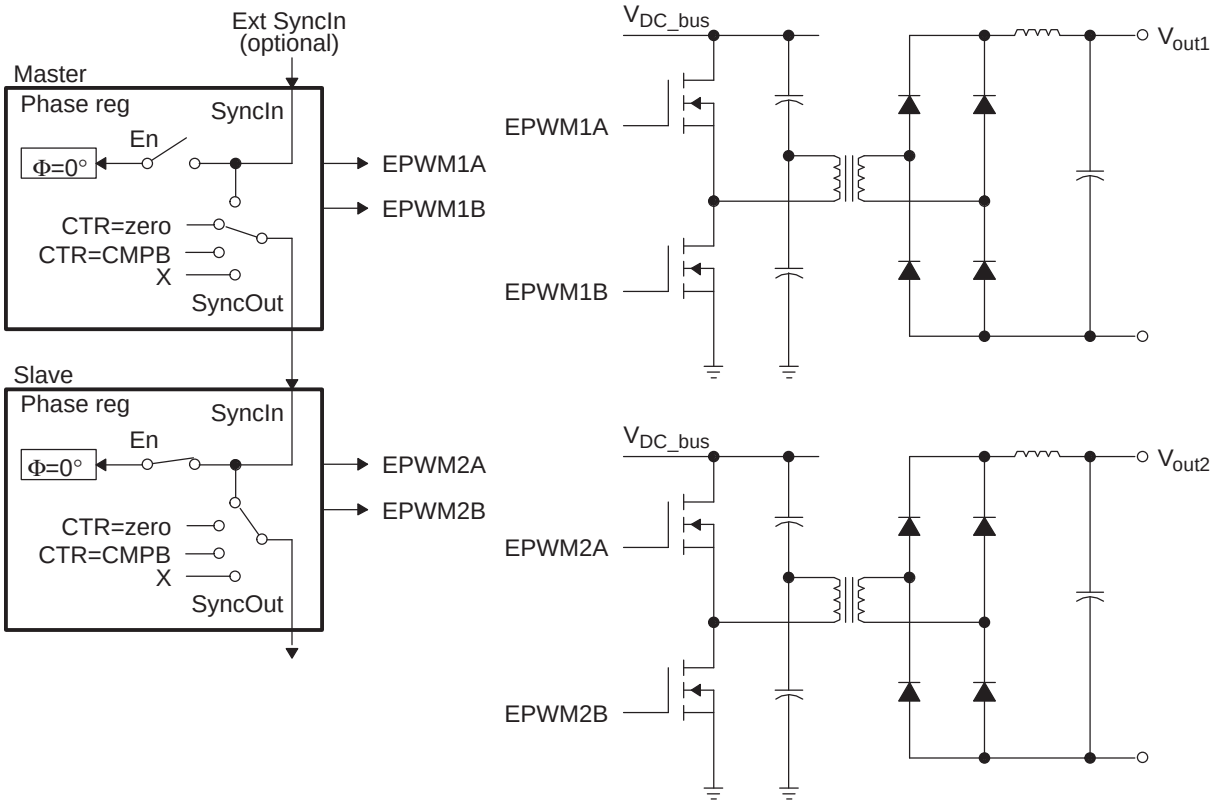
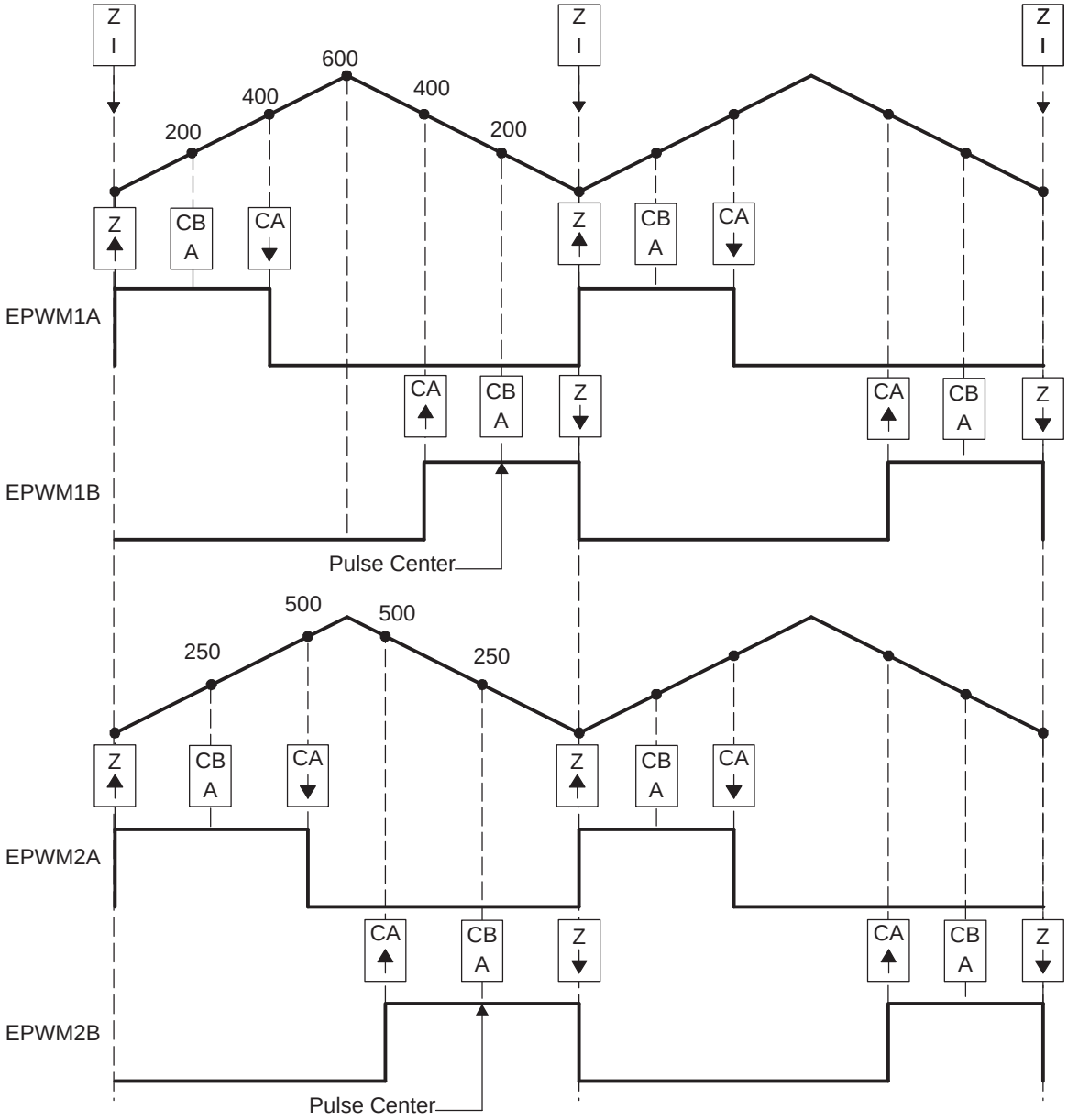


**Figure 18-65. Half-H Bridge Waveforms for Figure 18-64 (Note: Here $F_{PWM2} = F_{PWM1}$ )**




### 18.4.6 Controlling Dual 3-Phase Inverters for Motors (ACI and PMSM)

The idea of multiple modules controlling a single power stage can be extended to the 3-phase Inverter case. In such a case, six switching elements can be controlled using three PWM modules, one for each leg of the inverter. Each leg must switch at the same frequency and all legs must be synchronized. A master + two slaves configuration can easily address this requirement. Figure 18-66 shows how six PWM modules can control two independent 3-phase Inverters; each running a motor.

As in the cases shown in the previous sections, we have a choice of running each inverter at a different frequency (module 1 and module 4 are masters as in Figure 18-66), or both inverters can be synchronized by using one master (module 1) and five slaves. In this case, the frequency of modules 4, 5, and 6 (all equal) can be integer multiples of the frequency for modules 1, 2, 3 (also all equal).

**Figure 18-66. Control of Dual 3-Phase Inverter Stages as Is Commonly Used in Motor Control**

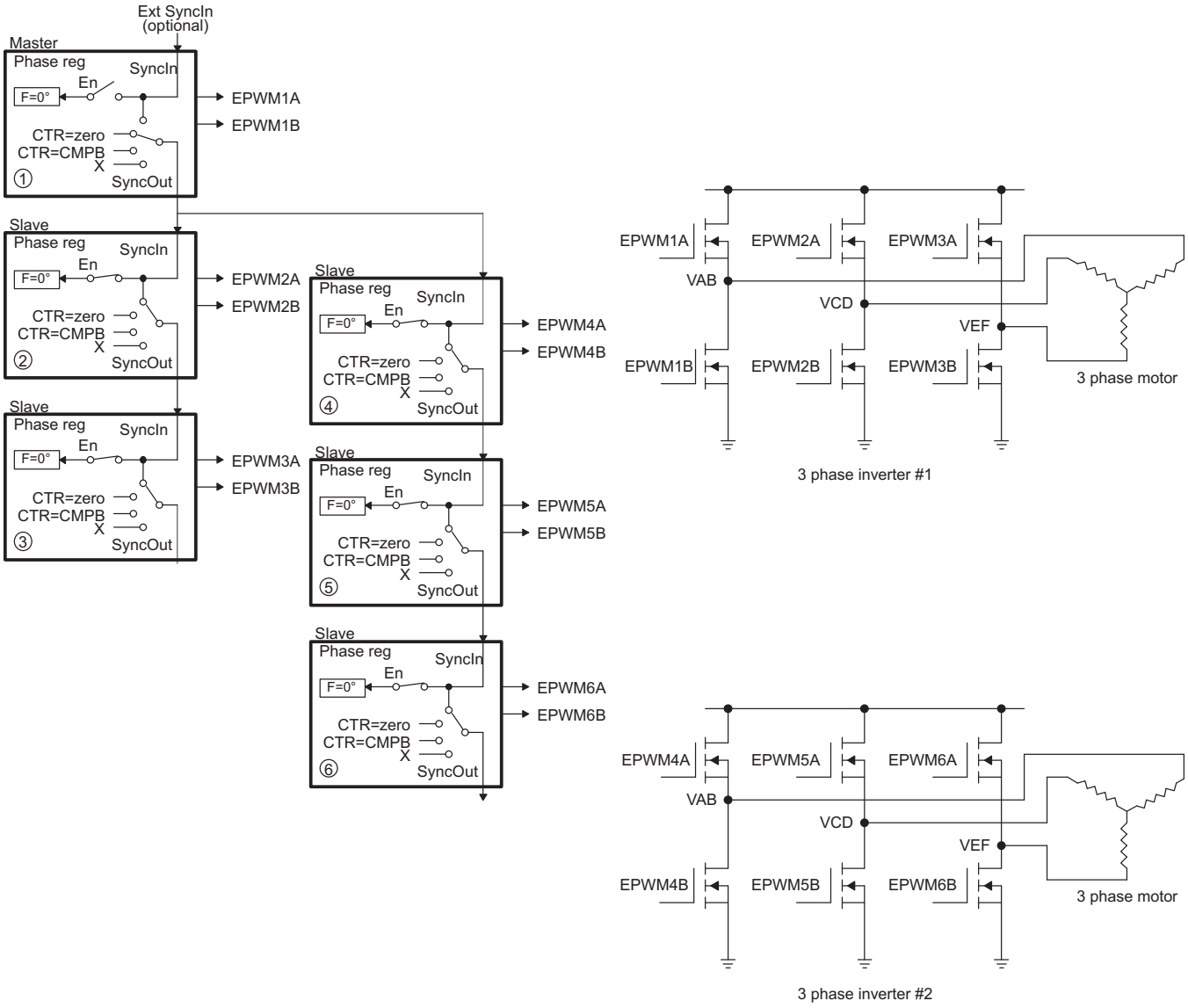
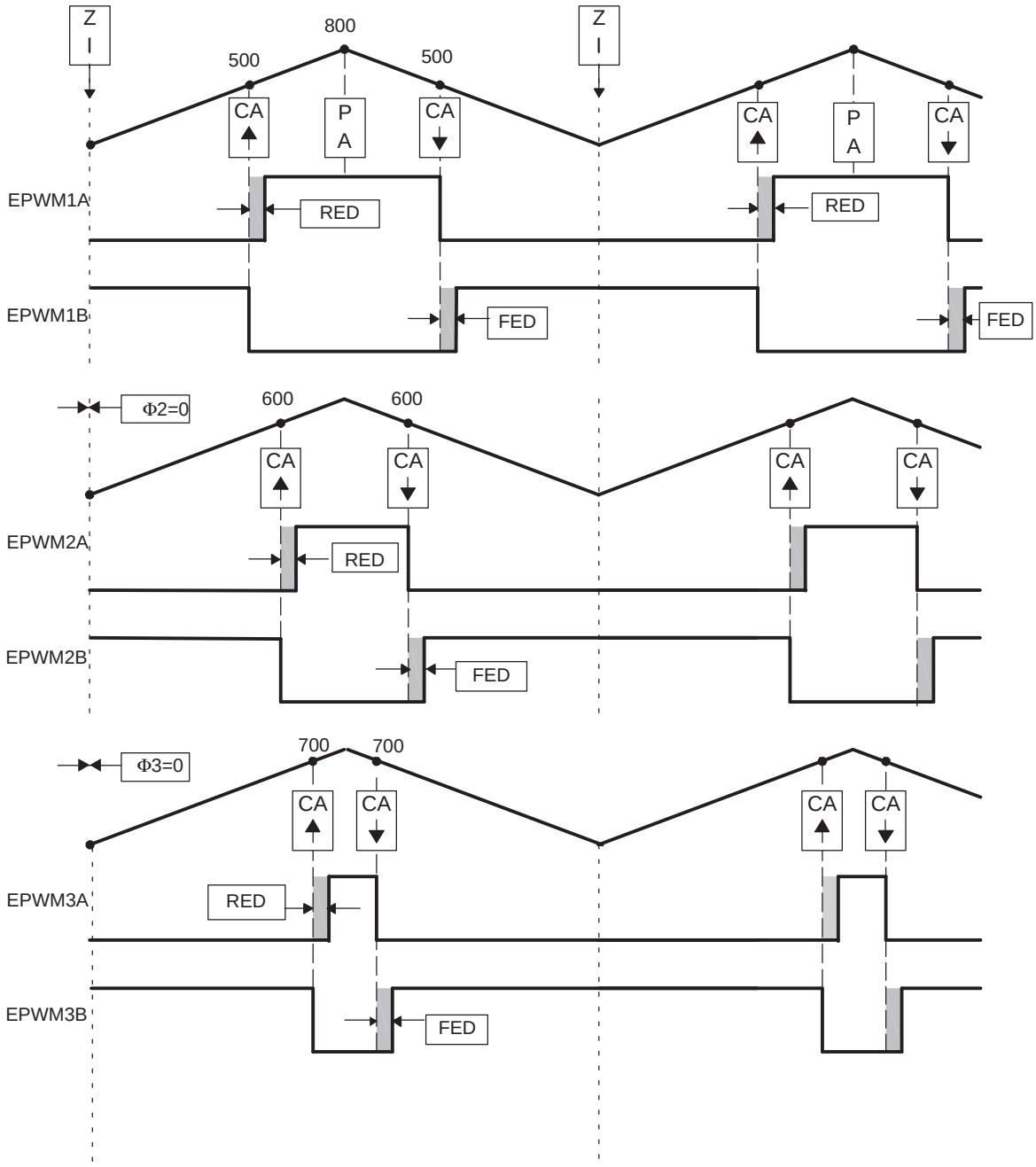


**Figure 18-67. 3-Phase Inverter Waveforms for Figure 18-66 (Only One Inverter Shown)**



### 18.4.7 Practical Applications Using Phase Control Between PWM Modules

So far, none of the examples have made use of the phase register (TBPHS). It has either been set to zero or its value has been a don't care. However, by programming appropriate values into TBPHS, multiple PWM modules can address another class of power topologies that rely on phase relationship between legs (or stages) for correct operation. As described in the TB module section, a PWM module can be configured to allow a SyncIn pulse to cause the TBPHS register to be loaded into the TBCTR register. To illustrate this concept, Figure 18-68 shows a master and slave module with a phase relationship of 120° (that is, the slave leads the master).

**Figure 18-68. Configuring Two PWM Modules for Phase Control**

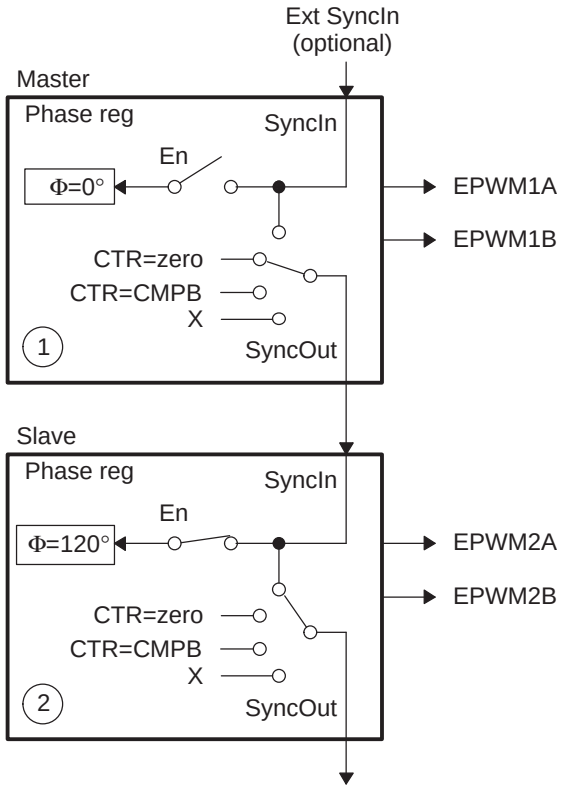
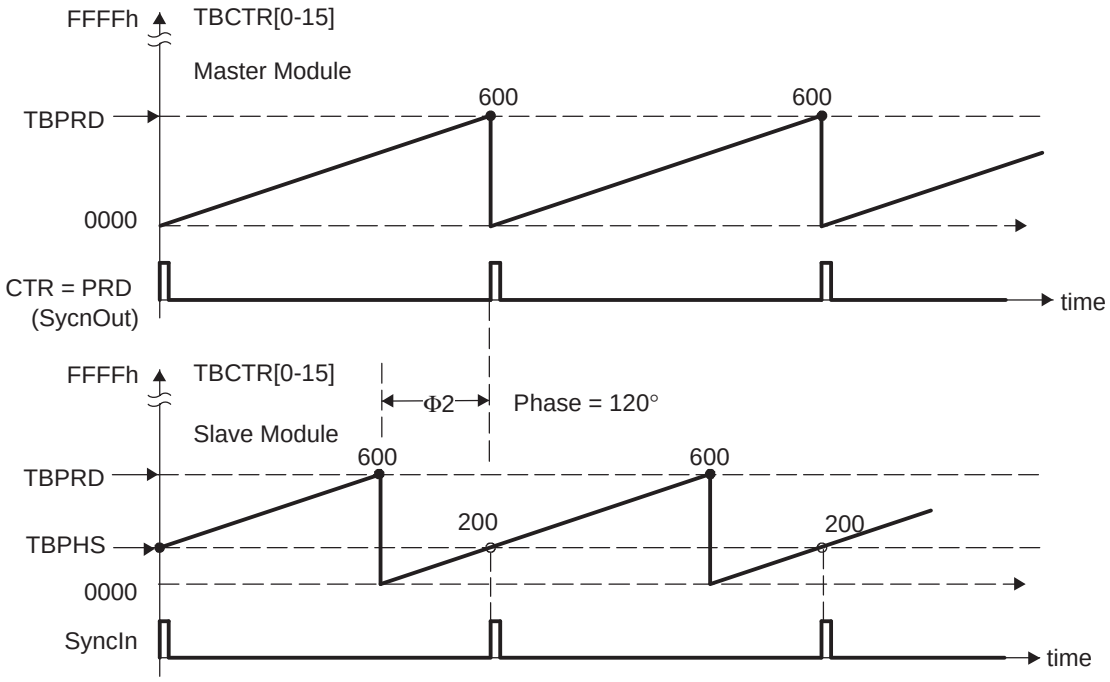



Figure 18-69 shows the associated timing waveforms for this configuration. Here, TBPRD = 600 for both master and slave. For the slave, TBPHS = 200 (that is, 200/600 X 360° = 120°). Whenever the master generates a SyncIn pulse (CTR = PRD), the value of TBPHS = 200 is loaded into the slave TBCTR register so the slave time-base is always leading the master's time-base by 120°.

**Figure 18-69. Timing Waveforms Associated With Phase Control Between Two Modules**




### 18.4.8 Controlling a 3-Phase Interleaved DC/DC Converter

A popular power topology that makes use of phase-offset between modules is shown in Figure 18-70. This system uses three PWM modules, with module 1 configured as the master. To work, the phase relationship between adjacent modules must be F = 120°. This is achieved by setting the slave TBPHS registers 2 and 3 with values of 1/3 and 2/3 of the period value, respectively. For example, if the period register is loaded with a value of 600 counts, then TBPHS (slave 2) = 200 and TBPHS (slave 3) = 400. Both slave modules are synchronized to the master 1 module.

This concept can be extended to four or more phases, by setting the TBPHS values appropriately. The following formula gives the TBPHS values for N phases:

TBPHS(N,M) = (TBPRD/N) x (—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 18-71 shows the waveforms for the configuration in Figure 18-70.

**Figure 18-70. Control of a 3-Phase Interleaved DC/DC Converter**

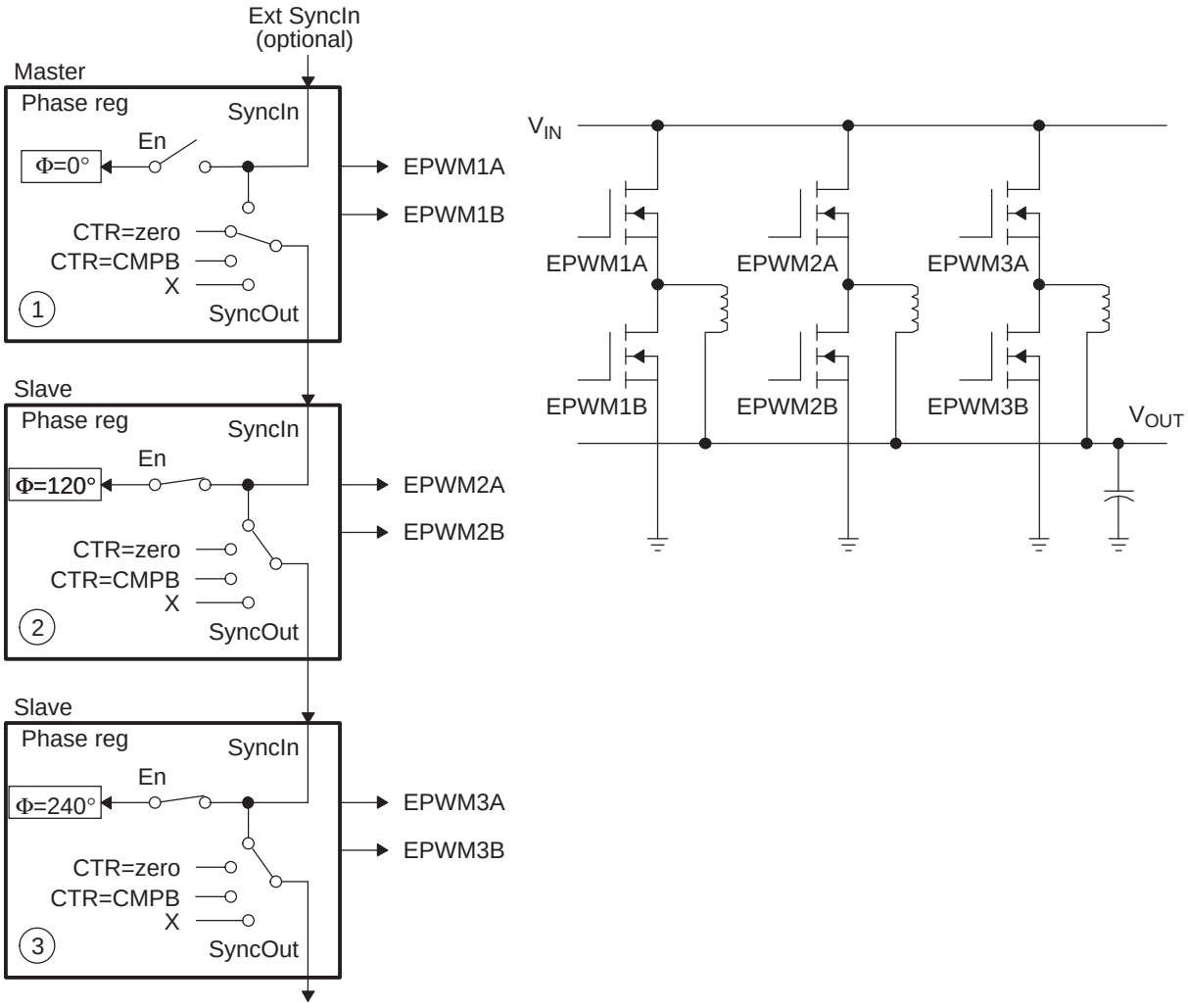
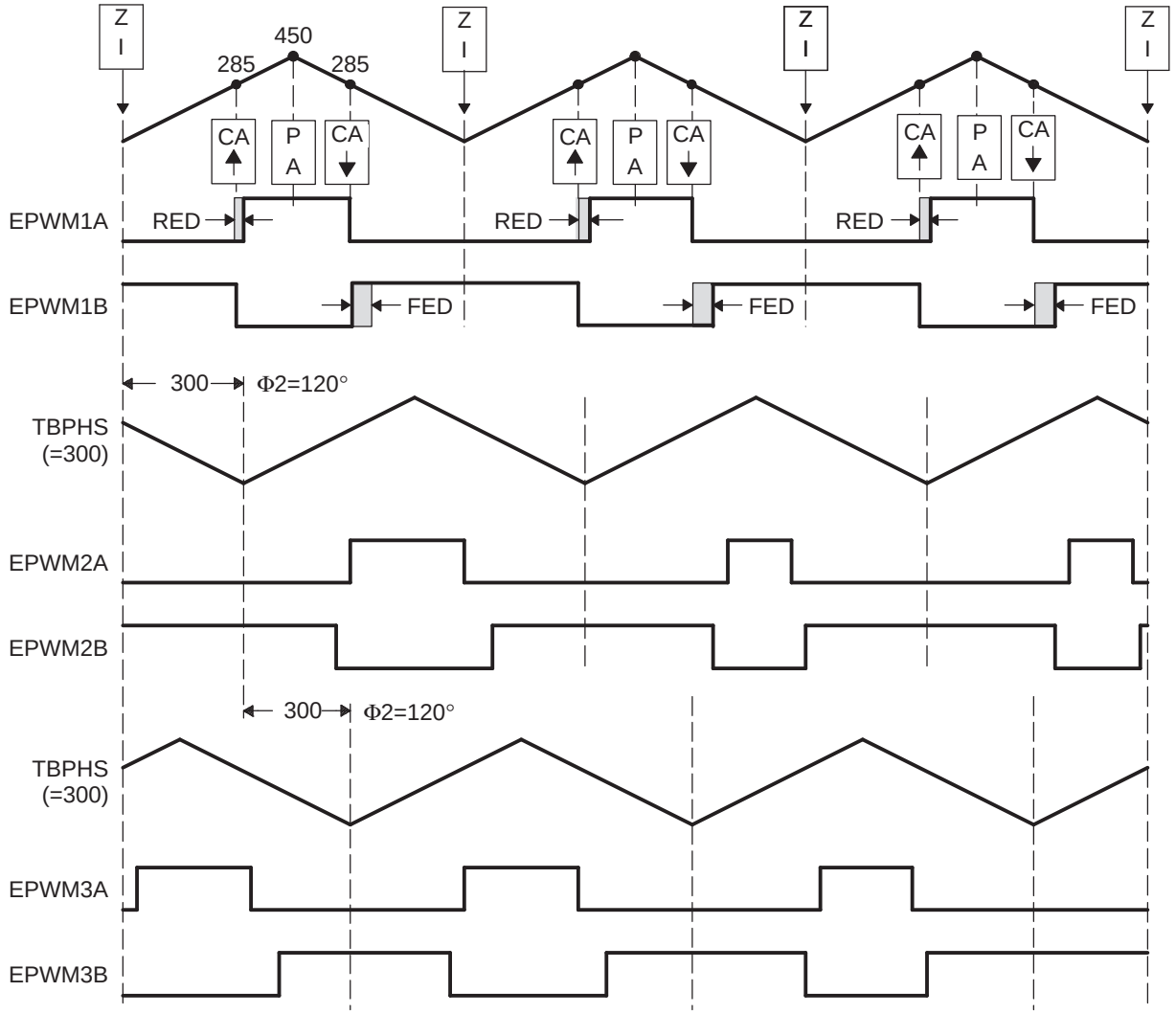


**Figure 18-71. 3-Phase Interleaved DC/DC Converter Waveforms for Figure 18-70**



### 18.4.9 Controlling Zero Voltage Switched Full Bridge (ZVSFB) Converter

The example given in Figure 18-72 assumes a static or constant phase relationship between legs (modules). In such a case, control is achieved by modulating the duty cycle. It is also possible to dynamically change the phase value on a cycle-by-cycle basis. This feature lends itself to controlling a class of power topologies known as *phase-shifted full bridge*, or *zero voltage switched full bridge.* Here the controlled parameter is not duty cycle (this is kept constant at approximately 50 percent); instead it is the phase relationship between legs. Such a system can be implemented by allocating the resources of two PWM modules to control a single power stage, which in turn requires control of four switching elements. Figure 18-73 shows a master/slave module combination synchronized together to control a full H-bridge. In this case, both master and slave modules are required to switch at the same PWM frequency. The phase is controlled by using the slave's phase register (TBPHS). The master's phase register is not used and therefore can be initialized to zero.

**Figure 18-72. Controlling a Full-H Bridge Stage ($F_{PWM2}$ = $F_{PWM1)}$**

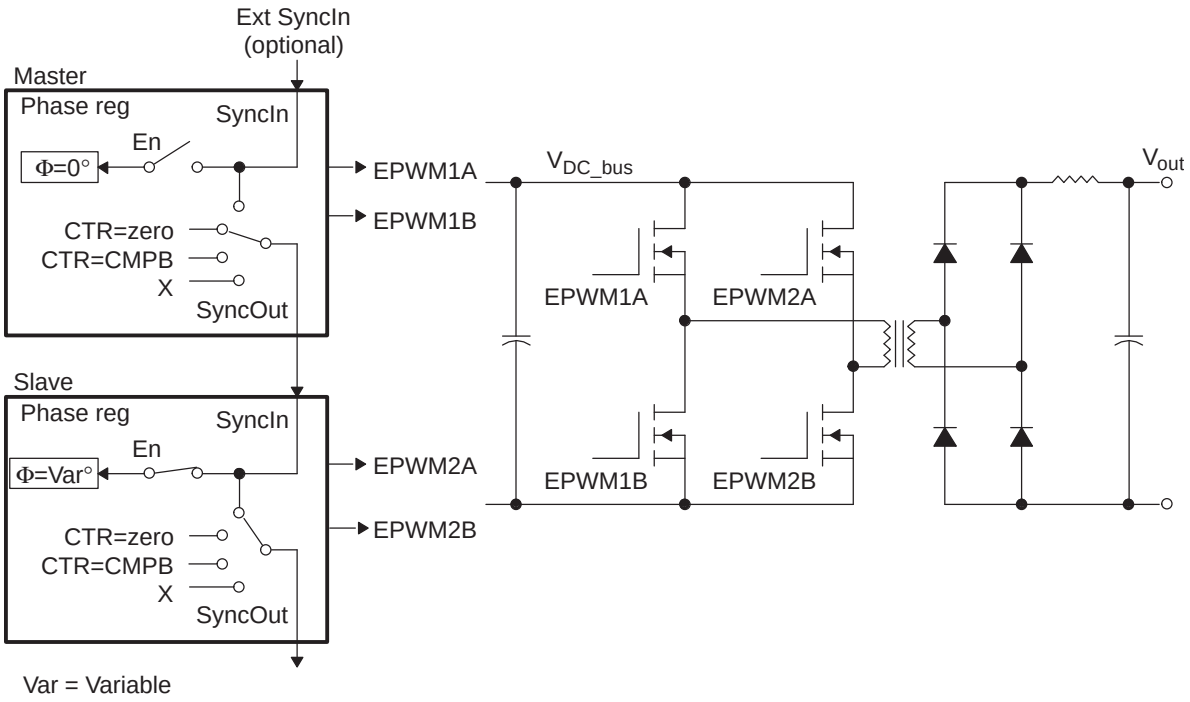
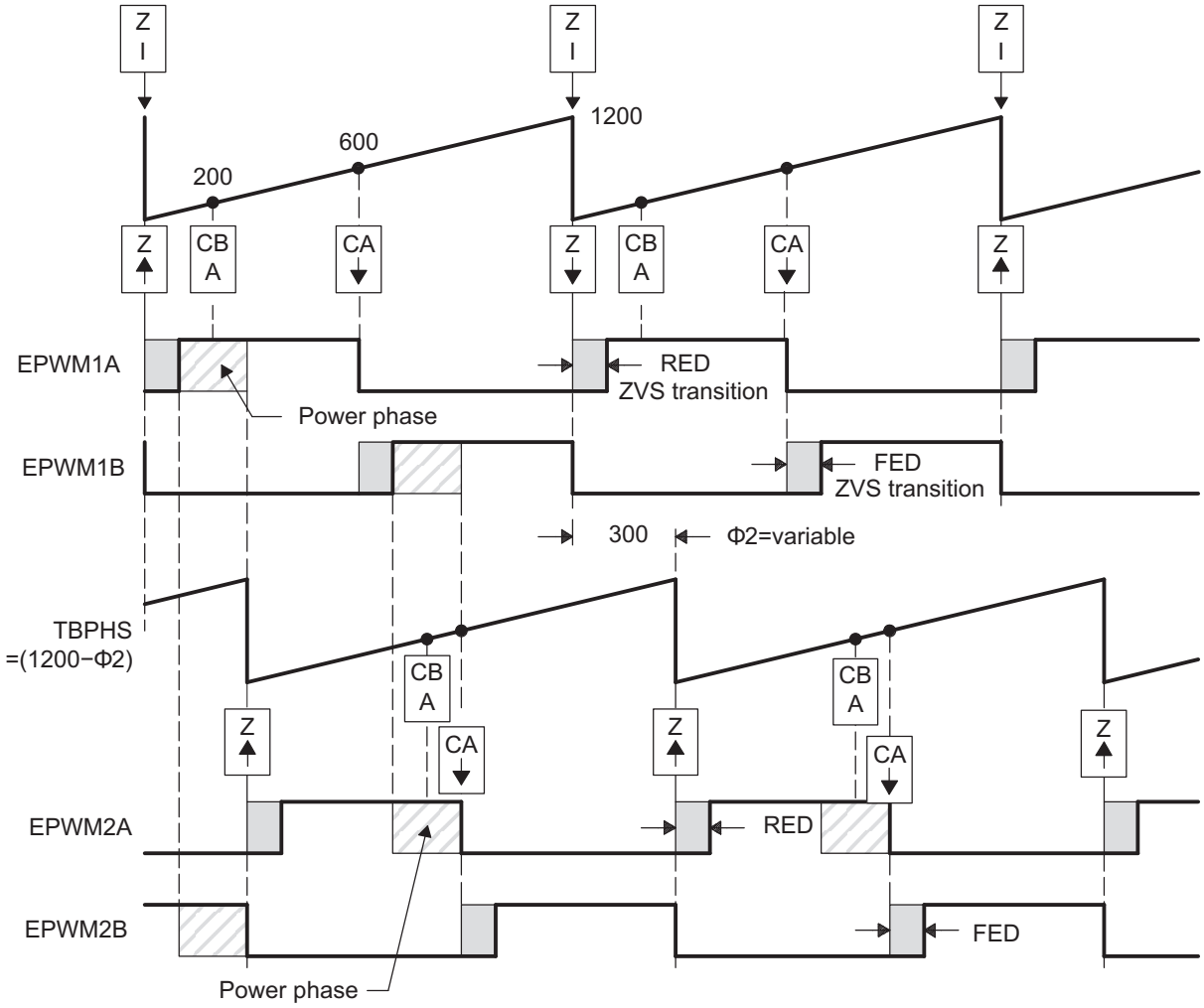


**Figure 18-73. ZVS Full-H Bridge Waveforms**




### 18.4.10 Controlling a Peak Current Mode Controlled Buck Module

Peak current control techniques offer a number of benefits like automatic over current limiting, fast correction for input voltage variations and reducing magnetic saturation. Figure 18-74 shows the use of ePWM1A along with the on-chip analog comparator for buck converter topology. The output current is sensed through a current sense resistor and fed to the positive terminal of the on-chip comparator. The internal programmable 10-bit DAC can be used to provide a reference peak current at the negative terminal of the comparator. Alternatively, an external reference could be connected at this input. The comparator output is an input to the Digital compare sub-module. The ePWM module is configured in such a way so as to trip the ePWM1A output as soon as the sensed current reaches the peak reference value. A cycle-by-cycle trip mechanism is used. Figure 18-75 shows the waveforms generated by the configuration.

**Figure 18-74. Peak Current Mode Control of a Buck Converter**

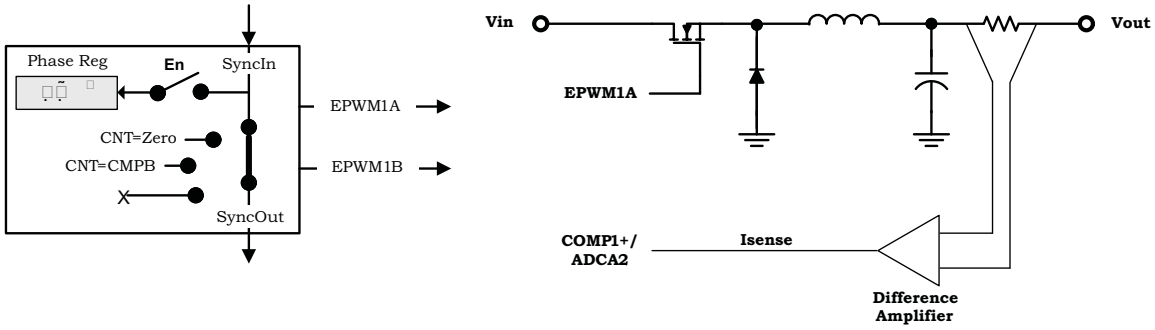
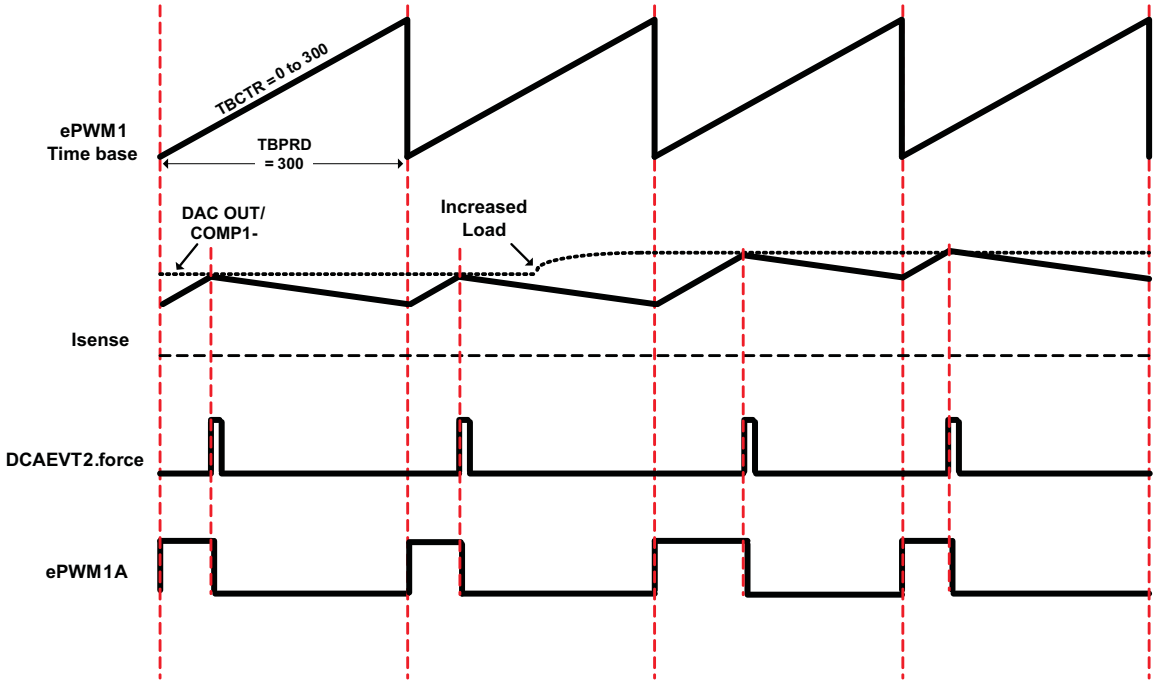


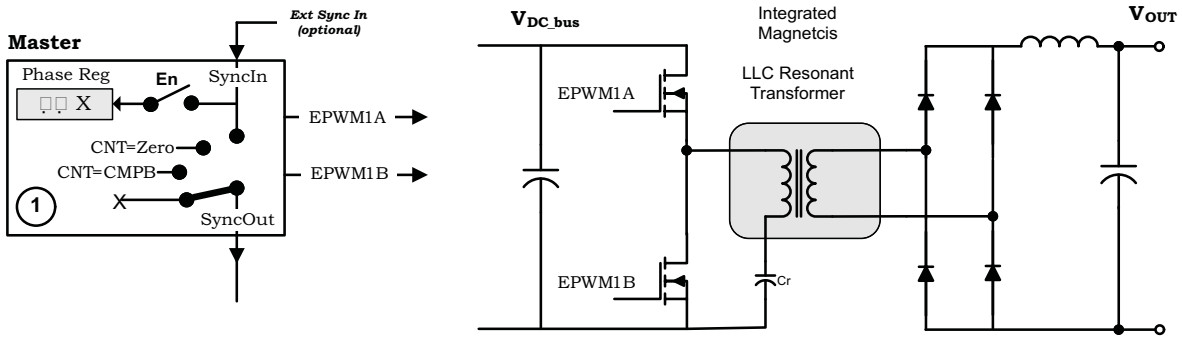

**Figure 18-75. Peak Current Mode Control Waveforms for Figure 18-74**




### 18.4.11 Controlling H-Bridge LLC Resonant Converter

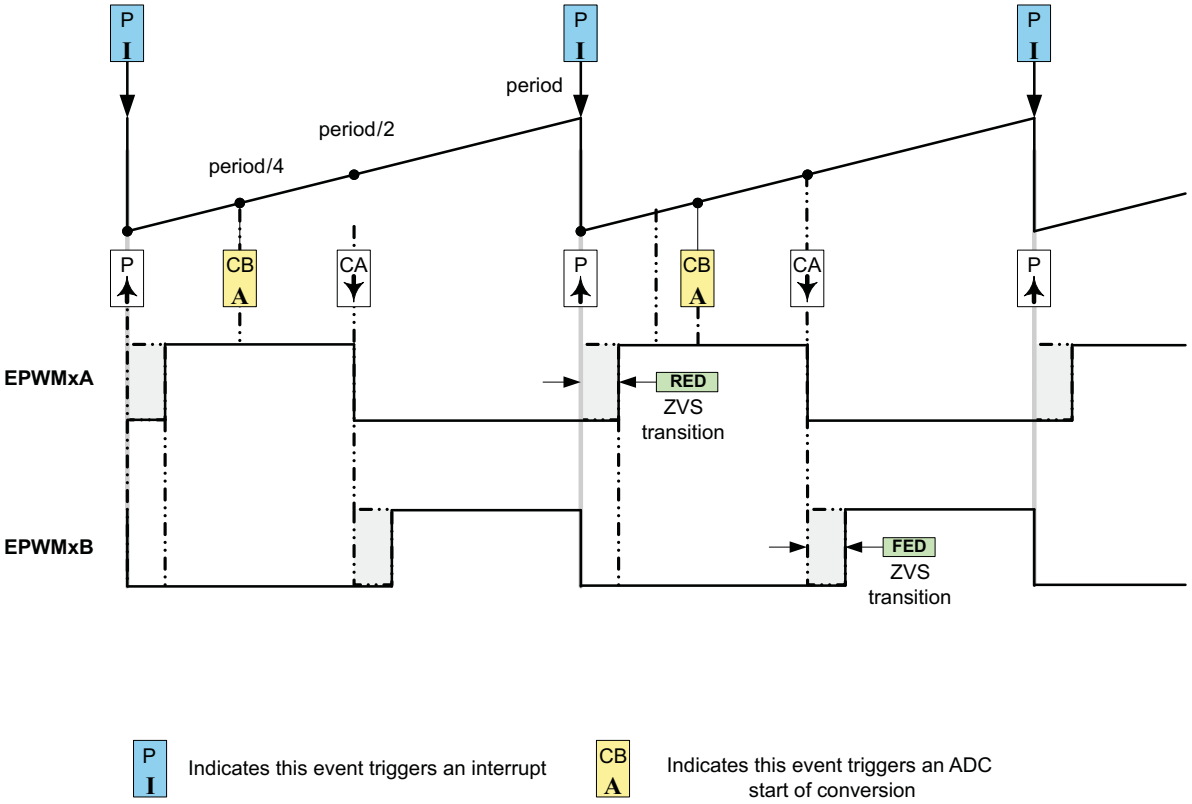
Various topologies of resonant converters are well-known in the field of power electronics for many years. In addition to these, H-bridge LLC resonant converter topology has recently gained popularity in many consumer electronics applications where high efficiency and power density are required. In this example single channel configuration of ePWM1 is detailed, yet the configuration can easily be extended to multi channel. Here the controlled parameter is not duty cycle (this is kept constant at approximately 50 percent); instead it is frequency. Although the deadband is not controlled and kept constant as 300ns (that is, 30 @100MHz TBCLK), it is up to user to update it in real time to enhance the efficiency by adjusting enough time delay for soft switching.

**Figure 18-76. Control of Two Resonant Converter Stages**




NOTE: Θ = X indicates value in phase register is a "don't care"

**Figure 18-77. H-Bridge LLC Resonant Converter PWM Waveforms**




## 18.5 Register Lock Protection

The register lock protection mechanism is added to protect the critical ePWM registers from being corrupted by accidental writes in case of runaway code. The register EPWMLOCK contains the definition of Lock bits(Table 18-14 shows the lock bits and the corresponding registers). This register also has a KEY field; writes to this register will succeed only if the KEY field is written with a value of 0xa5a5. Refer to the register descriptions for more details.

**Table 18-14. Lock Bits and Corresponding Registers**

| Bit Field | Definition | Registers Locked |
|---|---|---|
| HRLOCK | HRPWM Register Set Lock | HRCNFG, HRPWR, HRMSTEP, HRPCTL |
| GLLOCK | Global Load Register Set Lock | GLDCTL, GLDCFG |
| TZCFGLOCK | TripZone Register Set Lock | TZSEL, TZDCSEL, TZCTL, TZCTL2, TZCTLDCA, TZCTLDCB, TZEINT |
| TZCLRLOCK | TripZone Clear Register Set Lock | TZCLR, TZCBCCLR, TZOSTCLR, TZFRC |
| DCLOCK | Digital Compare Register Set Lock | DCTRIPSEL, DCACTL, DCBCTL, DCFCTL, DCCAPCTL, DCAHTRIPSEL, DCALTRIPSEL, DCBHTRIPSEL, DCBLTRIPSEL |

**NOTE:** Due to the presence of the KEY field in the same register, only 32-bit writes will succeed if the KEY matches. The 16-bit writes to the upper or lower half of this register will be ignored.

## 18.6 Registers

### 18.6.1 Enhanced Pulse Width Modulator Base Addresses

**Table 18-15. ePWM Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| EPwm1Regs | EPWM_REGS | 0x0000_4000 | 0x0000_40FF |
| EPwm2Regs | EPWM_REGS | 0x0000_4100 | 0x0000_41FF |
| EPwm3Regs | EPWM_REGS | 0x0000_4200 | 0x0000_42FF |
| EPwm4Regs | EPWM_REGS | 0x0000_4300 | 0x0000_43FF |
| EPwm5Regs | EPWM_REGS | 0x0000_4400 | 0x0000_44FF |
| EPwm6Regs | EPWM_REGS | 0x0000_4500 | 0x0000_45FF |
| EPwm7Regs | EPWM_REGS | 0x0000_4600 | 0x0000_46FF |
| EPwm8Regs | EPWM_REGS | 0x0000_4700 | 0x0000_47FF |
| SyncSocRegs | SYNC_SOC_REGS | 0x0000_7940 | 0x0000_794F |

### 18.6.1.1 EPWM_REGS Registers

Table 18-16 lists the memory-mapped registers for the EPWM_REGS. All register offset addresses not listed in Table 18-16 should be considered as reserved locations and the register contents should not be modified.

**Table 18-16. 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 | Translator 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 |
| 3Eh | EPWMREV | EPWM Revision Register | | Go |
| 3Fh | HRPWMREV | High Resolution Revision Register | | Go |
| 40h | AQCTLA | Action Qualifier Control Register For Output A | | Go |
| 41h | AQCTLA2 | Additional Action Qualifier Control Register For Output A | | Go |
| 42h | AQCTLB | Action Qualifier Control Register For Output B | | Go |
| 43h | AQCTLB2 | Additional Action Qualifier Control Register For Output B | | Go |
| 47h | AQSFRC | Action Qualifier Software Force Register | | Go |
| 49h | AQCSFRC | Action Qualifier Continuous S/W Force Register | | Go |
| 50h | DBREDHR | Dead-Band Generator Rising Edge Delay High Resolution Mirror Register | | Go |
| 51h | DBRED | Dead-Band Generator Rising Edge Delay High Resolution Mirror Register | | Go |
| 52h | DBFEDHR | Dead-Band Generator Falling Edge Delay High Resolution Register | | Go |
| 53h | DBFED | Dead-Band Generator Falling Edge Delay Count Register | | Go |
| 60h | TBPHS | Time Base Phase High | | Go |
| 62h | TBPRDHR | Time Base Period High Resolution Register | | Go |
| 63h | TBPRD | Time Base Period Register | | Go |
| 6Ah | CMPA | Counter Compare A Register | | Go |

**Table 18-16. 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 |
| FAh | EPWMLOCK | EPWM Lock Register | | Go |
| FDh | HWVDELVAL | Hardware Valley Mode Delay Register | | Go |
| FEh | VCNTVAL | Hardware Valley Counter Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 18-17 shows the codes that are used for access types in this section.

**Table 18-17. EPWM_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| W1C | 1C<br>W | 1 to clear<br>Write |
| W=1 | W | Write |
| WOnce | 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. |

#### 18.6.1.1.1 TBCTL Register (Offset = 0h) [reset = 83h]

TBCTL is shown in Figure 18-78 and described in Table 18-18.

Return to Summary Table.

Time Base Control Register

**Figure 18-78. TBCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| FREE_SOFT | | PHSDIR | CLKDIV | | | HSPCLKDIV | |
| R/W-0h | | R/W-0h | R/W-0h | | | R/W-1h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HSPCLKDIV | SWFSYNC | SYNCOSEL | | PRDLD | PHSEN | CTRMODE | |
| R/W-1h | R=0/W=1-0h | R/W-0h | | R/W-0h | R/W-0h | R/W-3h | |

**Table 18-18. 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 |

**Table 18-18. TBCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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.<br>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/W=1 | 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 |

**Table 18-18. TBCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |

#### 18.6.1.1.2 TBCTL2 Register (Offset = 1h) [reset = 0h]

TBCTL2 is shown in Figure 18-79 and described in Table 18-19.

Return to Summary Table.

Time Base Control Register 2

**Figure 18-79. TBCTL2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| PRDLDSYNC | | SYNCOSELX | | RESERVED | | | |
| R/W-0h | | R/W-0h | | R=0-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OSHTSYNC | OSHTSYNCMODE | RESERVED | RESERVED | | | | |
| R=0/W=1-0h | R/W-0h | R-0h | R=0-0h | | | | |

**Table 18-19. 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/W=1 | 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 | 0h | Reserved |
| 4-0 | RESERVED | R=0 | 0h | Reserved |

#### 18.6.1.1.3 TBCTR Register (Offset = 4h) [reset = 0h]

TBCTR is shown in Figure 18-80 and described in Table 18-20.

Return to Summary Table.

Time Base Counter Register

**Figure 18-80. TBCTR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TBCTR | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 18-20. TBCTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | TBCTR | R/W | 0h | Time Base Counter Register<br>Reset type: SYSRSn |

#### 18.6.1.1.4 TBSTS Register (Offset = 5h) [reset = 1h]

TBSTS is shown in Figure 18-81 and described in Table 18-21.

Return to Summary Table.

Time Base Status Register

**Figure 18-81. 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 18-21. 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 |

#### 18.6.1.1.5 CMPCTL Register (Offset = 8h) [reset = 0h]

CMPCTL is shown in Figure 18-82 and described in Table 18-22.

Return to Summary Table.

Counter Compare Control Register

**Figure 18-82. CMPCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | LOADBSYNC | | LOADASYNC | | SHDWBFULL | SHDWAFULL |
| R=0-0h | | R/W-0h | | R/W-0h | | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | SHDWBMODE | RESERVED | SHDWAMODE | LOADBMODE | | LOADAMODE | |
| R=0-0h | R/W-0h | R=0-0h | R/W-0h | R/W-0h | | R/W-0h | |

**Table 18-22. 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 FIFO not full yet<br>1: Indicates the CMPB shadow FIFO 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 FIFO not full yet<br>1: Indicates the CMPA shadow FIFO is full, a CPU write will overwrite the current shadow value<br>Reset type: SYSRSn |

**Table 18-22. CMPCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7 | RESERVED | R=0 | 0h | Reserved |
| 6 | SHDWBMODE | R/W | 0h | Counter-compare B (CMPB) Register Operating Mode<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 |

#### 18.6.1.1.6 CMPCTL2 Register (Offset = 9h) [reset = 0h]

CMPCTL2 is shown in Figure 18-83 and described in Table 18-23.

Return to Summary Table.

Counter Compare Control Register 2

**Figure 18-83. CMPCTL2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | LOADDSYNC | | LOADCSYNC | | RESERVED | |
| R=0-0h | | R/W-0h | | R/W-0h | | R=0-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | SHDWDMODE | RESERVED | SHDWCMODE | LOADDMODE | | LOADCMODE | |
| R=0-0h | R/W-0h | R=0-0h | R/W-0h | R/W-0h | | R/W-0h | |

**Table 18-23. 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 |

**Table 18-23. CMPCTL2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 18.6.1.1.7 DBCTL Register (Offset = Ch) [reset = 0h]

DBCTL is shown in Figure 18-84 and described in Table 18-24.

Return to Summary Table.

Dead-Band Generator Control Register

**Figure 18-84. DBCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| HALFCYCLE | DEDB_MODE | OUTSWAP | | SHDWDBFED MODE | SHDWDBRED MODE | LOADFEDMODE | |
| R/W-0h | R/W-0h | R/W-0h | | R/W-0h | R/W-0h | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LOADREDMODE | | IN_MODE | | POLSEL | | OUT_MODE | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 18-24. 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 S7 switch and bit 12 controls the S6 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 |

**Table 18-24. DBCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 18-24. DBCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-2 | POLSEL | R/W | 0h | Polarity Select Control<br><br>Bit 3 controls the S3 switch and bit 2 controls the S2 switch. This allows you to selectively invert one of the delayed signals before it is sent out of the dead-band submodule. The following descriptions correspond to classical upper/lower switch control as found in one leg of a digital motor control inverter. These assume that DBCTL[OUT_MODE] = 1,1 and DBCTL[IN_MODE] = 0x0. Other enhanced modes are also possible, but not regarded as typical usage modes.<br><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 |
| 1-0 | OUT_MODE | R/W | 0h | Dead-Band Output Mode Control<br><br>Bit 1 controls the S1 switch and bit 0 controls the S0 switch.<br><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 |

#### 18.6.1.1.8 DBCTL2 Register (Offset = Dh) [reset = 0h]

DBCTL2 is shown in Figure 18-85 and described in Table 18-25.

Return to Summary Table.

Dead-Band Generator Control Register 2

**Figure 18-85. 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 18-25. 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 via the CPU access the Shadow 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 |

#### 18.6.1.1.9 AQCTL Register (Offset = 10h) [reset = 0h]

AQCTL is shown in Figure 18-86 and described in Table 18-26.

Return to Summary Table.

Action Qualifier Control Register

**Figure 18-86. AQCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | LDAQBSYNC | | LDAQASYNC | |
| R=0-0h | | | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | SHDWAQBMODE | RESERVED | SHDWAQAMODE | LDAQBMODE | | LDAQAMODE | |
| R=0-0h | R/W-0h | R=0-0h | R/W-0h | R/W-0h | | R/W-0h | |

**Table 18-26. 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 AQCTLR[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 AQCTLR[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 |

**Table 18-26. 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 |

#### 18.6.1.1.10 AQTSRCSEL Register (Offset = 11h) [reset = 0h]

AQTSRCSEL is shown in Figure 18-87 and described in Table 18-27.

Return to Summary Table.

Action Qualifier Trigger Event Source Select Register

**Figure 18-87. AQTSRCSEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| T2SEL | | | | T1SEL | | | |
| R/W-0h | | | | R/W-0h | | | |

**Table 18-27. 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>1000: DCEVTFILT<br>Others: 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>1000: DCEVTFILT<br>Others: Reserved<br>Reset type: SYSRSn |

#### 18.6.1.1.11 PCCTL Register (Offset = 14h) [reset = 0h]

PCCTL is shown in Figure 18-88 and described in Table 18-28.

Return to Summary Table.

PWM Chopper Control Register

**Figure 18-88. PCCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | CHPDUTY | | |
| R=0-0h | | | | | R/W-0h | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CHPFREQ | | | OSHTWTH | | | | CHPEN |
| R/W-0h | | | R/W-0h | | | | R/W-0h |

**Table 18-28. 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 |

**Table 18-28. PCCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4-1 | OSHTWTH | R/W | 0h | One-Shot Pulse Width<br><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 |
| 0 | CHPEN | R/W | 0h | PWM-Chopping Enable<br><br>0: Disable (bypass) PWM chopping function<br>1: Enable chopping function<br>Reset type: SYSRSn |

#### 18.6.1.1.12 VCAPCTL Register (Offset = 18h) [reset = 0h]

VCAPCTL is shown in Figure 18-89 and described in Table 18-29.

Return to Summary Table.

Valley Capture Control Register

**Figure 18-89. 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/W=1-0h | R/W-0h |

**Table 18-29. 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 |

**Table 18-29. VCAPCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 1 | VCAPSTART | R=0/W=1 | 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 |

#### 18.6.1.1.13 VCNTCFG Register (Offset = 19h) [reset = 0h]

VCNTCFG is shown in Figure 18-90 and described in Table 18-30.

Return to Summary Table.

Valley Counter Config Register

**Figure 18-90. 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 18-30. 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 |

**Table 18-30. VCNTCFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-0 | STARTEDGE | R/W | 0h | Counter Start Edge Selection<br><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><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 |

#### 18.6.1.1.14 HRCNFG Register (Offset = 20h) [reset = 0h]

HRCNFG is shown in Figure 18-91 and described in Table 18-31.

Return to Summary Table.

HRPWM Configuration Register

This register is only accessible on EPWM modules with HRPWM capabilities.

**Figure 18-91. HRCNFG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | RESERVED | HRLOADB | | CTLMODEB | EDGMODEB | |
| R-0h | | R=0-0h | R/W-0h | | R/W-0h | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SWAPAB | AUTOCONV | SELOUTB | HRLOAD | | CTLMODE | EDGMODE | |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | | R/W-0h | R/W-0h | |

**Table 18-31. HRCNFG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | RESERVED | R | 0h | Reserved |
| 13 | RESERVED | R=0 | 0h | Reserved |
| 12-11 | HRLOADB | R/W | 0h | Shadow Mode Bit<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 |

**Table 18-31. 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>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 |

#### 18.6.1.1.15 HRPWR Register (Offset = 21h) [reset = 0h]

HRPWR is shown in Figure 18-92 and described in Table 18-32.

Return to Summary Table.

HRPWM Power Register

This register is only accessible on EPWM modules with HRPWM capabilities.

**Figure 18-92. HRPWR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| CALPWRON | RESERVED | | | | | RESERVED | |
| R/W-0h | R=0-0h | | | | | R-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | RESERVED | RESERVED | RESERVED | RESERVED | RESERVED | |
| R-0h | | R-0h | R-0h | R-0h | R-0h | R-0h | |

**Table 18-32. HRPWR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | CALPWRON | R/W | 0h | MEP Calibration Power Bits<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 | 0h | Reserved |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 0h | Reserved |
| 3 | RESERVED | R | 0h | Reserved |
| 2 | RESERVED | R | 0h | Reserved |
| 1-0 | RESERVED | R | 0h | Reserved |

#### 18.6.1.1.16 HRMSTEP Register (Offset = 26h) [reset = 0h]

HRMSTEP is shown in Figure 18-93 and described in Table 18-33.

Return to Summary Table.

HRPWM MEP Step Register

This register is only accessible on EPWM modules with HRPWM capabilities. Only 16 bit accesses are allowed for this register. Debugger access in 32 bit mode may display incorrect values.

**Figure 18-93. HRMSTEP Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HRMSTEP | | | | | | | |
| R/W-0h | | | | | | | |

**Table 18-33. 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 |

### 18.6.1.1.17 HRCNFG2 Register (Offset = 27h) [reset = 0h]

HRCNFG2 is shown in Figure 18-94 and described in Table 18-34.

Return to Summary Table.

HRPWM Configuration 2 Register

This register is only accessible on EPWM modules with HRPWM capabilities. Only 16 bit accesses are allowed for this register. Debugger access in 32 bit mode may display incorrect values.

**Figure 18-94. HRCNFG2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | RESERVED | RESERVED | | | | | |
| R-0h | R-0h | R=0-0h | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | CTLMODEDBFED | | CTLMODEDBRED | | EDGMODEDB | |
| R=0-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 18-34. HRCNFG2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14 | RESERVED | R | 0h | Reserved |
| 13-6 | RESERVED | R=0 | 0h | Reserved |
| 5-4 | CTLMODEDBFED | R/W | 0h | Shadow Mode Bit - selection should match DBCTL[LOADFEDMODE]<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 |

#### 18.6.1.1.18 HRPCTL Register (Offset = 2Dh) [reset = 0h]

HRPCTL is shown in Figure 18-95 and described in Table 18-35.

Return to Summary Table.

High Resolution Period Control Register

This register is only accessible on EPWM modules with HRPWM capabilities.

**Figure 18-95. 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-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 18-35. HRPCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-7 | RESERVED | R=0 | 0h | Reserved |
| 6-4 | PWMSYNCSELX | R/W | 0h | PWMSYNCX Source Select Bit<br>000: PWMSYNC is defined by PWMSYNCSEL - > default condition (compatible with previous EPWM versions)<br>001: Reserved<br>010: Reserved<br>011: Reserved<br>100: PWMSYNC = CMPC_eq, Count direction Up<br>101: PWMSYNC = CMPC_eq, Count direction Down<br>110: PWMSYNC = CMPD_eq, Count direction Up<br>111: PWMSYNC = CMPD_eq, Count direction Down<br>Reset type: SYSRSn |
| 3 | RESERVED | R | 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 |

**Table 18-35. HRPCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | PWMSYNCSEL | R/W | 0h | PWMSYNC Source Select Bit: This bit selects the source for the PWMSYNC signal. The PWMSYNC signal is used by external modules (such as COMP+DAC) for syncing timing to the selected EPWM mdoule:<br>0 PWMSYNC = PRD_eq signal pulse<br>1 PWMSYNC = CNT_zero signal pulse<br>Reset type: SYSRSn |
| 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 0 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 |

#### 18.6.1.1.19 TRREM Register (Offset = 2Eh) [reset = 0h]

TRREM is shown in Figure 18-96 and described in Table 18-36.

Return to Summary Table.

Translator High Resolution Remainder Register
This register is only accessible on EPWM modules with HRPWM capabilities.

**Figure 18-96. TRREM Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | TRREM | | |
| R=0-0h | | | | | R/W-0h | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TRREM | | | | | | | |
| R/W-0h | | | | | | | |

**Table 18-36. TRREM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-11 | RESERVED | R=0 | 0h | Reserved |
| 10-0 | TRREM | R/W | 0h | Translator Remainder Bits: This 11-bit value keeps track of the remainder portion of the translator algorithm calculations.<br>Reset type: SYSRSn |

#### 18.6.1.1.20 GLDCTL Register (Offset = 34h) [reset = 0h]

GLDCTL is shown in Figure 18-97 and described in Table 18-37.

Return to Summary Table.

Global PWM Load Control Register

**Figure 18-97. GLDCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | GLDCNT | | | GLDPRD | |
| R=0-0h | | | R-0h | | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| GLDPRD | RESERVED | OSHTMODE | GLDMODE | | | | GLD |
| R/W-0h | R=0-0h | R/W-0h | R/W-0h | | | | R/W-0h |

**Table 18-37. GLDCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R=0 | 0h | Reserved |
| 12-10 | GLDCNT | R | 0h | Global Reload Strobe Counter Register<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 Reload 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 |

**Table 18-37. 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 GLDCTL[GLDFRCLD] 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 |

#### 18.6.1.1.21 GLDCFG Register (Offset = 35h) [reset = 0h]

GLDCFG is shown in Figure 18-98 and described in Table 18-38.

Return to Summary Table.

Global PWM Load Config Register

**Figure 18-98. 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 18-38. 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 reload 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 reload 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 reload 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 reload 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 reload configuration if this bit is set and GLDCTL(GLD)=1<br>Reset type: SYSRSn |

**Table 18-38. GLDCFG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 reload configuration if this bit is set and GLDCTL(GLD)=1<br>Reset type: SYSRSn |
| 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 reload 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 reload 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 reload 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 reload 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 reload configuration if this bit is set and GLDCTL(GLD)=1<br>Reset type: SYSRSn |

#### 18.6.1.1.22 EPWMXLINK Register (Offset = 38h) [reset = X]

EPWMXLINK is shown in Figure 18-99 and described in Table 18-39.

Return to Summary Table.

EPWMx Link Register

This register controls which EPWMs are linked to other EPWM modules. The default reset value will vary for each module. The reset value will link each EPWM module to itself to prevent unintentional linking of modules.

**Figure 18-99. EPWMXLINK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GLDCTL2LINK | | | | RESERVED | | | | | | | | CMPDLINK | | | |
| R/W-X | | | | R=0-0h | | | | | | | | R/W-X | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CMPCLINK | | | | CMPBLINK | | | | CMPALINK | | | | TBPRDLINK | | | |
| R/W-X | | | | R/W-X | | | | R/W-X | | | | R/W-X | | | |

**Table 18-39. EPWMXLINK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | GLDCTL2LINK | R/W | X | 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>0010: ePWM3<br>0011: ePWM4<br>0100: ePWM5<br>0101: ePWM6<br>0110: ePWM7<br>0111: ePWM8<br>1000: ePWM9<br>1001: ePWM10<br>1010: ePWM11<br>1011: ePWM12<br>1100: Reserved<br>...<br>1111: Reserved<br>Reset type: SYSRSn |
| 27-20 | RESERVED | R=0 | 0h | Reserved |

**Table 18-39. EPWMXLINK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 19-16 | CMPDLINK | R/W | X | CMPD Link Bits<br><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><br>0000: ePWM1<br>0001: ePWM2<br>0010: ePWM3<br>0011: ePWM4<br>0100: ePWM5<br>0101: ePWM6<br>0110: ePWM7<br>0111: ePWM8<br>1000: ePWM9<br>1001: ePWM10<br>1010: ePWM11<br>1011: ePWM12<br>1100: Reserved<br>...<br>1111: Reserved<br>Reset type: SYSRSn |
| 15-12 | CMPCLINK | R/W | X | CMPC Link Bits<br><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><br>0000: ePWM1<br>0001: ePWM2<br>0010: ePWM3<br>0011: ePWM4<br>0100: ePWM5<br>0101: ePWM6<br>0110: ePWM7<br>0111: ePWM8<br>1000: ePWM9<br>1001: ePWM10<br>1010: ePWM11<br>1011: ePWM12<br>1100: Reserved<br>...<br>1111: Reserved<br>Reset type: SYSRSn |

**Table 18-39. EPWMXLINK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11-8 | CMPBLINK | R/W | X | CMPB_CMPBHR Link Bits<br><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><br>0000: ePWM1<br>0001: ePWM2<br>0010: ePWM3<br>0011: ePWM4<br>0100: ePWM5<br>0101: ePWM6<br>0110: ePWM7<br>0111: ePWM8<br>1000: ePWM9<br>1001: ePWM10<br>1010: ePWM11<br>1011: ePWM12<br>1100: Reserved<br>...<br>1111: Reserved<br>Reset type: SYSRSn |
| 7-4 | CMPALINK | R/W | X | CMPA_CMPAHR Link Bits<br><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><br>0000: ePWM1<br>0001: ePWM2<br>0010: ePWM3<br>0011: ePWM4<br>0100: ePWM5<br>0101: ePWM6<br>0110: ePWM7<br>0111: ePWM8<br>1000: ePWM9<br>1001: ePWM10<br>1010: ePWM11<br>1011: ePWM12<br>1100: Reserved<br>...<br>1111: Reserved<br>Reset type: SYSRSn |

**Table 18-39. EPWMXLINK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-0 | TBPRDLINK | R/W | X | TBPRD_TBPRDHR Link Bits<br><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><br>0000: ePWM1<br>0001: ePWM2<br>0010: ePWM3<br>0011: ePWM4<br>0100: ePWM5<br>0101: ePWM6<br>0110: ePWM7<br>0111: ePWM8<br>1000: ePWM9<br>1001: ePWM10<br>1010: ePWM11<br>1011: ePWM12<br>1100: Reserved<br>...<br>1111: Reserved<br>Reset type: SYSRSn |

#### 18.6.1.1.23 EPWMREV Register (Offset = 3Eh) [reset = X]

EPWMREV is shown in Figure 18-100 and described in Table 18-40.

Return to Summary Table.

EPWM Revision Register

**Figure 18-100. EPWMREV Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| TYPE | | | | | | | |
| R-4h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| REV | | | | | | | |
| R-X | | | | | | | |

**Table 18-40. EPWMREV Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | TYPE | R | 4h | EPWM Type Bits: These bits specify the EPWM type. These bits are changed if the functionality of the EPWM is changed or any feature is added or removed:<br>Reset type: SYSRSn |
| 7-0 | REV | R | X | EPWM Silicon Revision Bits: These bits specify the EPWM revision. These bits are changed if any bug fixes are performed:<br>Reset type: SYSRSn |

#### 18.6.1.1.24 HRPWMREV Register (Offset = 3Fh) [reset = 300h]

HRPWMREV is shown in Figure 18-101 and described in Table 18-41.

Return to Summary Table.

High Resolution Revision Register

**Figure 18-101. HRPWMREV Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| TYPE | | | | | | | |
| R-3h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| REV | | | | | | | |
| R-0h | | | | | | | |

**Table 18-41. HRPWMREV Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | TYPE | R | 3h | HRPWM Type Bits: These bits specify the HRPWM type. These bits are changed if the functionality of the HRPWM is changed or any feature is added or removed:<br>Reset type: SYSRSn |
| 7-0 | REV | R | 0h | HRPWM Silicon Revision Bits: These bits specify the HRPWM revision. These bits are changed if any bug fixes are performed:<br>Reset type: SYSRSn |

#### 18.6.1.1.25 AQCTLA Register (Offset = 40h) [reset = 0h]

AQCTLA is shown in Figure 18-102 and described in Table 18-42.

Return to Summary Table.

Action Qualifier Control Register For Output A

**Figure 18-102. AQCTLA Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | CBD | | CBU | |
| R=0-0h | | | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CAD | | CAU | | PRD | | ZRO | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 18-42. 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 |

**Table 18-42. AQCTLA Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 18.6.1.1.26 AQCTLA2 Register (Offset = 41h) [reset = 0h]

AQCTLA2 is shown in Figure 18-103 and described in Table 18-43.

Return to Summary Table.

Additional Action Qualifier Control Register For Output A

**Figure 18-103. AQCTLA2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| T2D | | T2U | | T1D | | T1U | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 18-43. 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 |

**Table 18-43. AQCTLA2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | T1U | R/W | 0h | Action when event occurs on T1 in UP-Count<br><br>Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.<br><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 |

#### 18.6.1.1.27 AQCTLB Register (Offset = 42h) [reset = 0h]

AQCTLB is shown in Figure 18-104 and described in Table 18-44.

Return to Summary Table.

Action Qualifier Control Register For Output B

**Figure 18-104. AQCTLB Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | CBD | | CBU | |
| R=0-0h | | | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CAD | | CAU | | PRD | | ZRO | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 18-44. 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 |

**Table 18-44. AQCTLB Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 18.6.1.1.28 AQCTLB2 Register (Offset = 43h) [reset = 0h]

AQCTLB2 is shown in Figure 18-105 and described in Table 18-45.

Return to Summary Table.

Additional Action Qualifier Control Register For Output B

**Figure 18-105. AQCTLB2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| T2D | | T2U | | T1D | | T1U | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 18-45. 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 |

**Table 18-45. AQCTLB2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |

#### 18.6.1.1.29 AQSFRC Register (Offset = 47h) [reset = 0h]

AQSFRC is shown in Figure 18-106 and described in Table 18-46.

Return to Summary Table.

Action Qualifier Software Force Register

**Figure 18-106. 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/W=1-0h | R/W-0h | | R=0/W=1-0h | R/W-0h | |

**Table 18-46. 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 event counter equals zero<br>01: Load on event counter equals period<br>10: Load on event 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/W=1 | 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/W=1 | 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 |

**Table 18-46. AQSFRC Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | ACTSFA | R/W | 0h | Action When One-Time Software Force A Is Invoked<br><br>00: Does nothing (action disabled)<br>01: Clear (low)<br>10: Set (high)<br>11: Toggle (Low -> High, High -> Low)<br><br>Note: This action is not qualified by counter direction (CNT_dir).<br>Reset type: SYSRSn |

#### 18.6.1.1.30 AQCSFRC Register (Offset = 49h) [reset = 0h]

AQCSFRC is shown in Figure 18-107 and described in Table 18-47.

Return to Summary Table.

Action Qualifier Continuous S/W Force Register

**Figure 18-107. AQCSFRC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | CSFB | | CSFA | |
| R=0-0h | | | | R/W-0h | | R/W-0h | |

**Table 18-47. 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 |

#### 18.6.1.1.31 DBREDHR Register (Offset = 50h) [reset = 0h]

DBREDHR is shown in Figure 18-108 and described in Table 18-48.

Return to Summary Table.

Dead-Band Generator Rising Edge Delay High Resolution Mirror Register

**Figure 18-108. 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 18-48. 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 |

#### 18.6.1.1.32 DBRED Register (Offset = 51h) [reset = 0h]

DBRED is shown in Figure 18-109 and described in Table 18-49.

Return to Summary Table.

Dead-Band Generator Rising Edge Delay High Resolution Mirror Register

**Figure 18-109. DBRED Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | DBRED | | | | | |
| R-0h | | R/W-0h | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DBRED | | | | | | | |
| R/W-0h | | | | | | | |

**Table 18-49. 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 |

#### 18.6.1.1.33 DBFEDHR Register (Offset = 52h) [reset = 0h]

DBFEDHR is shown in Figure 18-110 and described in Table 18-50.

Return to Summary Table.

Dead-Band Generator Falling Edge Delay High Resolution Register

**Figure 18-110. 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 18-50. 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 |

#### 18.6.1.1.34 DBFED Register (Offset = 53h) [reset = 0h]

DBFED is shown in Figure 18-111 and described in Table 18-51.

Return to Summary Table.

Dead-Band Generator Falling Edge Delay Count Register

**Figure 18-111. DBFED Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | DBFED | | | | | |
| R-0h | | R/W-0h | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DBFED | | | | | | | |
| R/W-0h | | | | | | | |

**Table 18-51. 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 |

#### 18.6.1.1.35 TBPHS Register (Offset = 60h) [reset = 0h]

TBPHS is shown in Figure 18-112 and described in Table 18-52.

Return to Summary Table.

Time Base Phase High

**Figure 18-112. TBPHS Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TBPHS | | | | | | | | | | | | | | | | TBPHSHR | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | R/W-0h | | | | | | | | | | | | | | | |

**Table 18-52. TBPHS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | TBPHS | R/W | 0h | Phase Offset Register<br><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><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>Reset type: SYSRSn |

#### 18.6.1.1.36 TBPRDHR Register (Offset = 62h) [reset = 0h]

TBPRDHR is shown in Figure 18-113 and described in Table 18-53.

Return to Summary Table.

Time Base Period High Resolution Register

**Figure 18-113. TBPRDHR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TBPRDHR | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 18-53. TBPRDHR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | TBPRDHR | R/W | 0h | Period High Resolution Bits<br><br>These 8-bits contain the high-resolution portion of the period value. The TBPRDHR register is not affected by the TBCTL[PRDLD] bit. Reads from this register always reflect the shadow register. Likewise writes are also to the shadow register. The TBPRDHR register is only used when the high resolution period feature is enabled. This register is only available with ePWM modules which support high-resolution period control.<br>Reset type: SYSRSn |

#### 18.6.1.1.37 TBPRD Register (Offset = 63h) [reset = 0h]

TBPRD is shown in Figure 18-114 and described in Table 18-54.

Return to Summary Table.

Time Base Period Register

**Figure 18-114. TBPRD Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TBPRD | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 18-54. TBPRD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | TBPRD | R/W | 0h | Time Base Period Register<br><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><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><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><br>- The active and shadow registers share the same memory map address.<br><br>Reset type: SYSRSn |

#### 18.6.1.1.38 CMPA Register (Offset = 6Ah) [reset = 0h]

CMPA is shown in Figure 18-115 and described in Table 18-55.

Return to Summary Table.

Counter Compare A Register

**Figure 18-115. CMPA Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CMPA | | | | | | | | | | | | | | | | CMPAHR | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | R/W-0h | | | | | | | | | | | | | | | |

**Table 18-55. CMPA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | CMPA | R/W | 0h | Compare A Register<br><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><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><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><br>These 8-bits contain the high-resolution portion (least significant 8-bits) of the counter-compare A value. CMPA:CMPAHR can be accessed in a single 32-bit read/write. Shadowing is enabled and disabled by the CMPCTL[SHDWAMODE] bit as described for the CMPA register.<br>Reset type: SYSRSn |

#### 18.6.1.1.39 CMPB Register (Offset = 6Ch) [reset = 0h]

CMPB is shown in Figure 18-116 and described in Table 18-56.

Return to Summary Table.

Compare B Register

**Figure 18-116. CMPB Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CMPB | | | | | | | | | | | | | | | | CMPBHR | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | R/W-0h | | | | | | | | | | | | | | | |

**Table 18-56. CMPB Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | CMPB | R/W | 0h | Compare B Register<br><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><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><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>Reset type: SYSRSn |

#### 18.6.1.1.40 CMPC Register (Offset = 6Fh) [reset = 0h]

CMPC is shown in Figure 18-117 and described in Table 18-57.

Return to Summary Table.

Counter Compare C Register
LINK feature access should always be 16-bit

**Figure 18-117. CMPC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CMPC | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 18-57. 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<br>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 |

#### 18.6.1.1.41 CMPD Register (Offset = 71h) [reset = 0h]

CMPD is shown in Figure 18-118 and described in Table 18-58.

Return to Summary Table.

Counter Compare D Register
LINK feature access should always be 16-bit

**Figure 18-118. CMPD Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CMPD | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 18-58. CMPD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | CMPD | R/W | 0h | Compare D Register<br><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><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<br>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 |

#### 18.6.1.1.42 GLDCTL2 Register (Offset = 74h) [reset = 0h]

GLDCTL2 is shown in Figure 18-119 and described in Table 18-59.

Return to Summary Table.

Global PWM Load Control Register 2

**Figure 18-119. GLDCTL2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | GFRCLD | OSHTLD |
| R=0-0h | | | | | | R=0/W=1-0h | R=0/W=1-0h |

**Table 18-59. GLDCTL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-2 | RESERVED | R=0 | 0h | Reserved |
| 1 | GFRCLD | R=0/W=1 | 0h | Force Load Event in One Shot Mode<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 as shown in the diagram below. This bit is intended to be used for testing and/or software force loading of the events in global load mode.<br>Reset type: SYSRSn |
| 0 | OSHTLD | R=0/W=1 | 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 |

#### 18.6.1.1.43 SWVDELVAL Register (Offset = 77h) [reset = 0h]

SWVDELVAL is shown in Figure 18-120 and described in Table 18-60.

Return to Summary Table.

Software Valley Mode Delay Register

**Figure 18-120. SWVDELVAL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SWVDELVAL | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 18-60. SWVDELVAL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | SWVDELVAL | R | 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 |

#### 18.6.1.1.44 TZSEL Register (Offset = 80h) [reset = 0h]

TZSEL is shown in Figure 18-121 and described in Table 18-61.

Return to Summary Table.

Trip Zone Select Register

**Figure 18-121. TZSEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| DCBEVT1 | DCAEVT1 | OSHT6 | OSHT5 | OSHT4 | OSHT3 | OSHT2 | OSHT1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DCBEVT2 | DCAEVT2 | CBC6 | CBC5 | CBC4 | CBC3 | CBC2 | CBC1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 18-61. 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 |

**Table 18-61. TZSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 18.6.1.1.45 TZDCSEL Register (Offset = 82h) [reset = 0h]

TZDCSEL is shown in Figure 18-122 and described in Table 18-62.

Return to Summary Table.

Trip Zone Digital Comparator Select Register

**Figure 18-122. TZDCSEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | DCBEVT2 | | | DCBEVT1 |
| R=0-0h | | | | R/W-0h | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DCBEVT1 | | DCAEVT2 | | | DCAEVT1 | | |
| R/W-0h | | R/W-0h | | | R/W-0h | | |

**Table 18-62. 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 |

**Table 18-62. TZDCSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |

#### 18.6.1.1.46 TZCTL Register (Offset = 84h) [reset = 0h]

TZCTL is shown in Figure 18-123 and described in Table 18-63.

Return to Summary Table.

Trip Zone Control Register

**Figure 18-123. TZCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | DCBEVT2 | | DCBEVT1 | |
| R=0-0h | | | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DCAEVT2 | | DCAEVT1 | | TZB | | TZA | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |

**Table 18-63. 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 |

**Table 18-63. 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><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><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 |

#### 18.6.1.1.47 TZCTL2 Register (Offset = 85h) [reset = 0h]

TZCTL2 is shown in Figure 18-124 and described in Table 18-64.

Return to Summary Table.

Additional Trip Zone Control Register

**Figure 18-124. TZCTL2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| ETZE | RESERVED | | | TZBD | | | TZBU |
| R/W-0h | R=0-0h | | | R/W-0h | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TZBU | | TZAD | | | TZAU | | |
| R/W-0h | | R/W-0h | | | R/W-0h | | |

**Table 18-64. 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, DCAEVT1/2, DCBEVT1/2 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 |

**Table 18-64. TZCTL2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5-3 | TZAD | R/W | 0h | TZ1 to TZ6, DCAEVT1/2, DCBEVT1/2 Trip Action On EPWMxA while Count direction is DOWN<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 |
| 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 |

#### 18.6.1.1.48 TZCTLDCA Register (Offset = 86h) [reset = 0h]

TZCTLDCA is shown in Figure 18-125 and described in Table 18-65.

Return to Summary Table.

Trip Zone Control Register Digital Compare A

**Figure 18-125. TZCTLDCA Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | DCAEVT2D | | | DCAEVT2U |
| R=0-0h | | | | R/W-0h | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DCAEVT2U | | DCAEVT1D | | | DCAEVT1U | | |
| R/W-0h | | R/W-0h | | | R/W-0h | | |

**Table 18-65. 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 |

**Table 18-65. 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 |

### 18.6.1.1.49 TZCTLDCB Register (Offset = 87h) [reset = 0h]

TZCTLDCB is shown in Figure 18-126 and described in Table 18-66.

Return to Summary Table.

Trip Zone Control Register Digital Compare B

**Figure 18-126. TZCTLDCB Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | DCBEVT2D | | | DCBEVT2U |
| R=0-0h | | | | R/W-0h | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DCBEVT2U | | DCBEVT1D | | | DCBEVT1U | | |
| R/W-0h | | R/W-0h | | | R/W-0h | | |

**Table 18-66. TZCTLDCB Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R=0 | 0h | Reserved |
| 11-9 | DCBEVT2D | R/W | 0h | Digital Compare Output A Event 2 Action On EPWMxB while Count direction is DOWN<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 A 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 A 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 |

**Table 18-66. TZCTLDCB Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2-0 | DCBEVT1U | R/W | 0h | Digital Compare Output A Event 1 Action On EPWMxB while Count direction is UP<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 |

#### 18.6.1.1.50 TZEINT Register (Offset = 8Dh) [reset = 0h]

TZEINT is shown in Figure 18-127 and described in Table 18-67.

Return to Summary Table.

Trip Zone Enable Interrupt Register

**Figure 18-127. TZEINT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | DCBEVT2 | DCBEVT1 | DCAEVT2 | DCAEVT1 | OST | CBC | RESERVED |
| R=0-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R=0-0h |

**Table 18-67. 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 |

#### 18.6.1.1.51 TZFLG Register (Offset = 93h) [reset = 0h]

TZFLG is shown in Figure 18-128 and described in Table 18-68.

Return to Summary Table.

Trip Zone Flag Register

**Figure 18-128. TZFLG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | DCBEVT2 | DCBEVT1 | DCAEVT2 | DCAEVT1 | OST | CBC | INT |
| R=0-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 18-68. 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 |

**Table 18-68. 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 |

#### 18.6.1.1.52 TZCBCFLG Register (Offset = 94h) [reset = 0h]

TZCBCFLG is shown in Figure 18-129 and described in Table 18-69.

Return to Summary Table.

Trip Zone CBC Flag Register

**Figure 18-129. TZCBCFLG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DCBEVT2 | DCAEVT2 | CBC6 | CBC5 | CBC4 | CBC3 | CBC2 | CBC1 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 18-69. 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 |

**Table 18-69. TZCBCFLG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 18.6.1.1.53 TZOSTFLG Register (Offset = 95h) [reset = 0h]

TZOSTFLG is shown in Figure 18-130 and described in Table 18-70.

Return to Summary Table.

Trip Zone OST Flag Register

**Figure 18-130. TZOSTFLG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DCBEVT1 | DCAEVT1 | OST6 | OST5 | OST4 | OST3 | OST2 | OST1 |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 18-70. 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 |

**Table 18-70. TZOSTFLG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 18.6.1.1.54 TZCLR Register (Offset = 97h) [reset = 0h]

TZCLR is shown in Figure 18-131 and described in Table 18-71.

Return to Summary Table.

Trip Zone Clear Register

**Figure 18-131. TZCLR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| CBCPULSE | | RESERVED | | | | | |
| R/W-0h | | R=0-0h | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | DCBEVT2 | DCBEVT1 | DCAEVT2 | DCAEVT1 | OST | CBC | INT |
| R=0-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 18-71. 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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 |

**Table 18-71. TZCLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | INT | R=0/W=1 | 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 |

#### 18.6.1.1.55 TZCBCCLR Register (Offset = 98h) [reset = 0h]

TZCBCCLR is shown in Figure 18-132 and described in Table 18-72.

Return to Summary Table.

Trip Zone CBC Clear Register

**Figure 18-132. TZCBCCLR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DCBEVT2 | DCAEVT2 | CBC6 | CBC5 | CBC4 | CBC3 | CBC2 | CBC1 |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 18-72. TZCBCCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R=0 | 0h | Reserved |
| 7 | DCBEVT2 | R=0/W=1 | 0h | Clear Flag for Digital Compare Output B Event 2 selected for CBC<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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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 |

**Table 18-72. TZCBCCLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | CBC1 | R=0/W=1 | 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 |

#### 18.6.1.1.56 TZOSTCLR Register (Offset = 99h) [reset = 0h]

TZOSTCLR is shown in Figure 18-133 and described in Table 18-73.

Return to Summary Table.

Trip Zone OST Clear Register

**Figure 18-133. TZOSTCLR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DCBEVT1 | DCAEVT1 | OST6 | OST5 | OST4 | OST3 | OST2 | OST1 |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 18-73. TZOSTCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R=0 | 0h | Reserved |
| 7 | DCBEVT1 | R=0/W=1 | 0h | Clear Flag for Digital Compare Output B Event 1 selected for OST<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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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 |

**Table 18-73. TZOSTCLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | OST1 | R=0/W=1 | 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 |

#### 18.6.1.1.57 TZFRC Register (Offset = 9Bh) [reset = 0h]

TZFRC is shown in Figure 18-134 and described in Table 18-74.

Return to Summary Table.

Trip Zone Force Register

**Figure 18-134. TZFRC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | DCBEVT2 | DCBEVT1 | DCAEVT2 | DCAEVT1 | OST | CBC | RESERVED |
| R=0-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0-0h |

**Table 18-74. TZFRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-7 | RESERVED | R=0 | 0h | Reserved |
| 6 | DCBEVT2 | R=0/W=1 | 0h | Force Flag for Digital Compare Output B Event 2<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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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 |

#### 18.6.1.1.58 ETSEL Register (Offset = A4h) [reset = 0h]

ETSEL is shown in Figure 18-135 and described in Table 18-75.

Return to Summary Table.

Event Trigger Selection Register

**Figure 18-135. ETSEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SOCBEN | SOCBSEL | | | SOCAEN | SOCASEL | | |
| R/W-0h | R/W-0h | | | R/W-0h | R/W-0h | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | INTSELCMP | SOCBSELCMP | SOCASELCMP | INTEN | INTSEL | | |
| R=0-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | | |

**Table 18-75. 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 |

**Table 18-75. 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 |

**Table 18-75. ETSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 18.6.1.1.59 ETPS Register (Offset = A6h) [reset = 0h]

ETPS is shown in Figure 18-136 and described in Table 18-76.

Return to Summary Table.

Event Trigger Pre-Scale Register

**Figure 18-136. ETPS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SOCBCNT | | SOCBPRD | | SOCACNT | | SOCAPRD | |
| R-0h | | R/W-0h | | R-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | SOCPSSEL | INTPSSEL | INTCNT | | INTPRD | |
| R=0-0h | | R/W-0h | R/W-0h | R-0h | | R/W-0h | |

**Table 18-76. 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 |

**Table 18-76. 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><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><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><br>0: Selects ETPS [SOCACNT/SOCBCNT] and [SOCAPRD/SOCBPRD] registers to determine frequency of events (interrupt once every 0-3 events).<br>1: Selects ETSOCPS [SOCACNT2/SOCBCNT2] and [SOCAPRD2/SOCBPRD2] registers to determine frequency of events (interrupt once every 0-15 events).<br>Reset type: SYSRSn |
| 4 | INTPSSEL | R/W | 0h | EPWMxINTn Pre-Scale Selection Bits<br><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><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><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 |

**Table 18-76. ETPS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | INTPRD | R/W | 0h | ePWM Interrupt (EPWMx_INT) Period Select<br><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><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><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><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 |

#### 18.6.1.1.60 ETFLG Register (Offset = A8h) [reset = 0h]

ETFLG is shown in Figure 18-137 and described in Table 18-77.

Return to Summary Table.

Event Trigger Flag Register

**Figure 18-137. ETFLG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | SOCB | SOCA | RESERVED | INT |
| R=0-0h | | | | R-0h | R-0h | R=0-0h | R-0h |

**Table 18-77. 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 |

#### 18.6.1.1.61 ETCLR Register (Offset = AAh) [reset = 0h]

ETCLR is shown in Figure 18-138 and described in Table 18-78.

Return to Summary Table.

Event Trigger Clear Register

**Figure 18-138. ETCLR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | SOCB | SOCA | RESERVED | INT |
| R=0-0h | | | | R=0/W=1-0h | R=0/W=1-0h | R=0-0h | R=0/W=1-0h |

**Table 18-78. ETCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-4 | RESERVED | R=0 | 0h | Reserved |
| 3 | SOCB | R=0/W=1 | 0h | ePWM ADC Start-of-Conversion A (EPWMxSOCB) Flag Clear Bit<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/W=1 | 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/W=1 | 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 |

#### 18.6.1.1.62 ETFRC Register (Offset = ACh) [reset = 0h]

ETFRC is shown in Figure 18-139 and described in Table 18-79.

Return to Summary Table.

Event Trigger Force Register

**Figure 18-139. ETFRC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | SOCB | SOCA | RESERVED | INT |
| R=0-0h | | | | R=0/W=1-0h | R=0/W=1-0h | R=0-0h | R=0/W=1-0h |

**Table 18-79. ETFRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-4 | RESERVED | R=0 | 0h | Reserved |
| 3 | SOCB | R=0/W=1 | 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/W=1 | 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/W=1 | 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 |

#### 18.6.1.1.63 ETINTPS Register (Offset = AEh) [reset = 0h]

ETINTPS is shown in Figure 18-140 and described in Table 18-80.

Return to Summary Table.

Event-Trigger Interrupt Pre-Scale Register

**Figure 18-140. ETINTPS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTCNT2 | | | | INTPRD2 | | | |
| R-0h | | | | R/W-0h | | | |

**Table 18-80. 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 |

#### 18.6.1.1.64 ETSOCPS Register (Offset = B0h) [reset = 0h]

ETSOCPS is shown in Figure 18-141 and described in Table 18-81.

Return to Summary Table.

Event-Trigger SOC Pre-Scale Register

**Figure 18-141. ETSOCPS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SOCBCNT2 | | | | SOCBPRD2 | | | |
| R-0h | | | | R/W-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SOCACNT2 | | | | SOCAPRD2 | | | |
| R-0h | | | | R/W-0h | | | |

**Table 18-81. 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 interrupt on SOCBCNT2 = 1 (first event)<br>0010: Generate interrupt on SOCBCNT2 = 2 (second event)<br>0011: Generate interrupt on SOCBCNT2 = 3 (third event)<br>0100: Generate interrupt on SOCBCNT2 = 4 (fourth event)<br>...<br>1111: Generate interrupt 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 |

**Table 18-81. 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 interrupt on SOCACNT2 = 1 (first event)<br>0010: Generate interrupt on SOCACNT2 = 2 (second event)<br>0011: Generate interrupt on SOCACNT2 = 3 (third event)<br>0100: Generate interrupt on SOCACNT2 = 4 (fourth event)<br>...<br>1111: Generate interrupt on SOCACNT2 = 15 (fifteenth event)<br>Reset type: SYSRSn |

#### 18.6.1.1.65 ETCNTINITCTL Register (Offset = B2h) [reset = 0h]

ETCNTINITCTL is shown in Figure 18-142 and described in Table 18-82.

Return to Summary Table.

Event-Trigger Counter Initialization Control Register

**Figure 18-142. ETCNTINITCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| SOCBINITEN | SOCAINITEN | INTINITEN | SOCBINITFRC | SOCAINITFRC | INTINITFRC | RESERVED | |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R=0-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |

**Table 18-82. 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/W | 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/W | 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/W | 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 |

#### 18.6.1.1.66 ETCNTINIT Register (Offset = B4h) [reset = 0h]

ETCNTINIT is shown in Figure 18-143 and described in Table 18-83.

Return to Summary Table.

Event-Trigger Counter Initialization Register

**Figure 18-143. ETCNTINIT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | SOCBINIT | | | |
| R-0h | | | | R/W-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SOCAINIT | | | | INTINIT | | | |
| R/W-0h | | | | R/W-0h | | | |

**Table 18-83. 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 |

#### 18.6.1.1.67 DCTRIPSEL Register (Offset = C0h) [reset = 0h]

DCTRIPSEL is shown in Figure 18-144 and described in Table 18-84.

Return to Summary Table.

Digital Compare Trip Select Register

**Figure 18-144. DCTRIPSEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| DCBLCOMPSEL | | | | DCBHCOMPSEL | | | |
| R/W-0h | | | | R/W-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DCALCOMPSEL | | | | DCAHCOMPSEL | | | |
| R/W-0h | | | | R/W-0h | | | |

**Table 18-84. 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 and (TZ1 input)<br>0001: TRIPIN2 and (TZ2 input)<br>0010: TRIPIN3 and (TZ3 input)<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 and (TZ1 input)<br>0001: TRIPIN2 and (TZ2 input)<br>0010: TRIPIN3 and (TZ3 input)<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 |

**Table 18-84. DCTRIPSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7-4 | DCALCOMPSEL | R/W | 0h | Digital Compare A Low Input Select Bits<br>0000: TRIPIN1 and (TZ1 input)<br>0001: TRIPIN2 and (TZ2 input)<br>0010: TRIPIN3 and (TZ3 input)<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 |
| 3-0 | DCAHCOMPSEL | R/W | 0h | Digital Compare A High Input Select Bits<br>0000: TRIPIN1 and (TZ1 input)<br>0001: TRIPIN2 and (TZ2 input)<br>0010: TRIPIN3 and (TZ3 input)<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 |

#### 18.6.1.1.68 DCACTL Register (Offset = C3h) [reset = 0h]

DCACTL is shown in Figure 18-145 and described in Table 18-85.

Return to Summary Table.

Digital Compare A Control Register

**Figure 18-145. 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 |
| | | | | EVT1SYNCE | EVT1SOCE | EVT1FRCSYNCSEL | EVT1SRCSEL |
| | | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 18-85. DCACTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14-13 | RESERVED | R | 0h | Reserved |
| 12 | RESERVED | R | 0h | Reserved |
| 11-10 | RESERVED | R=0 | 0h | Reserved |
| 9 | EVT2FRCSYNCSEL | R/W | 0h | DCAEVT2 Force Synchronization Signal Select<br>0: Source is passed through asynchronously<br>1: Source is synchronized with EPWMCLK<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 | RESERVED | R | 0h | Reserved |
| 6-5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 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 passed through asynchronously<br>1: Source is synchronized with EPWMCLK<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 |

#### 18.6.1.1.69 DCBCTL Register (Offset = C4h) [reset = 0h]

DCBCTL is shown in Figure 18-146 and described in Table 18-86.

Return to Summary Table.

Digital Compare B Control Register

**Figure 18-146. 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 |
| | | | | EVT1SYNCE | EVT1SOCE | EVT1FRCSYNCSEL | EVT1SRCSEL |
| | | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 18-86. DCBCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14-13 | RESERVED | R | 0h | Reserved |
| 12 | RESERVED | R | 0h | Reserved |
| 11-10 | RESERVED | R=0 | 0h | Reserved |
| 9 | EVT2FRCSYNCSEL | R/W | 0h | DCBEVT2 Force Synchronization Signal Select<br>0: Source is passed through asynchronously<br>1: Source is synchronized with EPWMCLK<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 | RESERVED | R | 0h | Reserved |
| 6-5 | RESERVED | R | 0h | Reserved |
| 4 | RESERVED | R | 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 passed through asynchronously<br>1: Source is synchronized with EPWMCLK<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 |

#### 18.6.1.1.70 DCFCTL Register (Offset = C7h) [reset = 0h]

DCFCTL is shown in Figure 18-147 and described in Table 18-87.

Return to Summary Table.

Digital Compare Filter Control Register

**Figure 18-147. DCFCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| EDGESTATUS | | | EDGECOUNT | | | EDGEMODE | |
| R-0h | | | R/W-0h | | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | EDGEFILTSEL | PULSESEL | | BLANKINV | BLANKE | SRCSEL | |
| R=0-0h | R/W-0h | R/W-0h | | R/W-0h | R/W-0h | R/W-0h | |

**Table 18-87. DCFCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | EDGESTATUS | R | 0h | Edge Status:<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>Reset type: SYSRSn |
| 9-8 | EDGEMODE | R/W | 0h | Edge Mode Select:<br>Reset type: SYSRSn |
| 7 | RESERVED | R=0 | 0h | Reserved |
| 6 | EDGEFILTSEL | R/W | 0h | Edge Filter Select:<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 |
| 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 |

#### 18.6.1.1.71 DCCAPCTL Register (Offset = C8h) [reset = 0h]

DCCAPCTL is shown in Figure 18-148 and described in Table 18-88.

Return to Summary Table.

Digital Compare Capture Control Register

**Figure 18-148. DCCAPCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| CAPMODE | CAPCLR | CAPSTS | RESERVED | | | | |
| R/W-0h | R=0/W=1-0h | R-0h | R=0-0h | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | SHDWMODE | CAPE |
| R=0-0h | | | | | | R/W-0h | R/W-0h |

**Table 18-88. DCCAPCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | CAPMODE | R/W | 0h | Counter Capture Mode<br><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><br>If active mode is enabled, via SHDWMODE bit in DCCAPCTL register, CPU reads of this register will return the active register value.<br><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><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><br>If active mode is enabled, via SHDWMODE bit in DCCAPCTL register, CPU reads of this register will return the active register value.<br><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/W=1 | 0h | DC Capture Latched Status Clear Flag<br><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><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 |

**Table 18-88. 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 |

#### 18.6.1.1.72 DCFOFFSET Register (Offset = C9h) [reset = 0h]

DCFOFFSET is shown in Figure 18-149 and described in Table 18-89.

Return to Summary Table.

Digital Compare Filter Offset Register

**Figure 18-149. DCFOFFSET Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DCFOFFSET | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 18-89. DCFOFFSET Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | DCFOFFSET | R/W | 0h | Blanking Window Offset<br><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 |

#### 18.6.1.1.73 DCFOFFSETCNT Register (Offset = CAh) [reset = 0h]

DCFOFFSETCNT is shown in Figure 18-150 and described in Table 18-90.

Return to Summary Table.

Digital Compare Filter Offset Counter Register

**Figure 18-150. DCFOFFSETCNT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DCFOFFSETCNT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 18-90. DCFOFFSETCNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | DCFOFFSETCNT | R | 0h | Blanking Offset Counter<br><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 |

#### 18.6.1.1.74 DCFWINDOW Register (Offset = CBh) [reset = 0h]

DCFWINDOW is shown in Figure 18-151 and described in Table 18-91.

Return to Summary Table.

Digital Compare Filter Window Register

**Figure 18-151. DCFWINDOW Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DCFWINDOW | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 18-91. 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-FFh: Specifies the width of the blanking window in TBCLK cycles. The blanking window begins when the offset counter expires. When this occurs, the window counter is loaded and begins to count down. If the blanking window is currently active and the offset counter expires, the blanking window counter is 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 |

#### 18.6.1.1.75 DCFWINDOWCNT Register (Offset = CCh) [reset = 0h]

DCFWINDOWCNT is shown in Figure 18-152 and described in Table 18-92.

Return to Summary Table.

Digital Compare Filter Window Counter Register

**Figure 18-152. DCFWINDOWCNT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DCFWINDOWCNT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 18-92. 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 |

#### 18.6.1.1.76 DCCAP Register (Offset = CFh) [reset = 0h]

DCCAP is shown in Figure 18-153 and described in Table 18-93.

Return to Summary Table.

Digital Compare Counter Capture Register

**Figure 18-153. DCCAP Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DCCAP | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 18-93. DCCAP Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | DCCAP | R | 0h | Digital Compare Time-Base Counter Capture<br><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><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><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><br>Reset type: SYSRSn |

#### 18.6.1.1.77 DCAHTRIPSEL Register (Offset = D2h) [reset = 0h]

DCAHTRIPSEL is shown in Figure 18-154 and described in Table 18-94.

Return to Summary Table.

Digital Compare AH Trip Select

**Figure 18-154. DCAHTRIPSEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | TRIPINPUT15 | TRIPINPUT14 | RESERVED | TRIPINPUT12 | TRIPINPUT11 | TRIPINPUT10 | TRIPINPUT9 |
| R-0h | R/W-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TRIPINPUT8 | TRIPINPUT7 | TRIPINPUT6 | TRIPINPUT5 | TRIPINPUT4 | TRIPINPUT3 | TRIPINPUT2 | TRIPINPUT1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 18-94. 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 | 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 |

**Table 18-94. DCAHTRIPSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 18.6.1.1.78 DCALTRIPSEL Register (Offset = D3h) [reset = 0h]

DCALTRIPSEL is shown in Figure 18-155 and described in Table 18-95.

Return to Summary Table.

Digital Compare AL Trip Select

**Figure 18-155. DCALTRIPSEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | TRIPINPUT15 | TRIPINPUT14 | RESERVED | TRIPINPUT12 | TRIPINPUT11 | TRIPINPUT10 | TRIPINPUT9 |
| R-0h | R/W-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TRIPINPUT8 | TRIPINPUT7 | TRIPINPUT6 | TRIPINPUT5 | TRIPINPUT4 | TRIPINPUT3 | TRIPINPUT2 | TRIPINPUT1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 18-95. 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 | 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 |

**Table 18-95. DCALTRIPSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 18.6.1.1.79 DCBHTRIPSEL Register (Offset = D4h) [reset = 0h]

DCBHTRIPSEL is shown in Figure 18-156 and described in Table 18-96.

Return to Summary Table.

Digital Compare BH Trip Select

**Figure 18-156. DCBHTRIPSEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | TRIPINPUT15 | TRIPINPUT14 | RESERVED | TRIPINPUT12 | TRIPINPUT11 | TRIPINPUT10 | TRIPINPUT9 |
| R-0h | R/W-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TRIPINPUT8 | TRIPINPUT7 | TRIPINPUT6 | TRIPINPUT5 | TRIPINPUT4 | TRIPINPUT3 | TRIPINPUT2 | TRIPINPUT1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 18-96. 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 | 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 |

**Table 18-96. DCBHTRIPSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 18.6.1.1.80 DCBLTRIPSEL Register (Offset = D5h) [reset = 0h]

DCBLTRIPSEL is shown in Figure 18-157 and described in Table 18-97.

Return to Summary Table.

Digital Compare BL Trip Select

**Figure 18-157. DCBLTRIPSEL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | TRIPINPUT15 | TRIPINPUT14 | RESERVED | TRIPINPUT12 | TRIPINPUT11 | TRIPINPUT10 | TRIPINPUT9 |
| R-0h | R/W-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TRIPINPUT8 | TRIPINPUT7 | TRIPINPUT6 | TRIPINPUT5 | TRIPINPUT4 | TRIPINPUT3 | TRIPINPUT2 | TRIPINPUT1 |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 18-97. 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 | 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 |

**Table 18-97. DCBLTRIPSEL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 18.6.1.1.81 EPWMLOCK Register (Offset = FAh) [reset = 0h]

EPWMLOCK is shown in Figure 18-158 and described in Table 18-98.

Return to Summary Table.

EPWM Lock Register

**Figure 18-158. EPWMLOCK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| KEY | | | | | | | |
| R=0/W-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| KEY | | | | | | | |
| R=0/W-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | DCLOCK | TZCLRLOCK | TZCFGLOCK | GLLOCK | HRLOCK |
| R-0h | | | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h | R/WOnce-0h |

**Table 18-98. EPWMLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | KEY | R=0/W | 0h | Write to this register succeeds only if this field is written with a value of 0xa5a5<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: SYSRSn |
| 15-5 | RESERVED | R | 0h | Reserved |
| 4 | DCLOCK | R/WOnce | 0h | 0:Digital Compare registers from 0xC0 to 0xD5 offsets are protected by EALLOW.<br>1: Digital Compare registers from 0xC0 and 0xD5 offsets are locked and not writable.<br>Reset type: SYSRSn |
| 3 | TZCLRLOCK | R/WOnce | 0h | 0:Digital Compare registers from 0x97 to 0x9B offsets are protected by EALLOW.<br>1: Digital Compare registers from 0x97 and 0x9B offsets are locked and not writable.<br>Reset type: SYSRSn |
| 2 | TZCFGLOCK | R/WOnce | 0h | 0:TripZone registers from 0x80 to 0x8D offsets are protected by EALLOW.<br>1: TripZone registers from 0x80 and 0x8D offsets are locked and not writable.<br>Reset type: SYSRSn |
| 1 | GLLOCK | R/WOnce | 0h | 0:TripZone registers from 0x34 to 0x35 offsets are protected by EALLOW.<br>1: TripZone registers from 0x34 to 0x35 offsets are locked and not writable<br>Reset type: SYSRSn |

**Table 18-98. EPWMLOCK Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | HRLOCK | R/WOnce | 0h | 0: HRPWM registers from 0x20 to 0x2D offsets are protected by EALLOW<br>1: HRPWM registers from 0x20 and 0x2D offsets are locked and not writable.<br>Reset type: SYSRSn |

#### 18.6.1.1.82 HWVDELVAL Register (Offset = FDh) [reset = 0h]

HWVDELVAL is shown in Figure 18-159 and described in Table 18-99.

Return to Summary Table.

Hardware Valley Mode Delay Register

**Figure 18-159. HWVDELVAL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HWVDELVAL | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 18-99. HWVDELVAL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | HWVDELVAL | R | 0h | Hardware Valley Delay Value Register<br><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 |

#### 18.6.1.1.83 VCNTVAL Register (Offset = FEh) [reset = 0h]

VCNTVAL is shown in Figure 18-160 and described in Table 18-100.

Return to Summary Table.

Hardware Valley Counter Register

**Figure 18-160. VCNTVAL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VCNTVAL | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 18-100. 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 |

### 18.6.1.2 SYNC_SOC_REGS Registers

Table 18-101 lists the memory-mapped registers for the SYNC_SOC_REGS. All register offset addresses not listed in Table 18-101 should be considered as reserved locations and the register contents should not be modified.

**Table 18-101. SYNC_SOC_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | SYNCSELECT | Sync Input and Output Select Register | EALLOW | Go |
| 2h | ADCSOCOUTSELECT | External ADCSOC 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 18-102 shows the codes that are used for access types in this section.

**Table 18-102. SYNC_SOC_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| WSOnce | SOnce<br>W | Set once<br>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. |

#### 18.6.1.2.1 SYNCSELECT Register (Offset = 0h) [reset = E003FFFFh]

SYNCSELECT is shown in Figure 18-161 and described in Table 18-103.

Return to Summary Table.

Sync Input and Output Select Register

**Figure 18-161. SYNCSELECT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| EPWM1SYNCIN | | | SYNCOUT | | RESERVED | | |
| R/W-7h | | | R/W-0h | | R=0-0h | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | ECAP6SYNCIN | |
| R=0-0h | | | | | | R/W-7h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| ECAP6SYNCIN | ECAP4SYNCIN | | | ECAP1SYNCIN | | | RESERVED |
| R/W-7h | R/W-7h | | | R/W-7h | | | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | EPWM7SYNCIN | | | EPWM4SYNCIN | | |
| R-0h | | R/W-7h | | | R/W-7h | | |

**Table 18-103. SYNCSELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-29 | EPWM1SYNCIN | R/W | 7h | Selects Sync Input Source for EPWM1:<br>000: EXTSYNCIN1 selected<br>001: Reserved<br>010: Reserved<br>011: Reserved<br>100: Reserved<br>101: Reserved<br>110: Reserved<br>111: Reserved<br>Notes:<br>[1] Programming Reserved positions cause O/p of the Sync. Mux to be driven low.<br>Reset type: SYSRSn |
| 28-27 | SYNCOUT | R/W | 0h | Select Syncout Source:<br>00: EPWM1SYNCOUT selected<br>01: EPWM4SYNCOUT selected<br>10: EPPW7SYNCOUT selected<br>11: EPWM10SYNCOUT selected(Reserved)<br>Notes:<br>[1] Reserved position defaults to 00 selection<br>Reset type: SYSRSn |
| 26-18 | RESERVED | R=0 | 0h | Reserved |

**Table 18-103. SYNCSELECT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 17-15 | ECAP6SYNCIN | R/W | 7h | Selects Sync Input Source for ECAP4:<br>000: EPWM1SYNCOUT selected<br>001: EPWM4SYNCOUT selected<br>010: EPPW7SYNCOUT selected<br>011: EPWM10SYNCOUT selected(Reserved)<br>100: ECAP1SYNCOUT selected<br>101: EXTSYNCIN1 selected<br>110: EXTSYNCIN2 selected<br>111: ECAP4SYNCOUT selected<br>Notes:<br>[1] Programming Reserved positions cause O/p of the Sync. Mux to be driven low.<br>Reset type: SYSRSn |
| 14-12 | ECAP4SYNCIN | R/W | 7h | Selects Sync Input Source for ECAP4:<br>000: EPWM1SYNCOUT selected<br>001: EPWM4SYNCOUT selected<br>010: EPPW7SYNCOUT selected<br>011: EPWM10SYNCOUT selected(Reserved)<br>100: ECAP1SYNCOUT selected<br>101: EXTSYNCIN1 selected<br>110: EXTSYNCIN2 selected<br>111: ECAP4SYNCOUT selected (Reserved)<br>Notes:<br>[1] Programming Reserved positions cause O/p of the Sync. Mux to be driven low.<br>Reset type: SYSRSn |
| 11-9 | ECAP1SYNCIN | R/W | 7h | Selects Sync Input Source for ECAP1:<br>000: EPWM1SYNCOUT selected<br>001: EPWM4SYNCOUT selected<br>010: EPPW7SYNCOUT selected<br>011: EPWM10SYNCOUT selected(Reserved)<br>100: ECAP1SYNCOUT selected (Reserved)<br>101: EXTSYNCIN1 selected<br>110: EXTSYNCIN2 selected<br>111: ECAP4SYNCOUT selected (Reserved)<br>Notes:<br>[1] Programming Reserved positions cause O/p of the Sync. Mux to be driven low.<br>Reset type: SYSRSn |
| 8-6 | RESERVED | R | 0h | Reserved |

**Table 18-103. SYNCSELECT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5-3 | EPWM7SYNCIN | R/W | 7h | Selects Sync Input Source for EPWM7:<br>000: EPWM1SYNCOUT selected<br>001: EPWM4SYNCOUT selected<br>010: EPPW7SYNCOUT selected (Reserved)<br>011: EPWM10SYNCOUT selected (Reserved)<br>100: ECAP1SYNCOUT selected (Reserved)<br>101: EXTSYNCIN1 selected<br>110: EXTSYNCIN2 selected<br>111: ECAP4SYNCOUT selected (Reserved)<br>Notes:<br>[1] Programming Reserved positions cause O/p of the Sync. Mux to be driven low.<br>Reset type: SYSRSn |
| 2-0 | EPWM4SYNCIN | R/W | 7h | Selects Sync Input Source for EPWM4:<br>000: EPWM1SYNCOUT selected<br>001: EPWM4SYNCOUT selected (Reserved)<br>010: EPPW7SYNCOUT 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] Programming Reserved positions cause O/p of the Sync. Mux to be driven low.<br>Reset type: SYSRSn |

#### 18.6.1.2.2 ADCSOCOUTSELECT Register (Offset = 2h) [reset = 0h]

ADCSOCOUTSELECT is shown in Figure 18-162 and described in Table 18-104.

Return to Summary Table.

External ADCSOC Select Register

**Figure 18-162. ADCSOCOUTSELECT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | RESERVED | RESERVED | RESERVED | RESERVED |
| R=0-0h | | | | R-0h | R-0h | R-0h | R-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 | | | | RESERVED | RESERVED | RESERVED | RESERVED |
| R=0-0h | | | | R-0h | R-0h | R-0h | R-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 18-104. ADCSOCOUTSELECT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | RESERVED | R=0 | 0h | Reserved |
| 27 | RESERVED | R | 0h | Reserved |
| 26 | RESERVED | R | 0h | Reserved |
| 25 | RESERVED | R | 0h | Reserved |
| 24 | RESERVED | R | 0h | Reserved |
| 23 | PWM8SOCBEN | R/W | 0h | ADCSOCBOn source select:<br>0: Respective EPWM SOCB output is not selected<br>1: Respective EPWM SOCB output is selected<br>Reset type: SYSRSn |
| 22 | PWM7SOCBEN | R/W | 0h | ADCSOCBOn source select:<br>0: Respective EPWM SOCB output is not selected<br>1: Respective EPWM SOCB output is selected<br>Reset type: SYSRSn |
| 21 | PWM6SOCBEN | R/W | 0h | ADCSOCBOn source select:<br>0: Respective EPWM SOCB output is not selected<br>1: Respective EPWM SOCB output is selected<br>Reset type: SYSRSn |
| 20 | PWM5SOCBEN | R/W | 0h | ADCSOCBOn source select:<br>0: Respective EPWM SOCB output is not selected<br>1: Respective EPWM SOCB output is selected<br>Reset type: SYSRSn |
| 19 | PWM4SOCBEN | R/W | 0h | ADCSOCBOn source select:<br>0: Respective EPWM SOCB output is not selected<br>1: Respective EPWM SOCB output is selected<br>Reset type: SYSRSn |

**Table 18-104. ADCSOCOUTSELECT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18 | PWM3SOCBEN | R/W | 0h | ADCSOCBOn source select:<br>0: Respective EPWM SOCB output is not selected<br>1: Respective EPWM SOCB output is selected<br>Reset type: SYSRSn |
| 17 | PWM2SOCBEN | R/W | 0h | ADCSOCBOn source select:<br>0: Respective EPWM SOCB output is not selected<br>1: Respective EPWM SOCB output is selected<br>Reset type: SYSRSn |
| 16 | PWM1SOCBEN | R/W | 0h | ADCSOCBOn source select:<br>0: Respective EPWM SOCB output is not selected<br>1: Respective EPWM SOCB output is selected<br>Reset type: SYSRSn |
| 15-12 | RESERVED | R=0 | 0h | Reserved |
| 11 | RESERVED | R | 0h | Reserved |
| 10 | RESERVED | R | 0h | Reserved |
| 9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 7 | PWM8SOCAEN | R/W | 0h | ADCSOCAOn source select:<br>0: Respective EPWM SOCA output is not selected<br>1: Respective EPWM SOCA output is selected<br>Reset type: SYSRSn |
| 6 | PWM7SOCAEN | R/W | 0h | ADCSOCAOn source select:<br>0: Respective EPWM SOCA output is not selected<br>1: Respective EPWM SOCA output is selected<br>Reset type: SYSRSn |
| 5 | PWM6SOCAEN | R/W | 0h | ADCSOCAOn source select:<br>0: Respective EPWM SOCA output is not selected<br>1: Respective EPWM SOCA output is selected<br>Reset type: SYSRSn |
| 4 | PWM5SOCAEN | R/W | 0h | ADCSOCAOn source select:<br>0: Respective EPWM SOCA output is not selected<br>1: Respective EPWM SOCA output is selected<br>Reset type: SYSRSn |
| 3 | PWM4SOCAEN | R/W | 0h | ADCSOCAOn source select:<br>0: Respective EPWM SOCA output is not selected<br>1: Respective EPWM SOCA output is selected<br>Reset type: SYSRSn |
| 2 | PWM3SOCAEN | R/W | 0h | ADCSOCAOn source select:<br>0: Respective EPWM SOCA output is not selected<br>1: Respective EPWM SOCA output is selected<br>Reset type: SYSRSn |
| 1 | PWM2SOCAEN | R/W | 0h | ADCSOCAOn source select:<br>0: Respective EPWM SOCA output is not selected<br>1: Respective EPWM SOCA output is selected<br>Reset type: SYSRSn |

**Table 18-104. ADCSOCOUTSELECT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | PWM1SOCAEN | R/W | 0h | ADCSOCAOn source select:<br>0: Respective EPWM SOCA output is not selected<br>1: Respective EPWM SOCA output is selected<br>Reset type: SYSRSn |

#### 18.6.1.2.3 SYNCSOCLOCK Register (Offset = 4h) [reset = 0h]

SYNCSOCLOCK is shown in Figure 18-163 and described in Table 18-105.

Return to Summary Table.

SYNCSEL and EXTADCSOC Select Lock register

**Figure 18-163. 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 18-105. SYNCSOCLOCK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-2 | 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 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: 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 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: SYSRSn |

# High-Resolution Pulse Width Modulator (HRPWM)

This document is used in conjunction with the device-specific *Enhanced Pulse Width Modulator (ePWM) Module Reference Guide.* The HRPWM module described in this reference guide is a Type 2 HRPWM. See the *TMS320x28xx, 28xxx DSP Peripheral Reference Guide* (SPRU566) for a list of all devices with an HRPWM module of the same type, to determine the differences between types, and for a list of device-specific differences within a type.

**Topic** **Page**

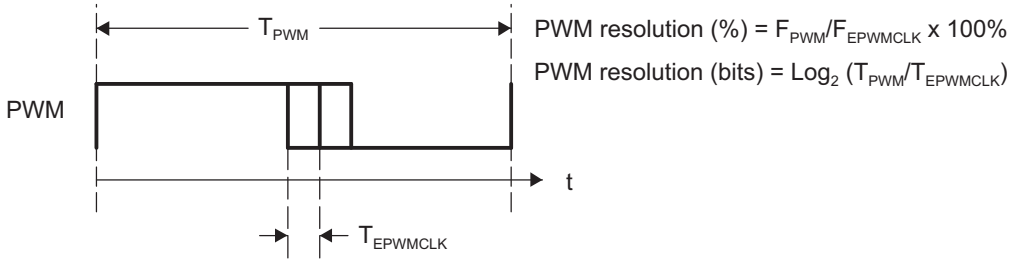
## 19.1 Introduction

This module extends the time resolution capabilities of the conventionally derived digital pulse width modulator (PWM). HRPWM is typically used when PWM resolution falls below ~ 9-10 bits. The key features of HRPWM are:

- Extended time resolution capability
- Used in both duty cycle and phase-shift control methods
- Finer ti me granularity control or edge positioning using extensions to the Compare A, Compare B and Phase registers
- Implemented using the A & B signal path of PWM, that is, on the EPWMxA and EPWMxB output.
- Dead band high-resolution control for falling and rising edge delay in half cycle clocking operation
- Self-check diagnostics software mode to check if the micro edge positioner (MEP) logic is running optimally
- Enables high resolution output swapping on the EPWMxA and EPWMxB output
- Enables high-resolution output on EPWMxB signal output via inversion of EPWMxA signal output
- Enables high-resolution period, duty and phase control on the EPWMxA and EPWMxB output on devices with an ePWM module. See the device-specific data manual to determine if your device has an ePWM module for high-resolution period support.

The ePWM peripheral is used to perform a function mathematically equivalent to a digital-to-analog converter (DAC). As shown in Figure 19-1, the effective resolution for conventionally generated PWM is a function of PWM frequency (or period) and system clock frequency.

**Figure 19-1. Resolution Calculations for Conventionally Generated PWM**




If the required PWM operating frequency does not offer sufficient resolution in PWM mode, you may want to consider HRPWM. As an example of improved performance offered by HRPWM, Table 19-1 shows resolution in bits for various PWM frequencies. These values assume a MEP step size of 180 ps. See the device-specific data sheet for typical and maximum performance specifications for the MEP.

**Table 19-1. Resolution for PWM and HRPWM**

| PWM Freq | Regular Resolution (PWM) | | High Resolution (HRPWM) | |
|---|---|---|---|---|
| | 100 MHz EPWMCLK | | | |
| (kHz) | Bits | % | Bits | % |
| 20 | 12.3 | 0.02 | 18.1 | 0.000 |
| 50 | 11 | 0.05 | 16.8 | 0.001 |
| 100 | 10 | 0.1 | 15.8 | 0.002 |
| 150 | 9.4 | 0.15 | 15.2 | 0.003 |
| 200 | 9 | 0.2 | 14.8 | 0.004 |
| 250 | 8.6 | 0.25 | 14.4 | 0.005 |
| 500 | 7.6 | 0.5 | 13.4 | 0.009 |
| 1000 | 6.6 | 1 | 12.4 | 0.018 |
| 1500 | 6.1 | 1.5 | 11.9 | 0.027 |
| 2000 | 5.6 | 2 | 11.4 | 0.036 |

Although each application may differ, typical low frequency PWM operation (below 250 kHz) may not require HRPWM. HRPWM capability is most useful for high frequency PWM requirements of power conversion topologies such as:

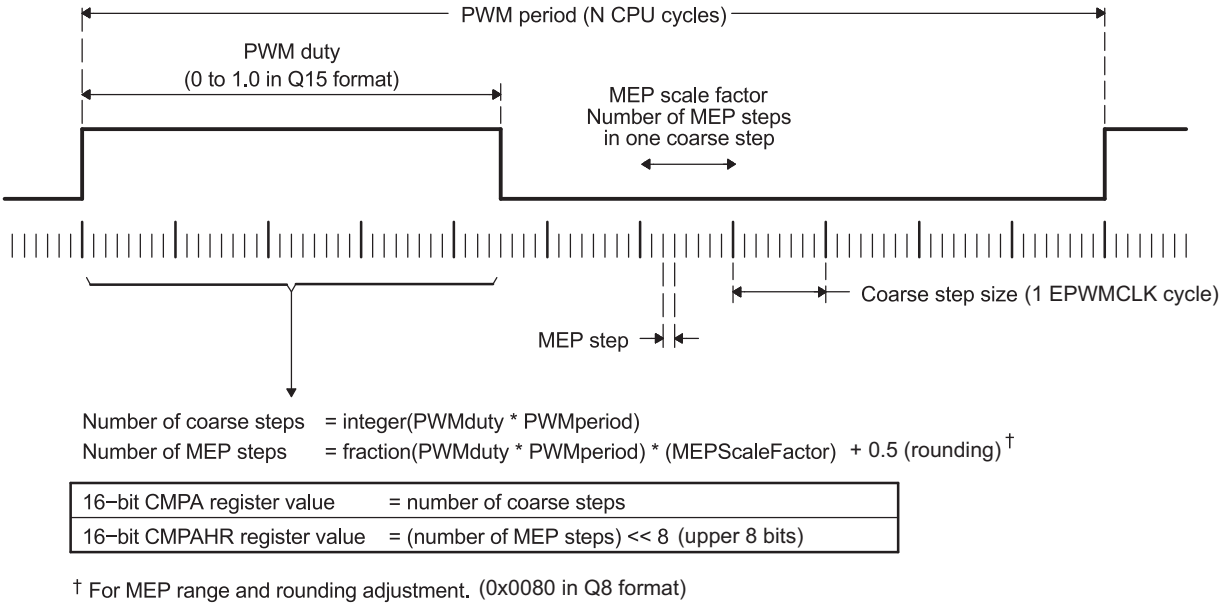
- Single-phase buck, boost, and flyback
- Multi-phase buck, boost, and flyback
- Phase-shifted full bridge
- Direct modulation of D-Class power amplifiers

## 19.2 Operational Description of HRPWM

The HRPWM is based on micro-edge positioner (MEP) technology. MEP logic is capable of positioning an edge very finely by subdividing one coarse system clock of a conventional PWM generator. The time step accuracy is on the order of 150 ps. See the device-specific data manual for the typical MEP step size on a particular device. The HRPWM also has a self-check software diagnostics mode to check if the MEP logic is running optimally, under all operating conditions. Details on software diagnostics and functions are in Section 19.2.6.

Figure 19-2 shows the relationship between one coarse system clock and edge position in terms of MEP steps, which are controlled via an 8-bit field in the Compare A extension register (CMPAHR). The same operating logic applies to CMPBHR as well.

**Figure 19-2. Operating Logic Using MEP**




To generate an HRPWM waveform, configure the ePWM registers as you would to generate a conventional PWM of a given frequency and polarity. The HRPWM works together with the ePWM registers to extend edge resolution. Although many programming combinations are possible, only a few are needed and practical. These methods are described in Section 19.2.7.

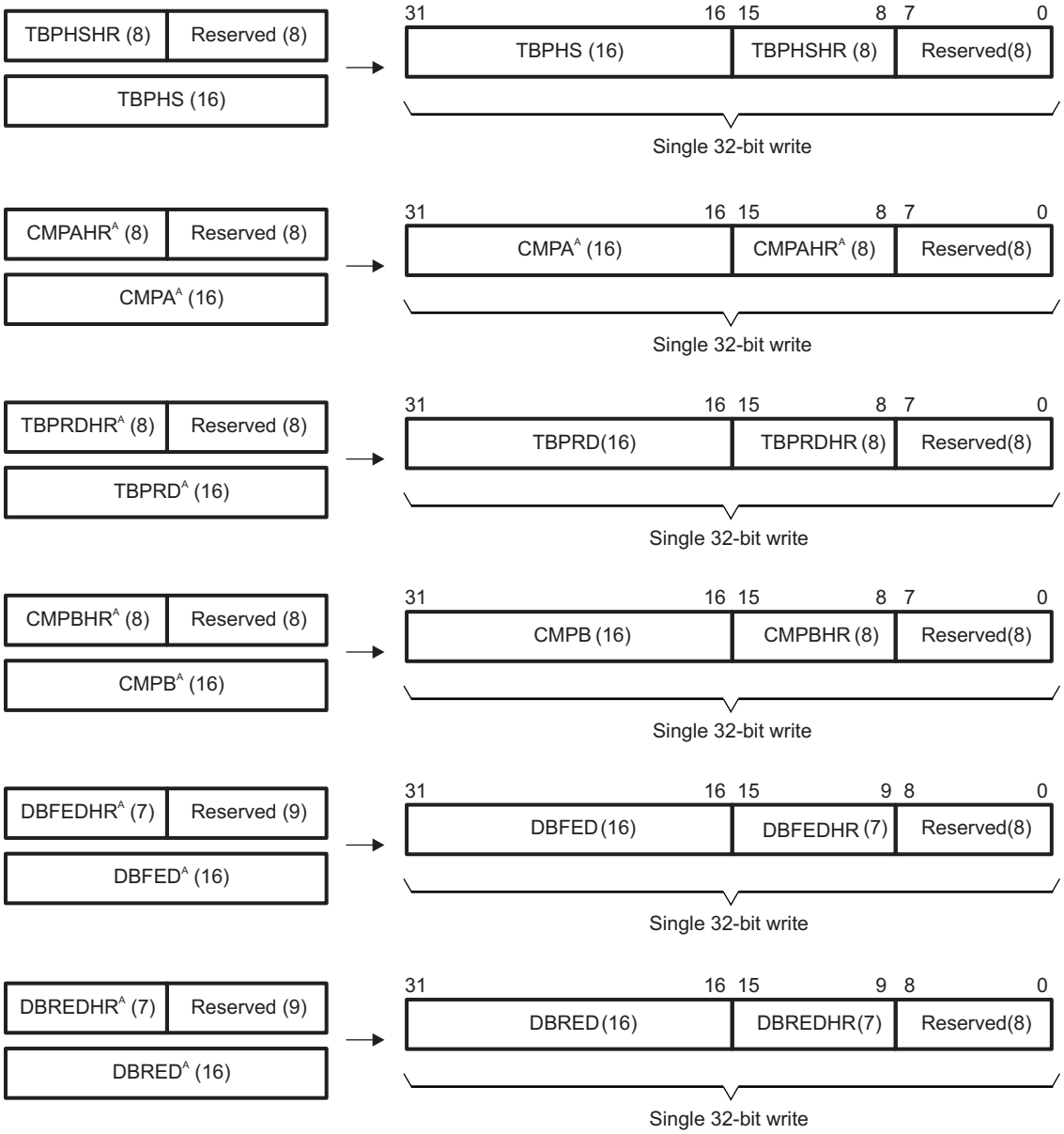
Registers discussed but not found in this document can be seen in the device-specific *Enhanced Pulse Width Modulator (ePWM) Module Reference Guide*.

### *19.2.1 Controlling the HRPWM Capabilities*

The MEP of the HRPWM is controlled by six extension registers. These HRPWM registers are concatenated with the 16-bit TBPHS, TBPRD, CMPA, CMPBM, DBREDM & DBFEDM registers used to control PWM operation.

- TBPHSHR - Time Base Phase High Resolution Register
- CMPAHR - Counter Compare A High Resolution Register
- TBPRDHR - Time Base Period High Resolution Register. (available on some devices)
- CMPBHR - Compare B High Resolution Register
- DBREDHR - Deadband Generator Rising Edge Delay High Resolution Register
- DBFEDHR - Deadband Generator Falling Edge Delay High Resolution Register

**Figure 19-3. HRPWM Extension Registers and Memory Configuration**

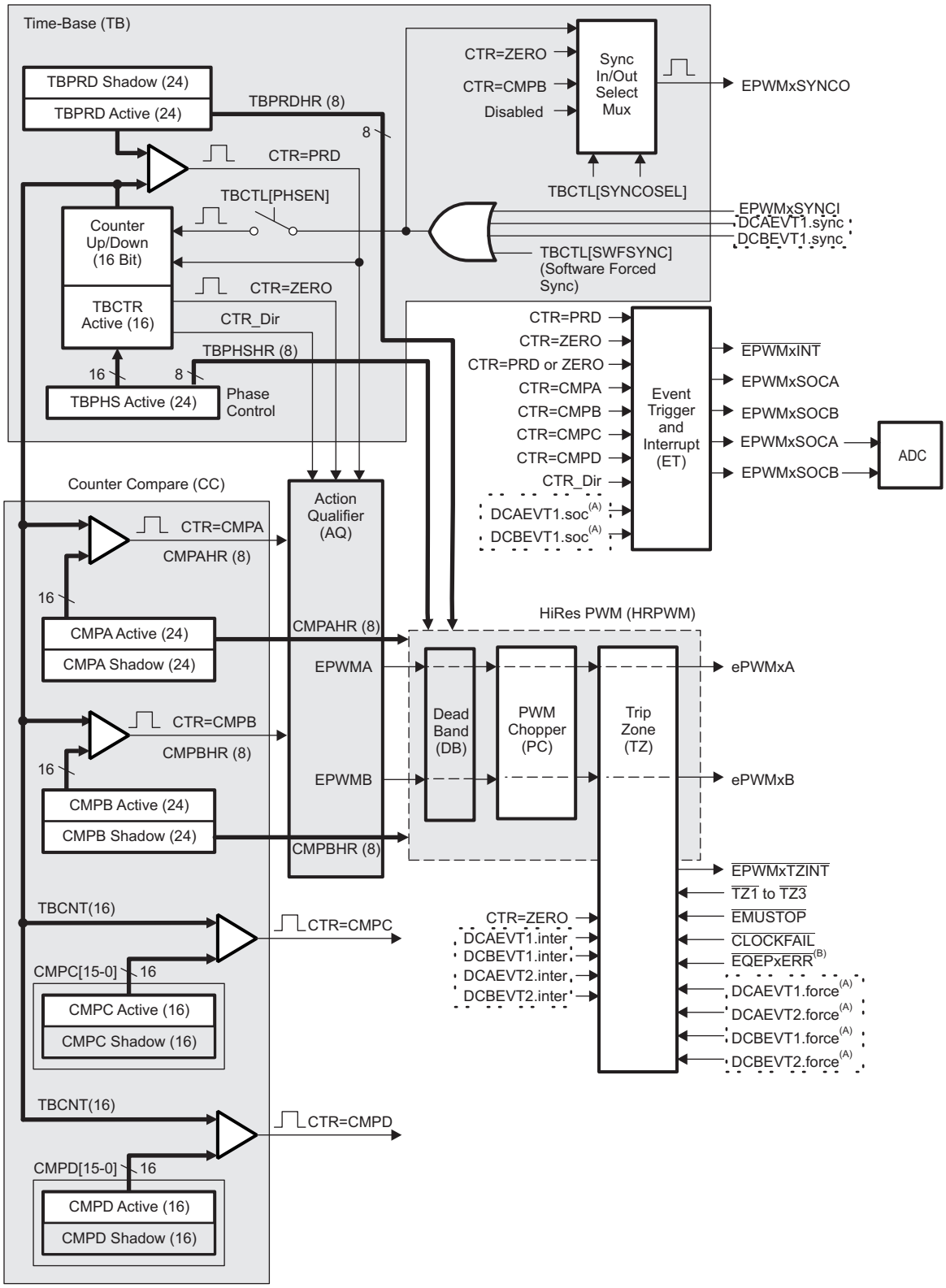



A Dependant upon your device, these registers may be mirrored and can be written to at two different memory locations. Check the device-specific Technical Reference Manual's *ePWM* chapter for more details on how to read and write to these locations.

**NOTE:** HRPWM capabilities on Deadband Rising Edge Delay and Falling Edge Delay is applicable only during Dead Band half cycle clocking Operation. The number of MEP steps is half in size [ bits 15:9 ]than duty and phase high resolution registers for the same reason

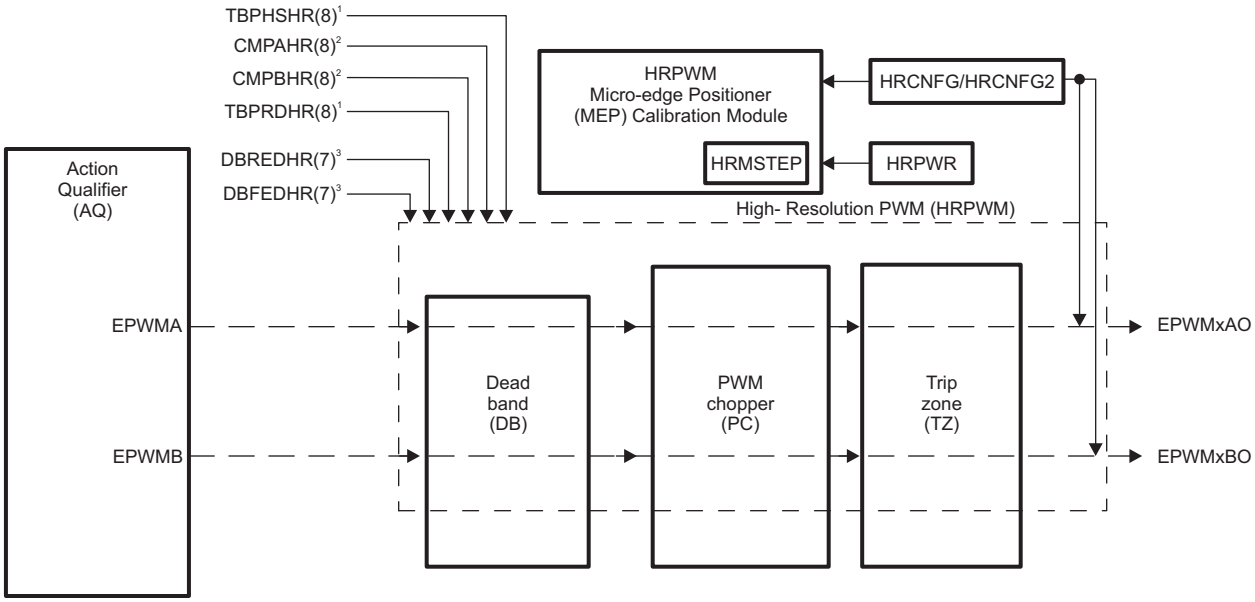
HRPWM capabilities are controlled using the Channel A & B PWM signal path. HRPWM support on the Dead band signal path is available by properly configuring the HRCNFG2 register. Figure 19-4 shows how the HRPWM interfaces with the 8-bit extension registers.

**Figure 19-4. HRPWM System Interface**




A These events are generated by the ePWM digital compare (DC) submodule.

**Figure 19-5. HRPWM Block Diagram**




(1) From ePWM Time-base (TB) submodule

(2) From ePWM counter-compare (CC) submodule

(3) From ePWM Deadband (DB) submodule

### 19.2.2 Configuring the HRPWM

Once the ePWM has been configured to provide conventional PWM of a given frequency and polarity, the HRPWM is configured by programming the HRCNFG register in that particular ePWM module's register space. This register provides the following configuration options:

**Edge Mode —** The MEP can be programmed to provide precise position control on the rising edge (RE), falling edge (FE) or both edges (BE) at the same time. FE and RE are used for power topologies requiring duty cycle control(CMPA or CMPB high-resolution control), while BE is used for topologies requiring phase shifting, for example, phase shifted full bridge (TBPHS or TBPRD high-resolution control).

**Control Mode —** The MEP is programmed to be controlled either from the CMPAHR / CMPBHR register in case of duty cycle control or the TBPHSHR register (phase control). RE or FE control mode should be used with CMPAHR or CMPBHR register. BE control mode should be used with TBPHSHR register. When the MEP is controlled from the TBPRDHR register (period control) the duty cycle and phase can also be controlled via their respective high-resolution registers.

**Shadow Mode —** This mode provides the same shadowing (double buffering) option as in regular PWM mode. This option is valid only when operating from the CMPAHR, CMPBHR and TBPRDHR registers and should be chosen to be the same as the regular load option for the CMPA, CMPB register. If TBPHSHR is used, then this option has no effect.

**High-Resolution B Signal Control —** The B signal path of an ePWM channel can generate a high-resolution output by outputting an inverted version of the high-resolution ePWMxA signal on the ePWMxB pin. A Type 2 HRPWM module can also enable high-resolution features on the B signal path indepedently of the A signal path as well.

**Swap ePWMxA and ePWMxB Outputs —** This mode enables the swapping of the high resolution A & B outputs . The mode selection allows either A & B Outputs Unchanged or A Output Comes Out On B and B Output Comes Out On A

**Auto-conversion Mode —** This mode is used in conjunction with the scale factor optimization (SFO) software only. For a type 2 HRPWM module, below is a description of the Auto-conversion Mode taking CMPAHR as an example. If auto-conversion is enabled, CMPAHR = fraction(PWMduty*PWMperiod)<<8. The scale factor optimization software will calculate the MEP scale factor in background code and automatically update the HRMSTEP register with the calculated number of MEP steps per coarse step. The MEP Calibration Module will then use the values in the HRMSTEP and CMPAHR register to automatically calculate the appropriate number of MEP steps represented by the fractional duty cycle and move the high-resolution ePWM signal edge accordingly. If auto-conversion is disabled, the CMPAHR register behaves like a type 0 HRPWM module and CMPAHR = (fraction(PWMduty * PWMperiod) * MEP Scale Factor + 0.5)<<8). All calculations will need to be performed by user code in this mode, and the HRMSTEP register is ignored. Auto-conversion for high-resolution period has the same behavior as auto-conversion for high-resolution duty cycle. Auto-conversion must always be enabled for high-resolution period mode.

**NOTE:** Auto-conversion Mode performs the calculation for CMPBHR , DBREDHR and DBFEDHR as well. The scale factor optimization software will calculate the MEP scale factor in background code and automatically update the HRMSTEP register with the calculated number of MEP steps per coarse step. The MEP Calibration Module will then use the values in the HRMSTEP and CMPBHR or DBREDHR / DBFEDHR register to automatically calculate the appropriate number of MEP steps represented by the fractional components and move the high-resolution ePWM signal edge accordingly. If auto-conversion is disabled, CMPBHR behaves same as CMPAHR. CMPBHR = (fraction(PWMduty * PWMperiod) * MEP Scale Factor + 0.5)<<8).

### 19.2.3 Configuring Hi-Res in Deadband Rising Edge and Falling Edge Delay

Once the ePWM has been configured to provide conventional PWM of a given frequency, polarity and deadband enabled in half cycle clocking mode, the high resolution operation on dead band RED and FED lines are enabled by programming the HRCNFG2 register in that particular ePWM module's register space. This register provides the following configuration options:

**Edge Mode —** The MEP can be programmed to provide precise position control on the dead band rising edge (RED), dead band falling edge (FED) or both edges (rising edge of DBRED signal and falling edge of DBFED signal ) at the same time.

**Control Mode —** Selects the time event that loads the shadow value in active register for DBRED and DBFED in high resolution mode. The user needs to select the pulse to match the selection in the ePWM DBCTL[LOADREDMODE] & DBCTL[LOADFEDMODE] bits .

### 19.2.4 Principle of Operation

The MEP logic is capable of placing an edge in one of 255 (8 bits) discrete time steps (see device-specific data sheet for typical MEP step size). The MEP works with the TBM and CCM registers to be certain that time steps are optimally applied and that edge placement accuracy is maintained over a wide range of PWM frequencies, system clock frequencies and other operating conditions. Table 19-2 shows the typical range of operating frequencies supported by the HRPWM.

**Table 19-2. Relationship Between MEP Steps, PWM Frequency and Resolution**

| System (MHz) | MEP Steps Per EPWMCLK (1)(2)(3) | PWM MIN (Hz) (4) | PWM MAX (MHz) | Res. @ MAX (Bits) (5) |
|---|---|---|---|---|
| 60.0 | 93 | 916 | 3.00 | 10.9 |
| 70.0 | 79 | 1068 | 3.50 | 10.6 |
| 80.0 | 69 | 1221 | 4.00 | 10.4 |
| 90.0 | 62 | 1373 | 4.50 | 10.3 |
| 100.0 | 56 | 1526 | 5.00 | 10.1 |

(1) TBCLK = EPWMCLK.
(2) Table data based on a MEP time resolution of 180 ps (this is an example value. See the device-specific data sheet for MEP limits)
(3) MEP steps applied = $T_{EPWMCLK}$/180 ps in this example.
(4) PWM minimum frequency is based on a maximum period value,(TBPRD = 65535). PWM mode is asymmetrical up-count.
(5) Resolution in bits is given for the maximum PWM frequency stated.

### 19.2.4.1 Edge Positioning

**NOTE:** The below example is presented using [CMPA:CMPAHR] register combination. The theory of operation and equations is same if the user intends to use [CMPBM:CMPBHRM] for duty cycle control.

In a typical power control loop, a digital controller issues a duty command, usually expressed in a per unit or percentage terms. Assume that for a particular operating point, the demanded duty cycle is 0.405 or 40.5% on time and the required converter PWM frequency is 1.25 MHz. In conventional PWM generation with a system clock of 100 MHz, the duty cycle choices are in the vicinity of 40.5%. As shown in Figure 19-6, a compare value of 32 counts (duty = 40%) is the closest to 40.5% that can be attained. This is equivalent to an edge position of 320 ns instead of the desired 324 ns. This data is shown in Table 19-3.

By utilizing the MEP, you can achieve an edge position much closer to the desired point of 324 ns. Table 19-3 shows that in addition to the CMPA value, 22 steps of the MEP (CMPAHR register) will position the edge at 323.96 ns, resulting in almost zero error. In this example, it is assumed that the MEP has a step resolution of 180 ps.

**Figure 19-6. Required PWM Waveform for a Requested Duty = 40.5%**

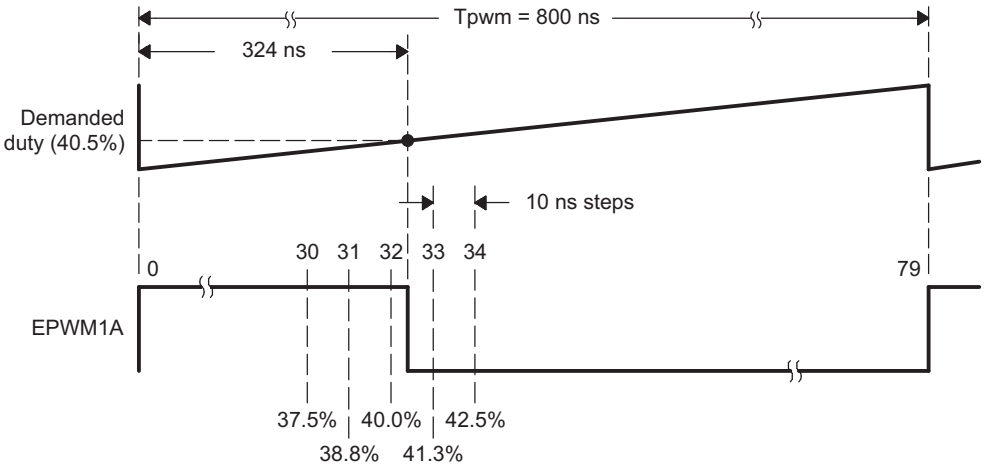


**Table 19-3. CMPA vs Duty (left), and [CMPA:CMPAHR] vs Duty (right)**

| CMPA (count)[1] [2] [3] | DUTY % | High Time (ns) | CMPA (count) | CMPAHR (count) | Duty (%) | High Time (ns) |
|---|---|---|---|---|---|---|
| 28 | ***35.0*** | 280 | 32 | 18 | ***40.405*** | 323.24 |
| 29 | ***36.3*** | 290 | 32 | 19 | ***40.428*** | 323.42 |
| 30 | ***37.5*** | 300 | 32 | 20 | ***40.450*** | 323.60 |
| 31 | ***38.8*** | 310 | 32 | 21 | ***40.473*** | 323.78 |
| 32 | ***40.0*** | 320 | 32 | 22 | ***40.495*** | 323.96 |
| 33 | ***41.3*** | 330 | 32 | 23 | ***40.518*** | 324.14 |
| 34 | ***42.5*** | 340 | 32 | 24 | ***40.540*** | 324.32 |
| | | | 32 | 25 | ***40.563*** | 324.50 |
| Required | | | 32 | 26 | ***40.585*** | 324.68 |
| 32.40 | ***40.5*** | 324 | 32 | 27 | ***40.608*** | 324.86 |

(1) TBCLK = 100 MHz, 10 ns

(2) For a PWM Period register value of 80 counts, PWM Period = 80 x 10 ns = 800 ns, PWM frequency = 1/800 ns = 1.25 MHz

(3) Assumed MEP step size for the above example = 180 ps
See the device-specific data manual for typical and maximum MEP values.

### 19.2.4.2 Scaling Considerations

The mechanics of how to position an edge precisely in time has been demonstrated using the resources of the standard CMPA and MEP (CMPAHR) registers. In a practical application, however, it is necessary to seamlessly provide the CPU a mapping function from a per-unit (fractional) duty cycle to a final integer (non-fractional) representation that is written to the [CMPA:CMPAHR] register combination.

To do this, first examine the scaling or mapping steps involved. It is common in control software to express duty cycle in a per-unit or percentage basis. This has the advantage of performing all needed math calculations without concern for the final absolute duty cycle, expressed in clock counts or high time in ns. Furthermore, it makes the code more transportable across multiple converter types running different PWM frequencies.

To implement the mapping scheme, a two-step scaling procedure is required.

**Assumptions for this example:**

| | | |
|---|---|---|
| TBCLK | = | 10 ns (100 MHz) |
| PWM frequency | = | 1.25 MHz (1/800 ns) |
| Required PWM duty cycle, **PWMDuty** | = | 0.405 (40.5%) |
| PWM period in terms of coarse steps, **PWMPeriod** (800 ns/10 ns) | = | 80 |
| Number of MEP steps per coarse step at 180 ps (10 n/180 ps ), **MEP_ScaleFactor** | = | 55 |
| Value to keep CMPAHR within the range of 1-255 and fractional rounding constant (default value) | = | 0.5 (0080h in Q8 format) |

**Step 1: Percentage Integer Duty value conversion for CMPA register**

| | | |
|---|---|---|
| CMPA register value | = | int(**PWMDuty*****PWMPeriod**); int means integer part |
| | = | int(0.405 * 80 ) |
| | = | int(32.4) |
| CMPA register value | = | 32 (20h) |

**Step 2: Fractional value conversion for CMPAHR register**

| | | |
|---|---|---|
| CMPAHR | = | (frac(**PWMDuty*****PWMPeriod**)***MEP_ScaleFactor** +0.5) << 8); frac means fractional part |
| | = | (frac(32.4) * 55 + 0.5) << 8; Shifting is to move the value to the high byte of CMPAHR. |
| | = | (0.4 * 55 + 0.5) << 8 |
| | = | (22 + 0.5) << 8 |
| | = | 22.5 * 256; Shifting left by 8 is the same as multiplying by 256. |
| | = | 5760 (1680h) |
| CMPAHR | = | 1680h CMPAHR value = 1600h (lower 8 bits will be ignored by hardware). |

**NOTE:** If the AUTOCONV bit (HRCNFG.6) is set and the MEP_ScaleFactor is in the HRMSTEP register, then CMPAHR / CMPBHR register value = frac (PWMDuty*PWMperiod<<8). The rest of the conversion calculations are performed automatically in hardware, and the correct MEP-scaled signal edge appears on the ePWM channel output. If AUTOCONV is not set, the above calculations must be performed by software.

**NOTE:** The MEP scale factor (MEP_ScaleFactor) varies with the system clock and DSP operating conditions. TI provides an MEP scale factor optimizing (SFO) software C function, which uses the built in diagnostics in each HRPWM and returns the best scale factor for a given operating point.

The scale factor varies slowly over a limited range so the optimizing C function can be run very slowly in a background loop.

The CMPA, CMPB, CMPAHR and CMPBHR registers are configured in memory so that the 32-bit data capability of the 28x CPU can write this as a single concatenated value, that is, [CMPA:CMPAHR] , [CMPB:CMPBHR], and so on.

The mapping scheme has been implemented in both C and assembly, as shown in Section 19.2.7. The actual implementation takes advantage of the 32-bit CPU architecture of the 28xx, and is somewhat different from the steps shown in Section 19.2.4.2.

For time-critical control loops where every cycle counts, the assembly version is recommended. This is a cycle optimized function (11 EPWMCLK cycles ) that takes a Q15 duty value as input and writes a single [CMPA:CMPAHR] value.

### 19.2.4.3 Duty Cycle Range Limitation

In high resolution mode, the MEP is not active for 100% of the PWM period. It becomes operational:

- Three EPWMCLK cycles after the period starts when high-resolution period (TBPRDHR) control is not enabled.
- When high resolution period (TBPRDHR) control is enabled via the HRPCTL register:
  - In up-count mode: three EPWMCLK cycles after the period starts until three EPWMCLK cycles before the period ends.
  - In up-down count mode: when counting up, three cycles after CTR = 0 until three cycles before CTR = PRD, and when counting down, three cycles after CTR = PRD until three cycles before CTR = 0.
- When using DBREDHR or DBFEDHR, DBRED and/or DBFED (the register corresponding to the edge with hi-resolution displacement) must be greater than or equal to 3.

Duty cycle range limitations are illustrated in Figure 19-7 to Figure 19-10. This limitation imposes a duty cycle limit on the MEP. For example, precision edge control is not available all the way down to 0% duty cycle. When high-resolution period control is disabled, regular PWM duty control is fully operational down to 0% duty cycle despite the unavailability of HRPWM features in the first three cycles. In most applications this should not be an issue as the controller regulation point is usually not designed to be close to 0% duty cycle. To better understand the useable duty cycle range, see Table 19-4. When high-resolution period control is enabled (HRPCTL[HRPE]=1), the duty cycle must not fall within the restricted range. Otherwise, there may be undefined behavior on the ePWMxA output.

**Figure 19-7. Low % Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 0)**

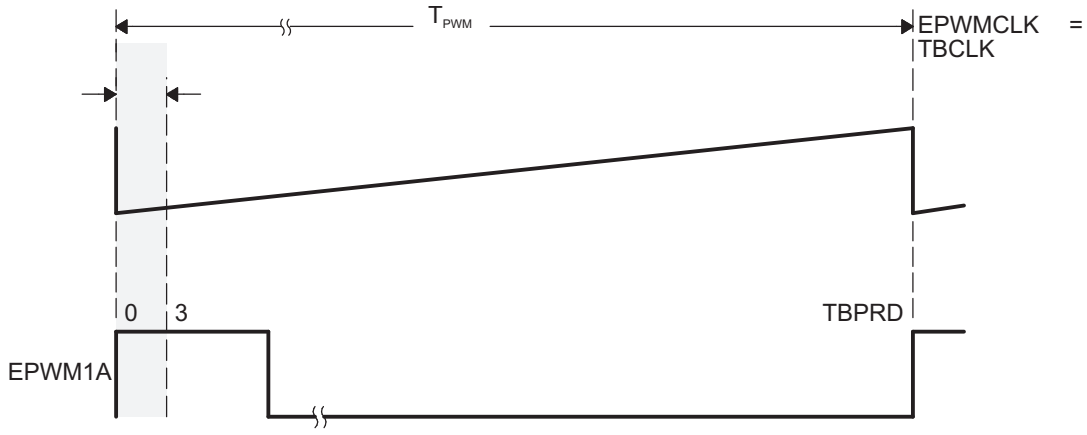


**Table 19-4. Duty Cycle Range Limitation for Three EPWMCLK/TBCLK Cycles**

| PWM Frequency [1] (kHz) | 3 Cycles Minimum Duty | 3 Cycles Maximum Duty[2] |
|---|---|---|
| 200 | 0.6% | 99.4% |
| 400 | 1.2% | 98.8% |
| 600 | 1.8% | 98.2% |
| 800 | 2.4% | 97.6% |
| 1000 | 3% | 97% |
| 1200 | 3.6% | 96.4% |
| 1400 | 4.2% | 95.8% |
| 1600 | 4.8% | 95.2% |
| 1800 | 5.4% | 94.6% |
| 2000 | 6% | 94% |

(1) EPWMCLK = TBCLK = 100 MHz

(2) This limitation applies only if high-resolution period (TBPRDHR) control is enabled.

If the application demands HRPWM operation below the minimum duty cycle limitation, then the HRPWM can be configured to operate in count-down mode with the rising edge position (REP) controlled by the MEP when high-resolution period is disabled (HRPCTL[HRPE] = 0). This is illustrated in Figure 19-8. In this configuration, the minimum duty cycle limitation is no longer an issue. However, there will be a maximum duty limitation with same percent numbers as given in Table 19-4.

**Figure 19-8. High % Duty Cycle Range Limitation Example (HRPCTL[HRPE] = 0)**

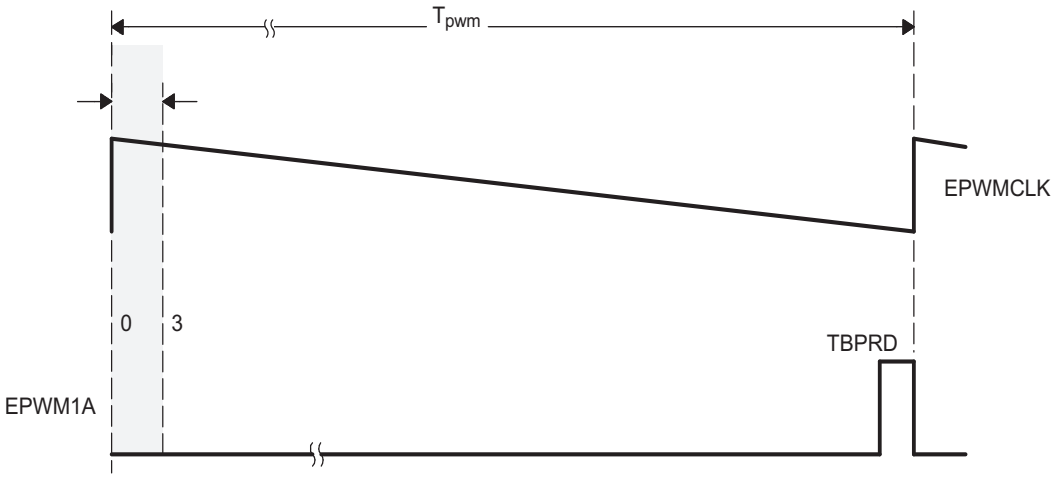



**Figure 19-9. Up-Count Duty Cycle Range Limitation Example (HRPCTL[HRPE]=1)**

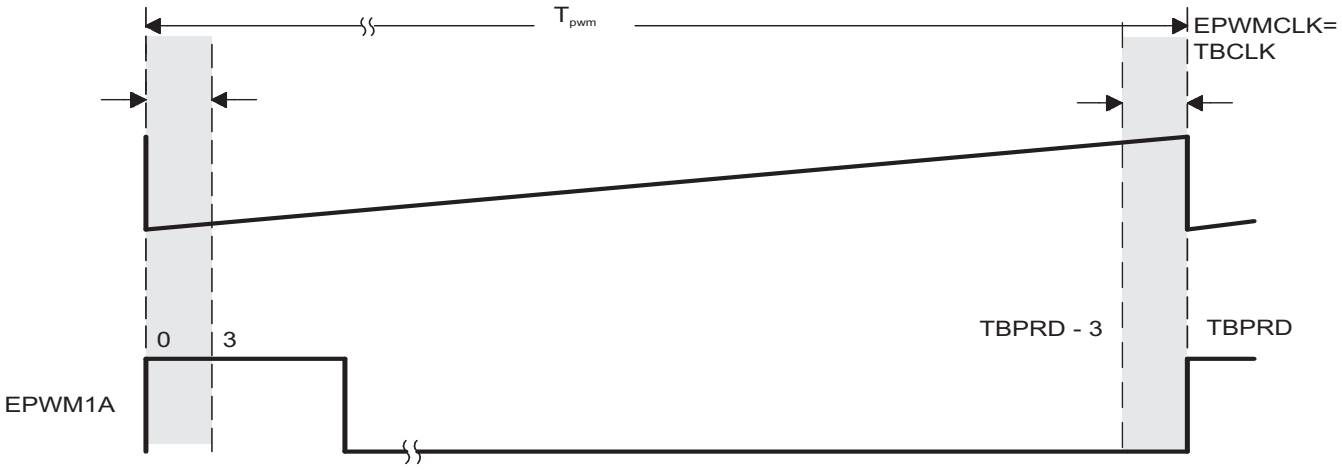
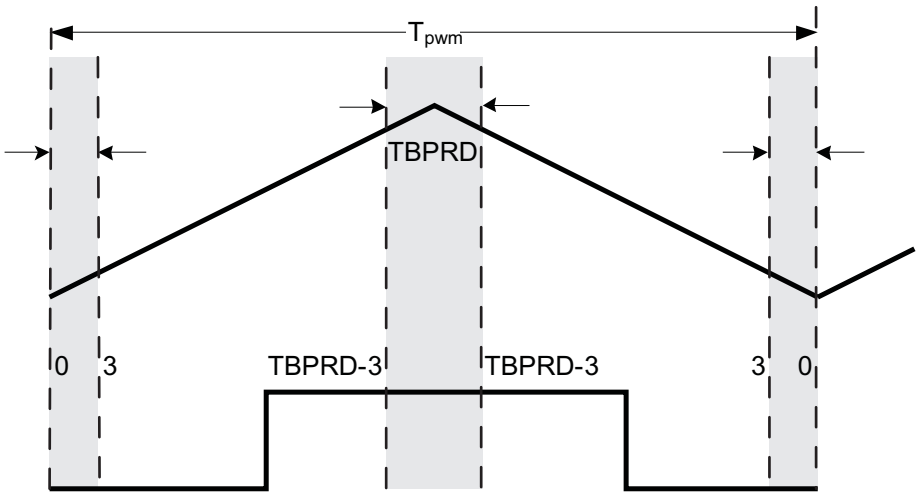


**Figure 19-10. Up-Down Count Duty Cycle Range Limitation Example (HRPCTL[HRPE]=1)**




**NOTE:** If the application has enabled high-resolution period control (HRPCTL[HRPE]=1), the duty cycle must not fall within the restricted range. Otherwise, there will be undefined behavior on the ePWM output.

#### 19.2.4.4 High Resolution Period

High resolution period control using the MEP logic is supported on devices with a Type 1 ePWM module or greater.

**NOTE:** When high-resolution period control is enabled, on ePWMxA only, and not ePWMxB output and vice versa, the non hi-res output will have +/- 1 TBCLK cycle jitter in up-count mode and +/- 2 TBCLK cycle jitter in up-down count mode.

The scaling procedure described for duty cycle in Section 19.2.4.2 applies for high-resolution period as well:

**Assumptions for this example:**

| | |
|---|---|
| TBCLK | = 10 ns (100 MHz) |
| Required PWM frequency | = 175 kHz (period of 571.428) |
| Number of MEP steps per coarse step at 180 ps (MEP_ScaleFactor) | = 55 (10 ns / 180 ps) |
| Value to keep TBPRDHR within range of 1-255 and fractional rounding constant (default value) | = 0.5 (0080h in Q8 format) |

**Problem:**

In up-count mode:

If TBPRD = 571, then PWM frequency = 174.82 kHz (period = (571+1)* $T_{TBCLK}$).

If TBPRD = 570, then PWM frequency = 175.13 kHz (period = (570+1)* $T_{TBCLK}$).

In up-down count mode:

If TBPRD = 286, then PWM frequency = 174.82 kHz (period = (286*2)* $T_{TBCLK}$).

If TBPRD = 285, then PWM frequency = 175.44 kHz (period = (285*2)* $T_{TBCLK}$).

**Solution:**

With 55 MEP steps per coarse step at 180 ps each:

**Step 1: Percentage Integer Period value conversion for TBPRD register**

| | |
|---|---|
| Integer period value | = 571 * $T_{TBCLK}$<br>= int (571.428) * $T_{TBCLK}$<br>= int (PWMperiod) * $T_{TBCLK}$ |
| In up-count mode: | |
| TBPRD | = 570 (TBPRD = period value - 1)<br>= 023Ah |
| In up-down count mode: | = 285 (TBPRD = period value / 2) |
| TBPRD | = 011Dh |

**Step 2: Fractional value conversion for TBPRDHR register**

| | |
|---|---|
| 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<br>=0.428 × 256<br>=6D00h |

| | |
|---|---|
| The autoconversion will then automatically perform 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 |

#### 19.2.4.4.1 High-Resolution Period Configuration

To use High Resolution Period, the ePWMx module must be initialized in the exact order presented.

The steps below use CMPA with shadow registers and the corresponding HRCNFG bits for hi-resolution operation on EPWMxA. For hi-resolution operation on EPWMxB, make the appropriate substitutions with the B channel fields.

1. Enable ePWMx clock
2. Enable HRPWM clock
3. Disable TBCLKSYNC
4. Configure ePWMx registers - AQ, TBPRD, CC, and so on.
   - ePWMx may only be configured for up-count or up-down count modes. High-resolution period is not compatible with down-count mode.
   - TBPRD and CC registers must be configured for shadow loads.
   - CMPCTL[LOADAMODE]
     - In up-count mode: CMPCTL[LOADAMODE] = 1 (load on CTR = PRD)
     - In up-down count mode: CMPCTL[LOADAMODE] = 2 (load on CTR=0 or CTR=PRD)
5. Configure the HRCNFG register such that:
   - HRCNFG[HRLOAD] = 2 (load on either CTR = 0 or CTR = PRD)
   - HRCNFG[AUTOCONV] = 1 (Enable auto-conversion)
   - HRCNFG[EDGMODE] = 3 (MEP control on both edges)
6. For TBPHS:TBPHSHR synchronization with high-resolution period, set both HRPCTL[TBPSHRLOADE] = 1 and TBCTL[PHSEN] = 1. In up-down count mode these bits must be set to 1 regardless of the contents of TBPHSHR.
7. Enable high-resolution period control (HRPCTL[HRPE] = 1)
8. Enable TBCLKSYNC
9. TBCTL[SWFSYNC] = 1
10. HRMSTEP must contain an accurate MEP scale factor (# of MEP steps per EPWMCLK coarse step) because auto-conversion is enabled. The MEP scale factor can be acquired via the SFO() function described in Section 19.3.
11. To control high-resolution period, write to the TBPRDHR(M) registers.

**NOTE:** When high-resolution period mode is enabled, an EPWMxSYNC pulse will introduce +/- 1 - 2 cycle jitter to the PWM (+/- 1 cycle in up-count mode and +/- 2 cycle in up-down count mode). For this reason, TBCTL[SYNCOSEL] should not be set to 1 (CTR = 0 is EPWMxSYNCO source) or 2 (CTR = CMPB is EPWMxSYNCO source). Otherwise, the jitter will occur on every PWM cycle with the synchronization pulse.

When TBCTL[SYNCOSEL] = 0 (EPWMxSYNCI is EPWMxSYNCO source), a software synchronization pulse should be issued only once during high-resolution period initialization. If a software sync pulse is applied while the PWM is running, the jitter will appear on the PWM output at the time of the sync pulse.

### 19.2.5 Deadband High Resolution Operation

**Assumptions for this example:**

| | |
|---|---|
| System clock | = 10 ns (100 MHz) & |
| Deadband Enabled in half cycle mode, TBCLK = EPWMCLK | |
| Required PWM frequency | 1.33MHz (1 / 750 ns) |
| Required PWM duty cycle | 0.5 (50%) |
| Required Dead band Rising Edge Delay | 5% over duty |
| Required Dead band Rising Edge Delay in ns | (0.05 * 375 ns ) = 18.75 ns |

**NOTE:** Just like the duty cycle restrictions when using HRPWM, the DBRED and DBFED values must be greater than 3 to use hi-res deadband.

**Deadband Delay Values as a Function of DBFED and DBRED:**

When half-cycle clocking is enabled, the formula to calculate the falling-edge-delay and rising-edge-delay becomes:

FED = DBFED * TBCLK / 2

RED = DBRED * TBCLK / 2

**DBRED and DBFED Calculated Values:**

Required Dead band Rising Edge Delay in ns = 18.75 ns

DBRED = RED / (TBCLK / 2)

DBRED = 18.75 ns / 5 ns

DBRED Required = 3.75 ns

With 55 MEP steps per coarse step at 180 ps each:

**Step 1: Integer Dead band value conversion for DBREDM register**

| | |
|---|---|
| Integer DBRED value | = int (RED / (TBCLK / 2))<br>= int (3.75) |
| DBRED | = 3 |

**Step 2: Fractional value conversion for Dead band high resolution register DBREDHR**

| | |
|---|---|
| DBREDHR register value | = (frac(DBRED Required) * MEP_ScaleFactor + 0.5) << 8 (Shifting is to move the value to the high byte of DBREDHR)<br>=(frac (3.75) * 55 + 0.5) << 8<br>= ( 0.75 * 55 + 0.5 ) << 8<br>=(41.75) * 256 Shifting left by 8 is the same as multiplying by 256. |
| DBREDHR value | =29C0h MEP Steps<br>Hardware will ignore lower 9 bits in the above calculated DBREDHR value |

**NOTE:** If the AUTOCONV bit (HRCNFG.6) is set and the MEP_ScaleFactor is in the HRMSTEP register, then DBREDHR:DBRED = 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.

### 19.2.6 Scale Factor Optimizing Software (SFO)

The micro edge positioner (MEP) logic is capable of placing an edge in one of 255 discrete time steps. As previously mentioned, the size of these steps is on the order of 150 ps (see device-specific data sheet for typical MEP step size on your device). The MEP step size varies based on worst-case process parameters, operating temperature, and voltage. MEP step size increases with decreasing voltage and increasing temperature and decreases with increasing voltage and decreasing temperature. Applications that use the HRPWM feature should use the TI-supplied MEP scale factor optimization (SFO) software function. The SFO function helps to dynamically determine the number of MEP steps per EPWMCLK period while the HRPWM is in operation.

To utilize the MEP capabilities effectively, the correct value for the MEP scaling factor needs to be known by the software. To accomplish this, the HRPWM module has built in self-check and diagnostic capabilities that can be used to determine the optimum MEP scale factor value for any operating condition. TI provides a C-callable library containing one SFO function that utilizes this hardware and determines the optimum MEP scale factor. As such, MEP control and diagnostics registers are reserved for TI use.

A detailed description of the SFO library - SFO_TI_Build_V7.lib software can be found in Section 19.3.

### 19.2.7 HRPWM Examples Using Optimized Assembly Code.

The best way to understand how to use the HRPWM capabilities is through two real examples:

1. Simple buck converter using asymmetrical PWM (count-up) with active high polarity.
2. DAC function using simple R+C reconstruction filter.

The following examples all have initialization and configuration code written in C. To make these easier to understand, the #defines shown below are used. Note, #defines introduced in the device-specific *Pulse Width Modulator (ePWM) Module Reference Guide* are also used.

Example 19-1 This example assumes MEP step size of 150 ps and does not use the SFO library.

***Example 19-1. #Defines for HRPWM Header Files***

```
// HRPWM (High Resolution PWM) //
================================
// HRCNFG
#define HR_Disable 0x0
#define HR_REP 0x1          // Rising Edge position
#define HR_FEP 0x2          // Falling Edge position
#define HR_BEP 0x3          // Both Edge position #define HR_CMP 0x0 // CMPAHR controlled
#define HR_PHS 0x1          // TBPHSHR controlled #define HR_CTR_ZERO 0x0 // CTR = Zero event
#define HR_CTR_PRD 0x1      // CTR = Period event
#define HR_CTR_ZERO_PRD 0x2 // CTR = ZERO or Period event
#define HR_NORM_B  0x0      // Normal ePWMxB output
#define HR_INVERT_B 0x1     // ePWMxB is inverted ePWMxA output
```

### 19.2.7.1 Implementing a Simple Buck Converter

In this example, the PWM requirements are:

- PWM frequency = 1 MHz (that is, TBPRD = 100 )
- PWM mode = asymmetrical, up-count
- Resolution = 12.7 bits (with a MEP step size of 150 ps)

Figure 19-11 and Figure 19-12 show the required PWM waveform. As explained previously, configuration for the ePWM1 module is almost identical to the normal case except that the appropriate MEP options need to be enabled/selected.

**Figure 19-11. Simple Buck Controlled Converter Using a Single PWM**

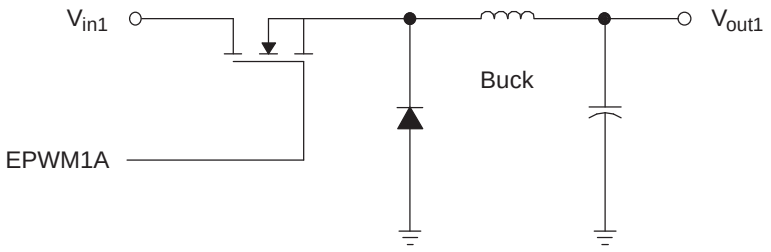
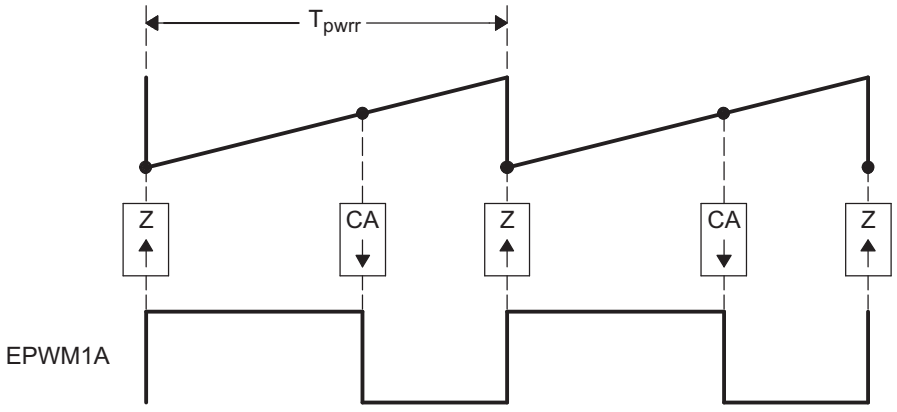



**Figure 19-12. PWM Waveform Generated for Simple Buck Controlled Converter**



The example code shown consists of two main parts:

- Initialization code (executed once)
- Run time code (typically executed within an ISR)

Example 19-2 shows the Initialization code. The first part is configured for conventional PWM. The second part sets up the HRPWM resources.

This example assumes MEP step size of 150 ps and does not use the SFO library.

***Example 19-2. 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 1000 kHz PWM
hrbuck_period = 200;                              // Used for Q15 to Q0 scaling
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;           // EPWM1 is the Master
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
// Note: ChB is initialized here only for comparison purposes, it is not required

EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;     // optional
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;       // optional

EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;
EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.ZRO = AQ_SET;                // optional
EPwm1Regs.AQCTLB.bit.CBU = AQ_CLEAR;              // optional
// Now configure the HRPWM resources
EALLOW;                                           // Note these registers are protected
                                                  // and act only on ChA
EPwm1Regs.HRCNFG.all = 0x0;                       // clear all bits first
EPwm1Regs.HRCNFG.bit.EDGMODE = HR_FEP;            // Control Falling Edge Position
EPwm1Regs.HRCNFG.bit.CTLMODE = HR_CMP;            // CMPAHR controls the MEP
EPwm1Regs.HRCNFG.bit.HRLOAD = HR_CTR_ZERO;        // Shadow load on CTR=Zero
EDIS;
MEP_ScaleFactor = 66*256;                        // Start with typical Scale Factor
                                                  // value for 100 MHz
                                                  // Note: Use SFO functions to update
                                                     MEP_ScaleFactor dynamically
}
```

Example 19-3 shows an assembly example of run-time code for the HRPWM buck converter.

***Example 19-3. HRPWM Buck Converter Run-Time Code***

```
EPWM1_BASE .set 0x6800
CMPAHR1 .set EPWM1_BASE+0x8
;=============================================
HRBUCK_DRV; (can execute within an ISR or loop)
;=============================================
     MOVW DP, #_HRBUCK_In
     MOVL XAR2,@_HRBUCK_In      ; Pointer to Input Q15 Duty (XAR2)
     MOVL XAR3,#CMPAHR1         ; Pointer to HRPWM CMPA reg (XAR3)
; Output for EPWM1A (HRPWM)
     MOV T,*XAR2 ; T <= Duty
     MPYU ACC,T,@_hrbuck_period ; Q15 to Q0 scaling based on Period
     MOV T,@_MEP_ScaleFactor    ; MEP scale factor (from optimizer s/w)
     MPYU P,T,@AL               ; P <= T * AL, Optimizer scaling
     MOVH @AL,P                 ; AL <= P, move result back to ACC
     ADD ACC, #0x080            ; MEP range and rounding adjustment
     MOVL *XAR3,ACC             ; CMPA:CMPAHR(31:8) <= ACC
; Output for EPWM1B (Regular Res) Optional - for comparison purpose only
     MOV *+XAR3[2],AH           ; Store ACCH to regular CMPB
```

### 19.2.7.2 Implementing a DAC function Using an R+C Reconstruction Filter

In this example, the PWM requirements are:

- PWM frequency = 400 kHz (that is, TBPRD = 250)
- PWM mode = Asymmetrical, Up-count
- Resolution = 14 bits ( MEP step size = 150 ps)

Figure 19-13 and Figure 19-14 show the DAC function and the required PWM waveform. As explained previously, configuration for the ePWM1 module is almost identical to the normal case except that the appropriate MEP options need to be enabled/selected.

**Figure 19-13. Simple Reconstruction Filter for a PWM-based DAC**

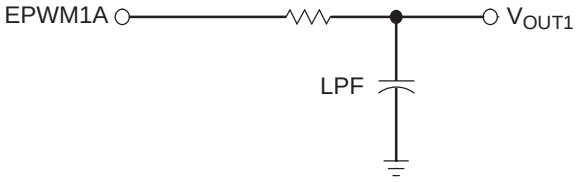
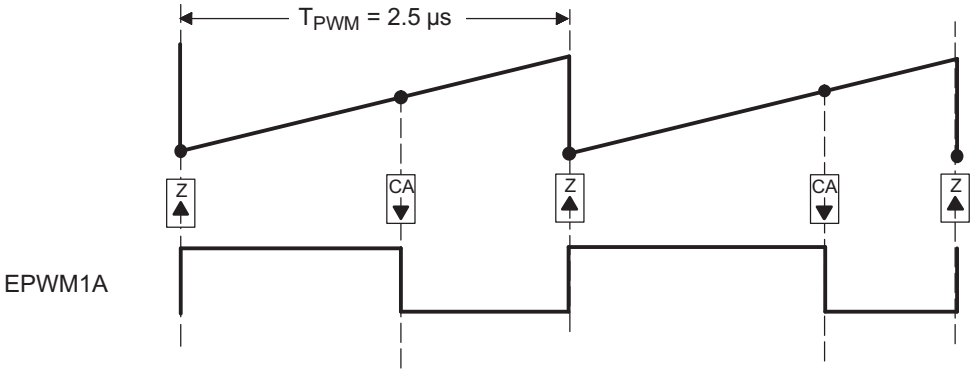



**Figure 19-14. PWM Waveform Generated for the PWM DAC Function**




The example code shown consists of two main parts:

- Initialization code (executed once)
- Run time code (typically executed within an ISR)

This example assumes a typical MEP_SP and does not use the SFO library.

Example 19-4 shows the Initialization code. The first part is configured for conventional PWM. The second part sets up the HRPWM resources.

***Example 19-4. 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 400 kHz PWM
hrDAC_period = 250;                            // Used for Q15 to Q0 scaling
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;        // EPWM1 is the Master
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
// Note: ChB is initialized here only for comparison purposes, it is not required

EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;  // optional
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;    // optional

EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;
EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.ZRO = AQ_SET;             // optional
EPwm1Regs.AQCTLB.bit.CBU = AQ_CLEAR;           // optional
// Now configure the HRPWM resources
EALLOW;                                        // Note these registers are protected
                                               // and act only on ChA.
EPwm1Regs.HRCNFG.all = 0x0; // Clear all bits first
EPwm1Regs.HRCNFG.bit.EDGMODE = HR_FEP;         // Control falling edge position
EPwm1Regs.HRCNFG.bit.CTLMODE = HR_CMP;         // CMPAHR controls the MEP.
EPwm1Regs.HRCNFG.bit.HRLOAD = HR_CTR_ZERO;     // Shadow load on CTR=Zero.
EDIS;
MEP_ScaleFactor = 66*256;                     // Start with typical Scale Factor
                                               // value for 100 MHz.
                                               // Use SFO functions to update MEP_ScaleFactor
                                               // dynamically.
}
```

Example 19-5 shows an assembly example of run-time code that can execute in a high-speed ISR loop.

### Example 19-5. PWM DAC Function Run-Time Code

```
EPWM1_BASE .set 0x6800
CMPAHR1 .set EPWM1_BASE+0x8
;================================================
HRPWM_DAC_DRV; (can execute within an ISR or loop)
;================================================
      MOVW DP, #_HRDAC_In
      MOVL XAR2,@_HRDAC_In            ; Pointer to input Q15 duty (XAR2)
      MOVL XAR3,#CMPAHR1              ; Pointer to HRPWM CMPA reg (XAR3)

; Output for EPWM1A (HRPWM
      MOV T,*XAR2                      ; T <= duty
      MPY ACC,T,@_hrDAC_period         ; Q15 to Q0 scaling based on period
      ADD ACC,@_HrDAC_period<<15 ; Offset for bipolar operation
      MOV T,@_MEP_ScaleFactor          ; MEP scale factor (from optimizer s/w)
      MPYU P,T,@AL                     ; P <= T * AL, optimizer scaling
      MOVH @AL,P                       ; AL <= P, move result back to ACC
      ADD ACC, #0x080                  ; MEP range and rounding adjustment
      MOVL *XAR3,ACC                   ; CMPA:CMPAHR(31:8) <= ACC

; Output for EPWM1B (Regular Res) Optional - for comparison purpose only
      MOV *+XAR3[2],AH                 ; Store ACCH to regular CMPB
```

## 19.3 Appendix A: SFO Library Software - SFO_TI_Build_V7.lib

The following table lists several features of the SFO_TI_Build_V7.lib library.

**Table 19-5. SFO Library Features**

| | SFO_TI_Build_V7.lib | Unit |
|---|---|---|
| Completion-checking? | Yes | Function return value |
| Typical cycles required for SFO() to update MEP_ScaleFactor if called repetitively without interrupts | 130,000 | EPWMCLK cycles |

A functional description of the SFO library routine, SFO(), is found below.

### *19.3.1 Scale Factor Optimizer Function - int SFO()*

This routine drives the micro-edge positioner (MEP) calibration module to run SFO diagnostics and determine the appropriate MEP scale factor (number of MEP steps per coarse EPWMCLK step) for a device at any given time.

If EPWMCLK = TBCLK = 100 MHz and assuming the MEP step size is 150 ps, the typical scale factor value at 100 MHz = 66 MEP steps per TBCLK unit (10 ns)

The function returns a MEP scale factor value:

MEP_ScaleFactor = Number of MEP steps per EPWMCLK.

**Constraints when using this function:**

- SFO() can be used with a minimum EPWMCLK = TBCLK = 50 MHz. MEP diagnostics logic uses EPWMCLK and not TBCLK, so the EPWMCLK restriction is an important constraint. Below 50 MHz, with device process variation, the MEP step size may decrease under cold temperature and high core voltage conditions to such a point, that 255 MEP steps will not span an entire EPWMCLK cycle.
- At any time, SFO() can be called to run SFO diagnostics on the MEP calibration module

**Usage:**

- SFO() can be called at any time in the background while the ePWM channels are running in HRPWM mode. The scale factor result obtained can be applied to all ePWM channels running in HRPWM mode because the function makes use of the diagnostics logic in the MEP calibration module (which runs independently of ePWM channels).
- This routine returns a 1 when calibration is finished and a new scale factor has been calculated, or a 0 if calibration is still running. The routine returns a 2 if there is an error, and the MEP_ScaleFactor is greater than the maximum 255 fine steps per coarse EPWMCLK cycle. In this case, the HRMSTEP register will maintain the last MEP scale factor value less than 256 for auto conversion.
- All ePWM modules operating in HRPWM incur only a 3-EPWMCLK cycle minimum duty cycle limitation when high-resolution period control is not used. If high-resolution period control is enabled, there is an additional duty cycle limitation 3-EPWMCLK cycles before the end of the PWM period (see Section 19.2.4.3).
- The SFO() function also updates the HRMSTEP register with the scale factor result. If the HRCNFG[AUTOCONV] bit is set, the application software is responsible only for setting CMPAHR = fraction(PWMduty*PWMperiod) << 8 or CMPBHR = fraction(PWMduty*PWMperiod) << 8 or TBPRDHR = fraction (PWMperiod) while running SFO() in the background. The MEP Calibration Module will then use the values in the HRMSTEP and CMPAHR/CMPBHR/TBPRDHR register to automatically calculate the appropriate number of MEP steps represented by the fractional duty cycle or period and move the high-resolution ePWM signal edge accordingly.
- If the HRCNFG[AUTOCONV] bit is clear, the HRMSTEP register is ignored. The application software will need to perform the necessary calculations manually so that:
  - CMPAHR = (fraction(PWMduty * PWMperiod) * MEP Scale Factor) << 8 + 0x080.
  - Similar behavior applies for TBPHSHR, CMPBHR, DBREDHR, DBFEDHR. Auto-conversion must be enabled when using TBPRDHR.

The routine can be run as a background task in a slow loop requiring negligible CPU cycles. The repetition rate at which an SFO function needs to be executed depends on the application's operating environment. As with all digital CMOS devices, temperature and supply voltage variations have an effect on MEP operation. However, in most applications these parameters vary slowly and therefore it is often sufficient to execute the SFO function once every 5 to 10 seconds. If more rapid variations are expected, then execution may have to be performed more frequently to match the application. Note, there is no high limit restriction on the SFO function repetition rate, hence it can execute as quickly as the background loop is capable.

While using the HRPWM feature, HRPWM logic will not be active for the first three EPWMCLK cycles of the PWM period (and the last three EPWMCLK cycles of the PWM period if TBPRDHR is used). While running the application in this configuration, if high-resolution period control is disabled (HRPCTL[HRPE=0]) and the CMPA/CMPB register value is less than three cycles, then its CMPAHR/CMPBHR register must be cleared to zero. If high-resolution period control is enabled (HRPCTL[HRPE=1]), the CMPA register value must not fall below three or above TBPRD-3.This would avoid any unexpected transitions on the PWM signal.

### 19.3.2 Software Usage

The software library function SFO(), calculates the MEP scale factor for the HRPWM-supported ePWM modules. The scale factor is an integer value in the range 1-255, and represents the number of micro step edge positions available for a system clock period. The scale factor value is returned in an integer variable called MEP_ScaleFactor. For example, see Table 19-6.

**Table 19-6. 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_V7.h file needs to be included as follows. This include file is mandatory while using the SFO library function. For the SFO() to operate, the appropriate (Device)_Device.h and (Device)_Epwm_defines.h must be included in the project. These include files are optional if customized header files are used in the end applications.

***Example 19-6. A Sample of How to Add "Include" Files***

```
#include "F28x7x_Device.h"       // F28x7x Headerfile
#include "F28x7x_EPwm_defines.h" // init defines
#include "SFO_V7.h"                // SFO lib functions (needed for HRPWM)
```

**Step 2. Element Declaration**

Declare an integer variable for the scale factor value as shown below.

***Example 19-7. Declaring an Element***

```
int MEP_ScaleFactor = 0;   //scale factor value
volatile struct EPWM_REGS *ePWM[] = {0, &EPwm1Regs, &EPwm2Regs, &EPwm3Regs,
&EPwm4Regs};
```

#### Step 3. MEP_ScaleFactor Initialization

The SFO() function does not require a starting scale factor value in MEP_ScaleFactor. Prior to using the MEP_ScaleFactor variable in application code, SFO() should be called to drive the MEP calibration module to calculate an MEP_ScaleFactor value.

As part of the one-time initialization code prior to using MEP_ScaleFactor, include the following:

***Example 19-8. Initializing With a Scale Factor Value***

```
MEP_ScaleFactor initialized using function SFO ()
while (SFO() == 0) {} // MEP_ScaleFactor calculated by MEP Cal Module
```

#### Step 4. Application Code

While the application is running, fluctuations in both device temperature and supply voltage may be expected. To be sure that optimal Scale Factors are used for each ePWM module, the SFO function should be re-run periodically as part of a slower back-ground loop. Some examples of this are shown here.

**NOTE:** See the HRPWM_SFO example in the device-specific C/C++ header files and peripheral examples available from the TI website.

***Example 19-9. SFO Function Calls***

```
main ()
 {
 int status;
    // User code
    // ePWM1, 2, 3, 4 are running in HRPWM mode
    // The status variable returns 1 once a new MEP_ScaleFactor has been
    // calculated by the MEP Calibration Module running SFO
    // diagnostics.

status = SFO();
if(status==2) {ESTOP0;}   // The function returns a 2 if MEP_ScaleFactor is greater
                          // than the maximum 255 allowed (error condition)
 }
```

# Enhanced Capture (eCAP)

This chapter describes the enhanced capture (eCAP), which is used in systems where accurate timing of external events is important.

The eCAP module described in this reference guide is a Type 1 eCAP. See the *TMS320C28xx, 28xxx DSP Peripheral Reference Guide* (SPRU566) for a list of all devices with a eCAP module of the same type, to determine the differences between the types, and for a list of device-specific differences within a type.

**Topic** **Page**

## 20.1 Introduction

Features for eCAP include:

- Speed measurements of rotating machinery (for example, toothed sprockets sensed via Hall sensors)
- Elapsed time measurements between position sensor pulses
- Period and duty cycle measurements of pulse train signals
- Decoding current or voltage amplitude derived from duty cycle encoded current/voltage sensors

The eCAP module described in this guide includes the following features:

- 4-event time-stamp registers (each 32 bits)
- Edge polarity selection for up to four sequenced time-stamp capture events
- Interrupt on either of the four events
- Single shot capture of up to four event time-stamps
- Continuous mode capture of time-stamps in a four-deep circular buffer
- Absolute time-stamp capture
- Difference (Delta) mode time-stamp capture
- When not used in capture mode, the ECAP module can be configured as a single channel PWM output

The capture functionality of the Type-1 eCAP is enhanced from the Type 0 eCAP with the following added features:

- Event filter reset bit
  - Writing a 1 to ECCTL2[CTRFILTRESET] will clear the event filter, the modulo counter, and any pending interrupts flags. This is useful for initialization and debug.
- Modulo counter status bits
  - The modulo counter (ECCTL2 [MODCTRSTS]) indicates which capture register will be loaded next. In the Type-0 ECAP it was not possible to know current state of modulo counter.
- DMA trigger source
  - eCAPxDMA was added as a DMA trigger. CEVT[1-4] can be configured as the source for eCAPxDMA.
- Input multiplexer
  - ECCTL0 [INPUTSEL] selects one of 128 input signals which are detailed in Table 20-1.
- EALLOW protection
  - EALLOW protection was added to critical registers.

## 20.2 Description

The eCAP module represents one complete capture channel that can be instantiated multiple times depending on the target device. In the context of this guide, one eCAP channel has the following independent key resources:

- Capture inputs can be connected using the Input X-BAR
- 128:1 input multiplexer
- Output X-BAR is used to configure output in APWM mode
- 32-bit time base (counter)
- 4 x 32-bit time-stamp capture registers (CAP1-CAP4)
- Four-stage sequencer (Modulo4 counter) that is synchronized to external events, ECAP pin rising/falling edges.
- Modulo counter status register (MODCNTRSTS) to indicate sequencer state
- Independent edge polarity (rising/falling edge) selection for all four events
- Input capture signal prescaling (from 2-62 or bypass)
- One-shot compare register (two bits) to freeze captures after 1-4 time-stamp events
- Control for continuous time-stamp captures using a four-deep circular buffer (CAP1-CAP4) scheme

- Ability to reset event filter, modulo counter, and interrupt flags
- Interrupt capabilities on any of the four capture events
- Separate DMA trigger
- EALLOW protection to control registers

## 20.3 Configuring Device Pins for the eCAP

To connect the device input pins to the module, the Input X-BAR must be used. Any GPIO on the device can be configured as an input. The GPIO input qualification should be set to asynchronous mode by setting the appropriate GPxQSELn register bits to 11b. The internal pull-ups can be configured in the GPyPUD register. Since the GPIO mode is used the GPyINV register can invert the signals.

New to the Type 1 eCAP module, a 128:1 input multiplexer must also be configured (see Figure 20-3). This multiplexer can select a variety of inputs detailed in Table 20-1.

**Table 20-1. eCAP Input Selection**

| Selection of eCAP Input | Select Index |
|---|---|
| Reserved | 127 |
| CMPSS[7:1]. CTRIPH_OR_CTRIPL | 126:120 |
| Reserved | 119:115 |
| CMPSS[7:1].CTRIPH | 114:108 |
| Reserved | 107:103 |
| CMPSS[7:1].CTRIPL | 102:96 |
| Reserved | 95:92 |
| SD1FLT[4:1].COMPH_OR_COMPL | 91:88 |
| Reserved | 87:84 |
| SD1FLT[4:1].COMPH | 83:80 |
| Reserved | 79:76 |
| SD1FLT[4:1].COMPZ | 75:72 |
| Reserved | 71:68 |
| SD1FLT[4:1].COMPL | 67:64 |
| Reserved | 63:51 |
| Reserved | 50 |
| Reserved | 49 |
| Reserved | 48 |
| ADCAEVT[4:1] | 47:44 |
| ADCBEVT[4:1] | 43:40 |
| ADCCEVT[4:1] | 39:36 |
| Reserved | 35:32 |
| Output X-BAR OUTPUT[8:1] | 31:24 |
| Reserved | 23:22 |
| CANB_0 interrupt | 21 |
| CANA_0 interrupt | 20 |
| Reserved | 19:16 |
| Input X-BAR INPUT[16:1] | 15:0 |

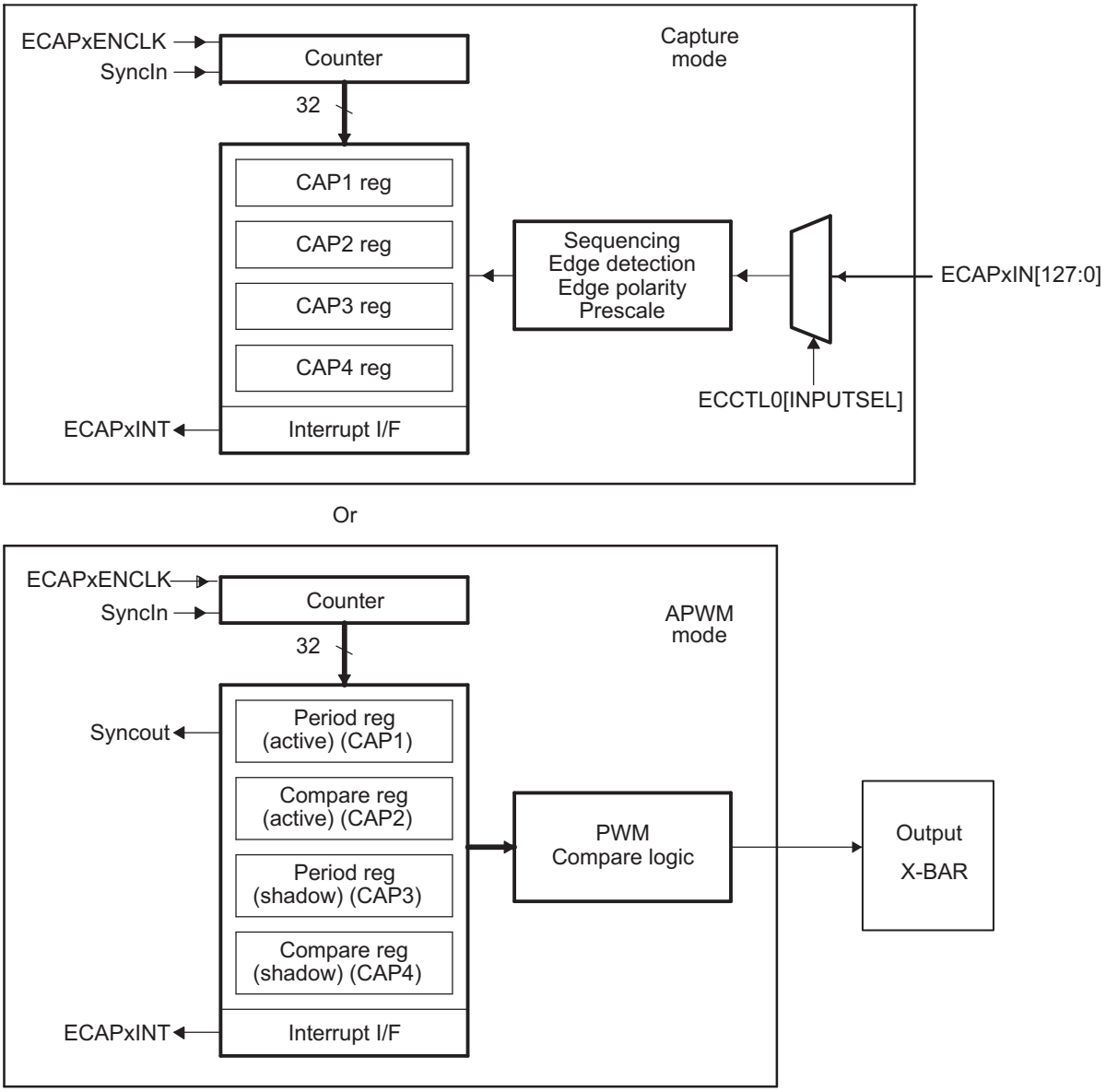
The Output X-BAR must be used to connect output signals to the OUTPUTXBARx output locations. The GPIO mux then be configured to connect the OUTPUTXBARx lines to any of several IO pins with the GPIO Mux. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

See the *GPIO Chapter* for more details on GPIO mux, GPIO settings, and XBAR configuration.

## 20.4 Capture and APWM Operating Mode

You can use the eCAP module resources to implement a single-channel PWM generator (with 32-bit capabilities) when it is not being used for input captures. The counter operates in count-up mode, providing a time-base for asymmetrical pulse width modulation (PWM) waveforms. The CAP1 and CAP2 registers become the active period and compare registers, respectively, while CAP3 and CAP4 registers become the period and capture shadow registers, respectively. Figure 20-1 is a high-level view of both the capture and auxiliary pulse-width modulator (APWM) modes of operation.

**Figure 20-1. Capture and APWM Modes of Operation**

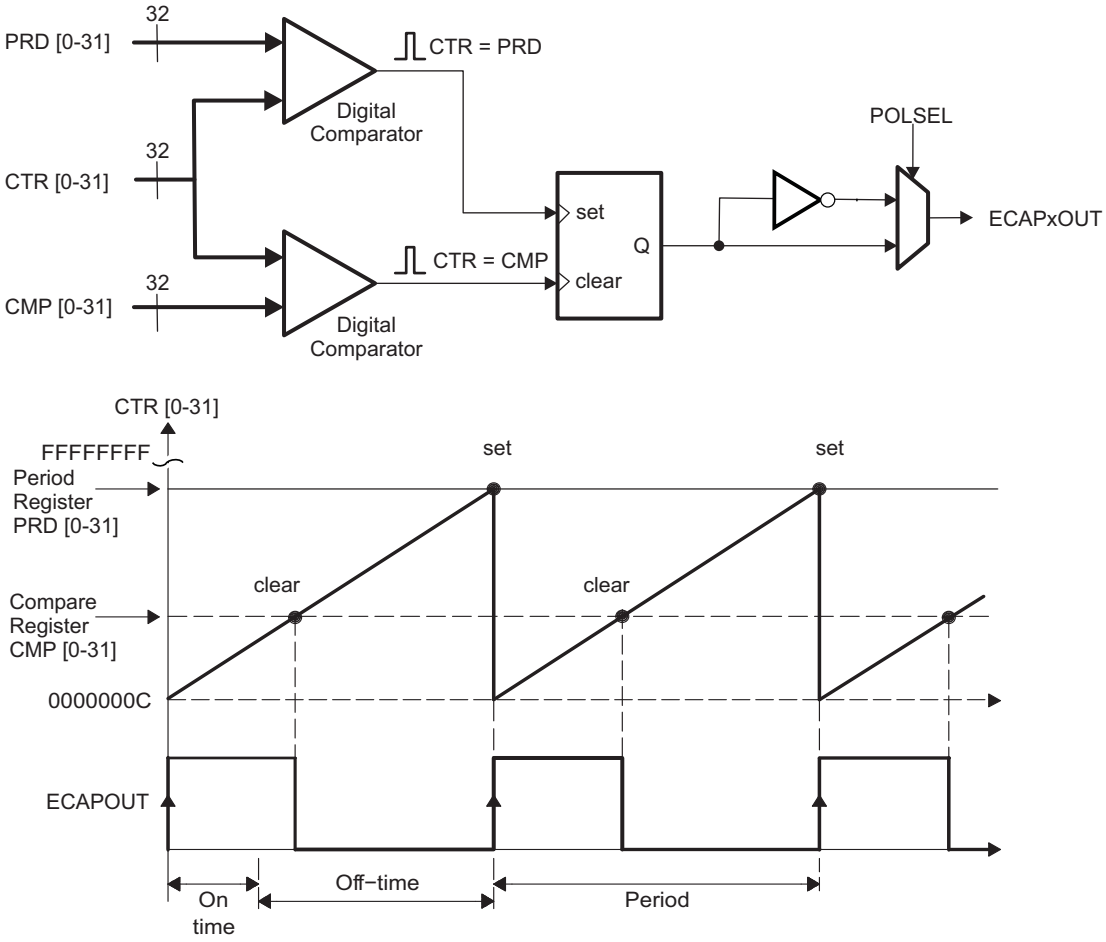



A A single pin is shared between CAP and APWM functions. In capture mode, it is an input; in APWM mode, it is an output.

B In APWM mode, writing any value to CAP1/CAP2 active registers also writes the same value to the corresponding shadow registers CAP3/CAP4. This emulates immediate mode. Writing to the shadow registers CAP3/CAP4 invokes the shadow mode.

Figure 20-2 further descries the output of the eCAP in APWM mode based on the CMP and PRD values.

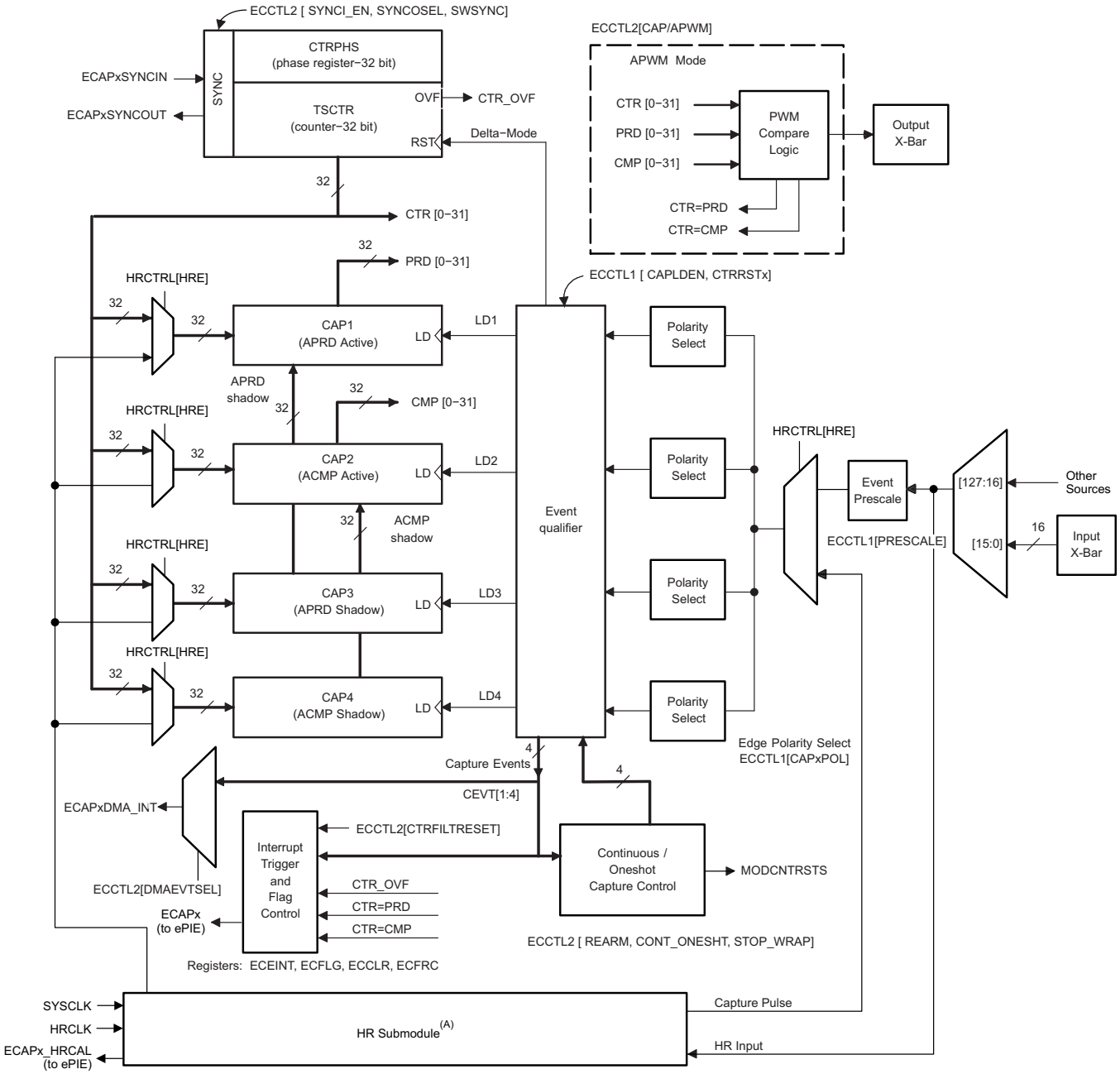
**Figure 20-2. Counter Compare and PRD Effects on the eCAP Output in APWM Mode**




## 20.5 Capture Mode Description

Figure 20-3 shows the various components that implement the capture function.

**Figure 20-3. eCAP Block Diagram**



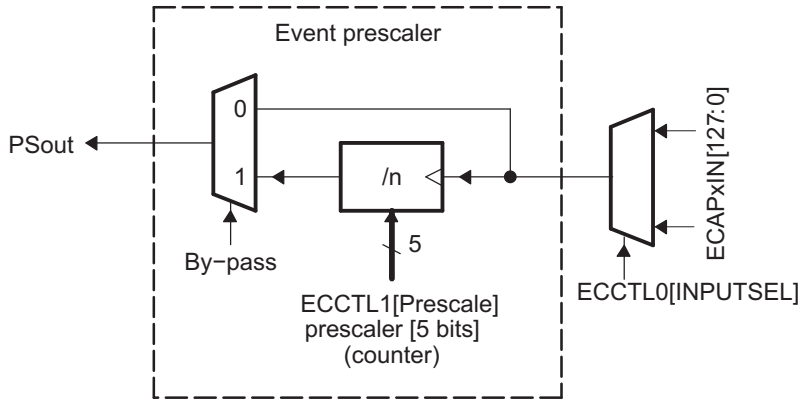

A The HRCAP submodule is available on all eCAP modules; in this case, the high-resolution muxes and hardware are not implemented.

### 20.5.1 Event Prescaler

- An input capture signal (pulse train) can be prescaled by N = 2-62 (in multiples of 2) or can bypass the prescaler.

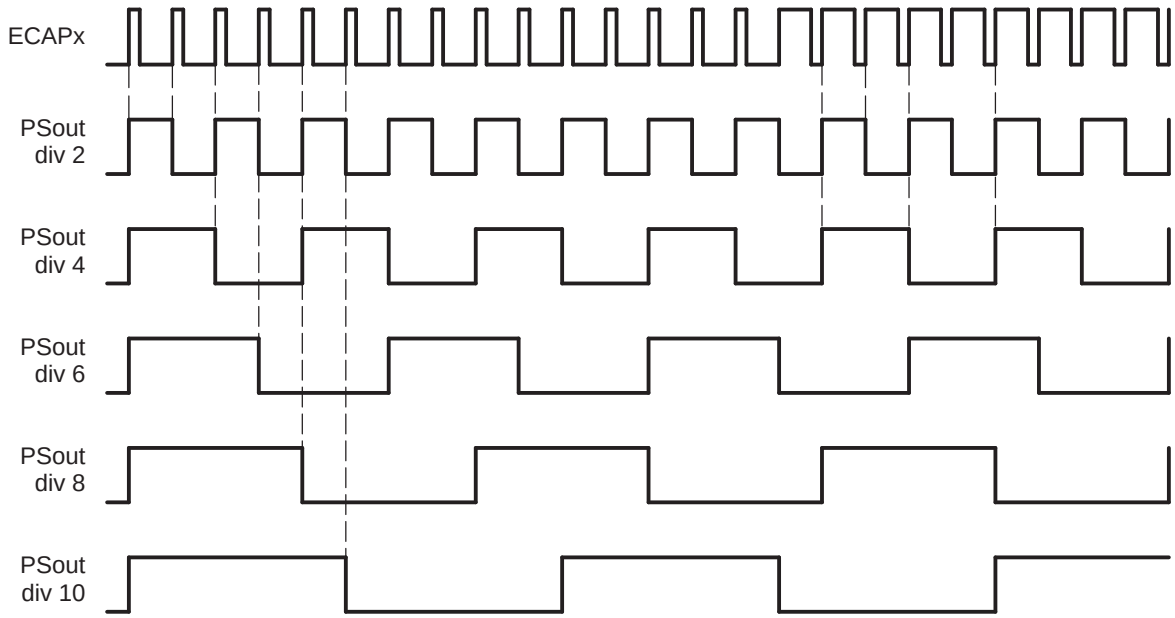
  This is useful when very high frequency signals are used as inputs. Figure 20-4 shows a functional diagram and Figure 20-5 shows the operation of the prescale function.

**Figure 20-4. Event Prescale Control**




A When a prescale value of 1 is chosen ( ECCTL1[13:9] = 0,0,0,0,0 ) the input capture signal bypasses the prescale logic completely.

**Figure 20-5. Prescale Function Waveforms**




### 20.5.2 Edge Polarity Select and Qualifier

- Four independent edge polarity (rising edge/falling edge) selection MUXes are used, one for each capture event.
- Each edge (up to 4) is event qualified by the Modulo4 sequencer.
- The edge event is gated to its respective CAPx register by the Mod4 counter. The CAPx register is loaded on the falling edge.

### 20.5.3 Continuous/One-Shot Control

- The Mod4 (2 bit) counter is incremented via edge qualified events (CEVT1-CEVT4).
- The Mod4 counter continues counting (0->1->2->3->0) and wraps around unless stopped.
- A 2-bit stop register is used to compare the Mod4 counter output, and when equal stops the Mod4 counter and inhibits further loads of the CAP1-CAP4 registers. This occurs during one-shot operation.

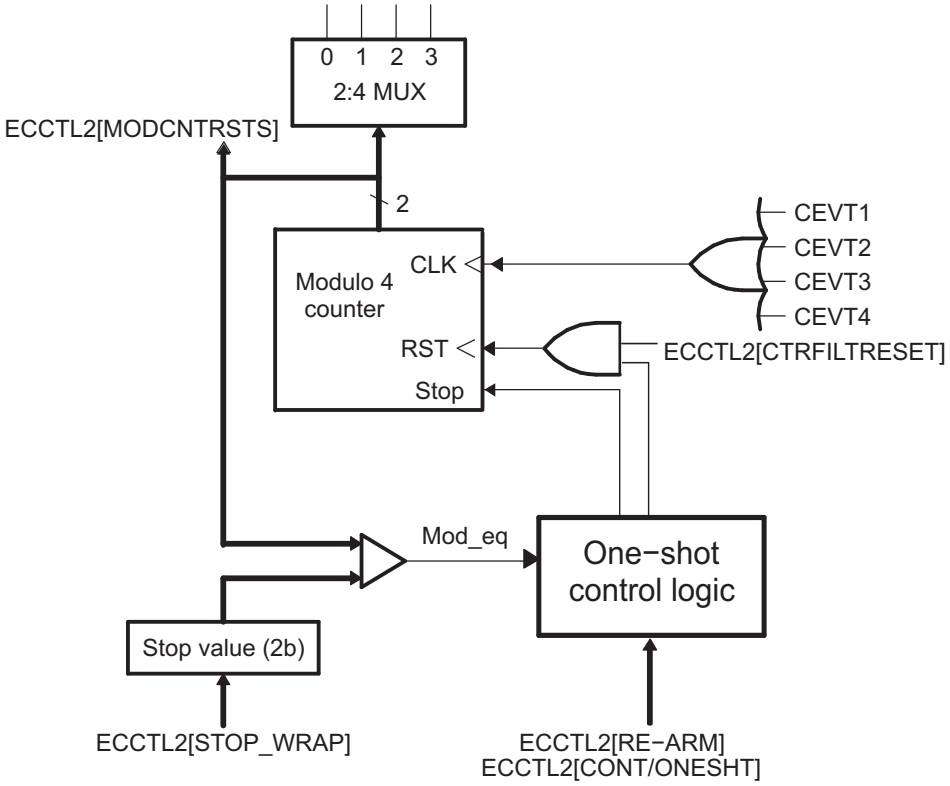
The continuous/one-shot block controls the start/stop and reset (zero) functions of the Mod4 counter via a mono-shot type of action that can be triggered by the stop-value comparator and re-armed via software control.

Once armed, the eCAP module waits for 1-4 (defined by stop-value) capture events before freezing both the Mod4 counter and contents of CAP1-4 registers (time-stamps).

Re-arming prepares the eCAP module for another capture sequence. Also re-arming clears (to zero) the Mod4 counter and permits loading of CAP1-4 registers again, providing the CAPLDEN bit is set.

In continuous mode, the Mod4 counter continues to run (0->1->2->3->0, the one-shot action is ignored, and capture values continue to be written to CAP1-4 in a circular buffer sequence.

**Figure 20-6. Details of the Continuous/One-shot Block**



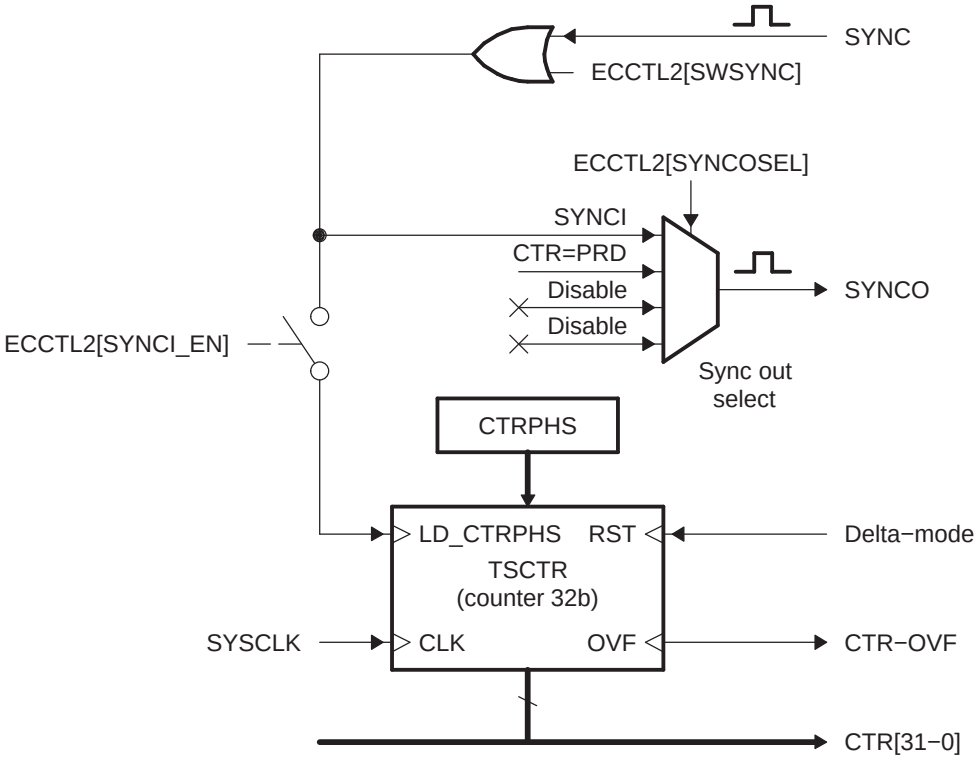

### 20.5.4 32-Bit Counter and Phase Control

This counter provides the time-base for event captures, and is clocked via the system clock.

A phase register is provided to achieve synchronization with other counters, via a hardware and software forced sync. This is useful in APWM mode when a phase offset between modules is needed.

On any of the four event loads, an option to reset the 32-bit counter is given. This is useful for time difference capture. The 32-bit counter value is captured first, then it is reset to 0 by any of the LD1-LD4 signals.

**Figure 20-7. Details of the Counter and Synchronization Block**




### 20.5.5 CAP1-CAP4 Registers

These 32-bit registers are fed by the 32-bit counter timer bus, CTR[0-31] and are loaded (capture a time-stamp) when their respective LD inputs are strobed.

Loading of the capture registers can be inhibited via control bit CAPLDEN. During one-shot operation, this bit is cleared (loading is inhibited) automatically when a stop condition occurs, StopValue = Mod4.

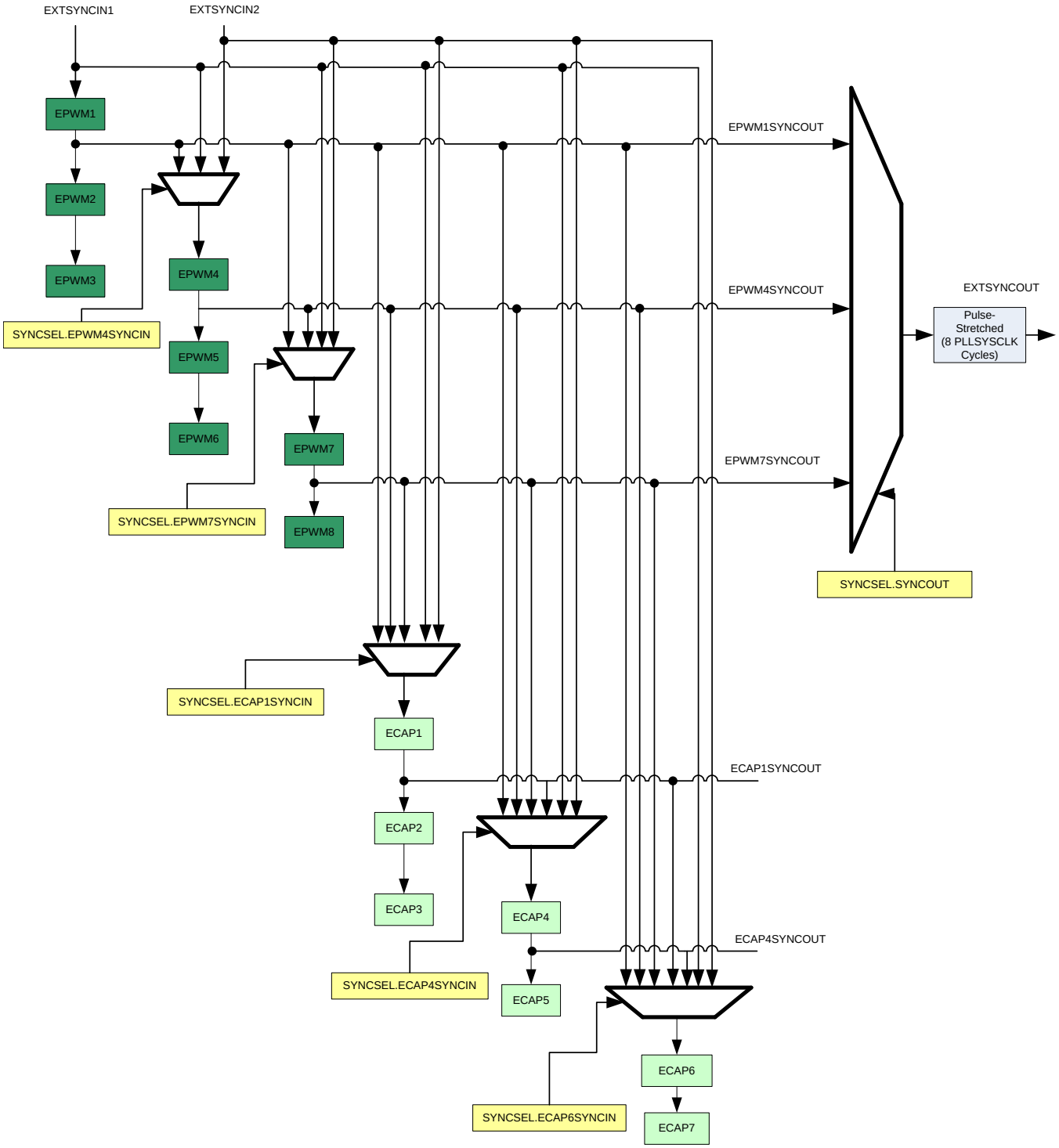
CAP1 and CAP2 registers become the active period and compare registers, respectively, in APWM mode.

CAP3 and CAP4 registers become the respective shadow registers (APRD and ACMP) for CAP1 and CAP2 during APWM operation.

### 20.5.6 Using SWSYNC with the ECAP Module

The SWSYNC of the ECAP module is logical OR'd with the SYNC signal as shown in Figure 20-7. The SYNC signal is defined by the selection in the SYNCSEL[ECAPxSYNCIN] bit for ECAP1 and ECAP4 as shown in Figure 20-8.

**Figure 20-8. Time-Base Counter Synchronization Scheme**




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).

Other eCAP modules receive the SYNC signal from the previous eCAP module. To use SWYNC with ECAP1 and ECAP4, the following workaround can be implemented.

- Select an unused GPIO in InputXbarRegs.INPUT5SELECT. Configure this GPIO in output mode and Write ‘0’ to GPIO DAT register. By default this is programmed to GPIO0 so any activity on this pin will

cause problems with the SWSYNC.

- Program SYNCSEL[ECAPxSYNCIN] = 0x101.
- This will take ECAPx.EXTSYNCIN to an inactive state.

To use SWSYNC with other eCAP modules, take measures to ensure that the previous eCAP chain is not generating a SYNCOUT signal which will interfere with the software synchronization.

### 20.5.7 Interrupt Control

An Interrupt can be generated on capture events (CEVT1-CEVT4, CTROVF) or APWM events (CTR = PRD, CTR = CMP).

A counter overflow event (FFFFFFFF->00000000) is also provided as an interrupt source (CTROVF).

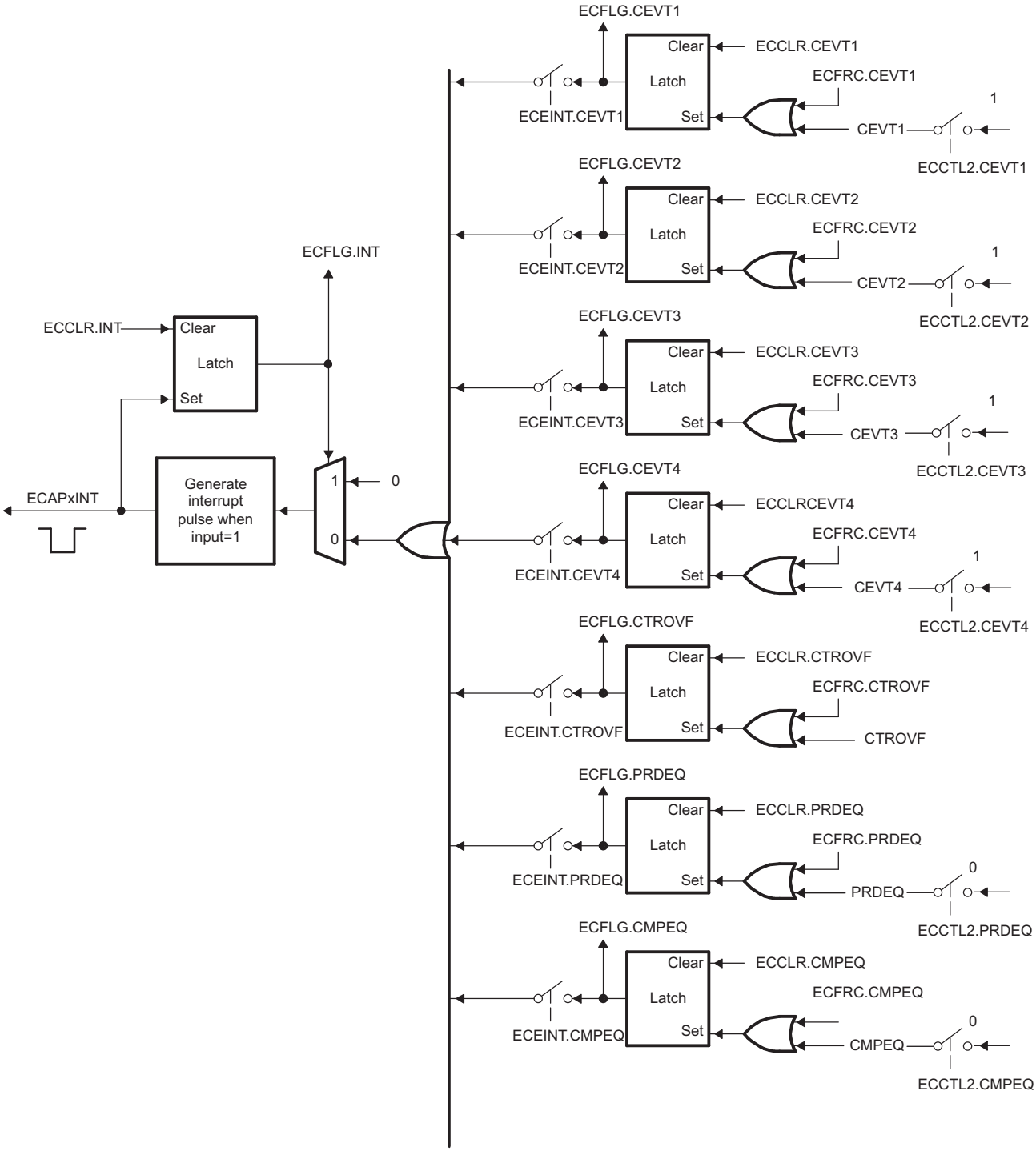
The capture events are edge and sequencer qualified (ordered in time) by the polarity select and Mod4 gating, respectively.

One of these events can be selected as the interrupt source (from the eCAPx module) going to the PIE.

Seven interrupt events (CEVT1, CEVT2, CEVT3, CEVT4, CNTOVF, CTR=PRD, CTR=CMP) can be generated. The interrupt enable register (ECEINT) is used to enable/disable individual interrupt event sources. The interrupt flag register (ECFLG) indicates if any interrupt event has been latched and contains the global interrupt flag bit (INT). An interrupt pulse is generated to the PIE only if any of the interrupt events are enabled, the flag bit is 1, and the INT flag bit is 0. The interrupt service routine must clear the global interrupt flag bit and the serviced event via the interrupt clear register (ECCLR) before any other interrupt pulses are generated. All interrupt flags will be cleared upon an event filter reset by writing a "1" to ECCTL2[CLRFILTRESET]. You can force an interrupt event via the interrupt force register (ECFRC). This is useful for test purposes.

**Note:** The CEVT1, CEVT2, CEVT3, CEVT4 flags are only active in capture mode (ECCTL2[CAP/APWM == 0]). The CTR=PRD, CTR=CMP flags are only valid in APWM mode (ECCTL2[CAP/APWM == 1]). CNTOVF flag is valid in both modes.

**Figure 20-9. Interrupts in eCAP Module**




### 20.5.8 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 (CEVT1, CEVT2, CEVT3 and CEVT4) can be selected as the trigger source for ECAP_DMA_INT using ECCCTL2 {DMAEVTSEL].

### 20.5.9 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:

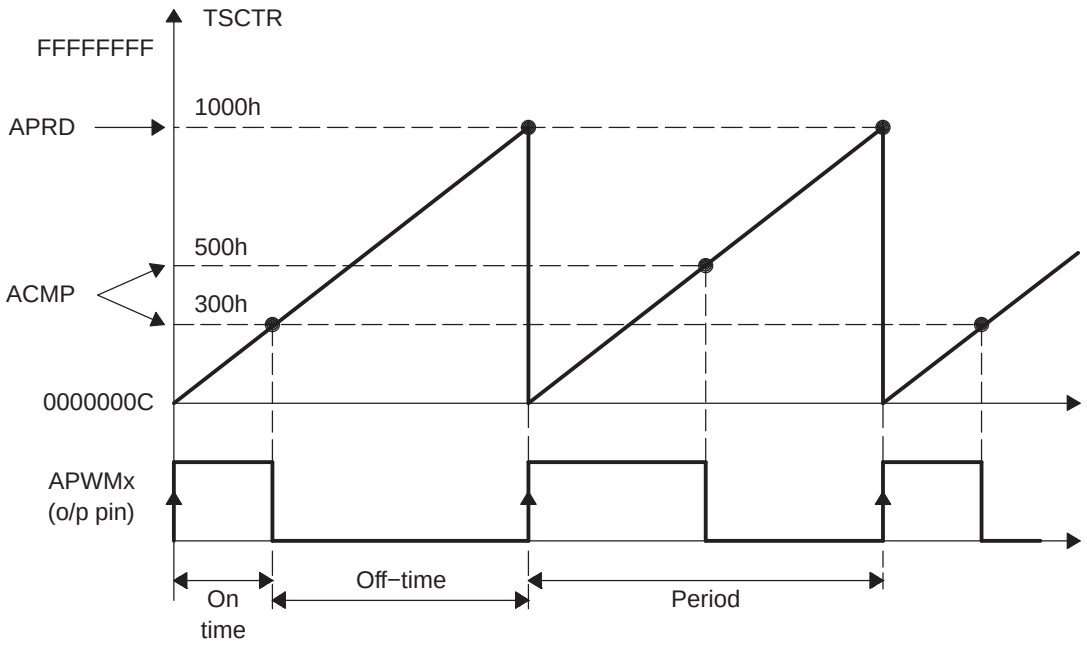
- Immediate - APRD or ACMP are transferred to CAP1 or CAP2 immediately upon writing a new value.
- On period equal, CTR[31:0] = PRD[31:0].

### 20.5.10 APWM Mode Operation

Main operating highlights of the APWM section:

- The time-stamp counter bus is made available for comparison via 2 digital (32-bit) comparators.
- When CAP1/2 registers are not used in capture mode, their contents can be used as Period and Compare values in APWM mode.
- Double buffering is achieved via shadow registers APRD and ACMP (CAP3/4). The shadow register contents are transferred over to CAP1/2 registers either immediately upon a write, or on a CTR = PRD trigger.
- In APWM mode, writing to CAP1/CAP2 active registers will also write the same value to the corresponding shadow registers CAP3/CAP4. This emulates immediate mode. Writing to the shadow registers CAP3/CAP4 will invoke the shadow mode.
- During initialization, you must write to the active registers for both period and compare. This automatically copies the initial values into the shadow values. For subsequent compare updates, during run-time, you only need to use the shadow registers.

**Figure 20-10. PWM Waveform Details Of APWM Mode Operation**



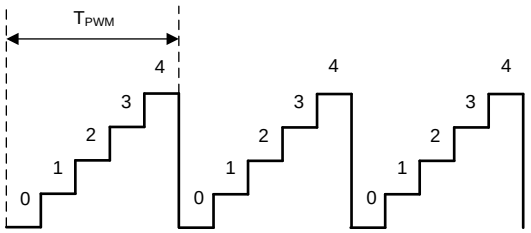

The behavior of APWM active high mode (APWMPOL == 0) is as follows:

```
CMP = 0x00000000, output low for duration of period (0% duty)
CMP = 0x00000001, output high 1 cycle
CMP = 0x00000002, output high 2 cycles
CMP = PERIOD, output high except for 1 cycle (<100% duty)
CMP = PERIOD+1, output high for complete period (100% duty)
CMP > PERIOD+1, output high for complete period
```

The behavior of APWM active low mode (APWMPOL == 1) is as follows:

```
CMP = 0x00000000, output high for duration of period (0% duty)
CMP = 0x00000001, output low 1 cycle
CMP = 0x00000002, output low 2 cycles
CMP = PERIOD, output low except for 1 cycle (<100% duty)
CMP = PERIOD+1, output low for complete period (100% duty)
CMP > PERIOD+1, output low for complete period
```

**Figure 20-11. Time-Base Frequency and Period Calculation**




$$T_{PWM} = (CAP1 + 1) \times T_{TSCTR}$$

$$F_{PWM} = \frac{1}{T_{PWM}}$$

## 20.6 Application of the ECAP Module

The following sections will provide Applications examples and code snippets to show how to configure and operate the eCAP module. For clarity and ease of use, the examples use the eCAP “C” header files. Below are useful #defines which will help in the understanding of the examples.

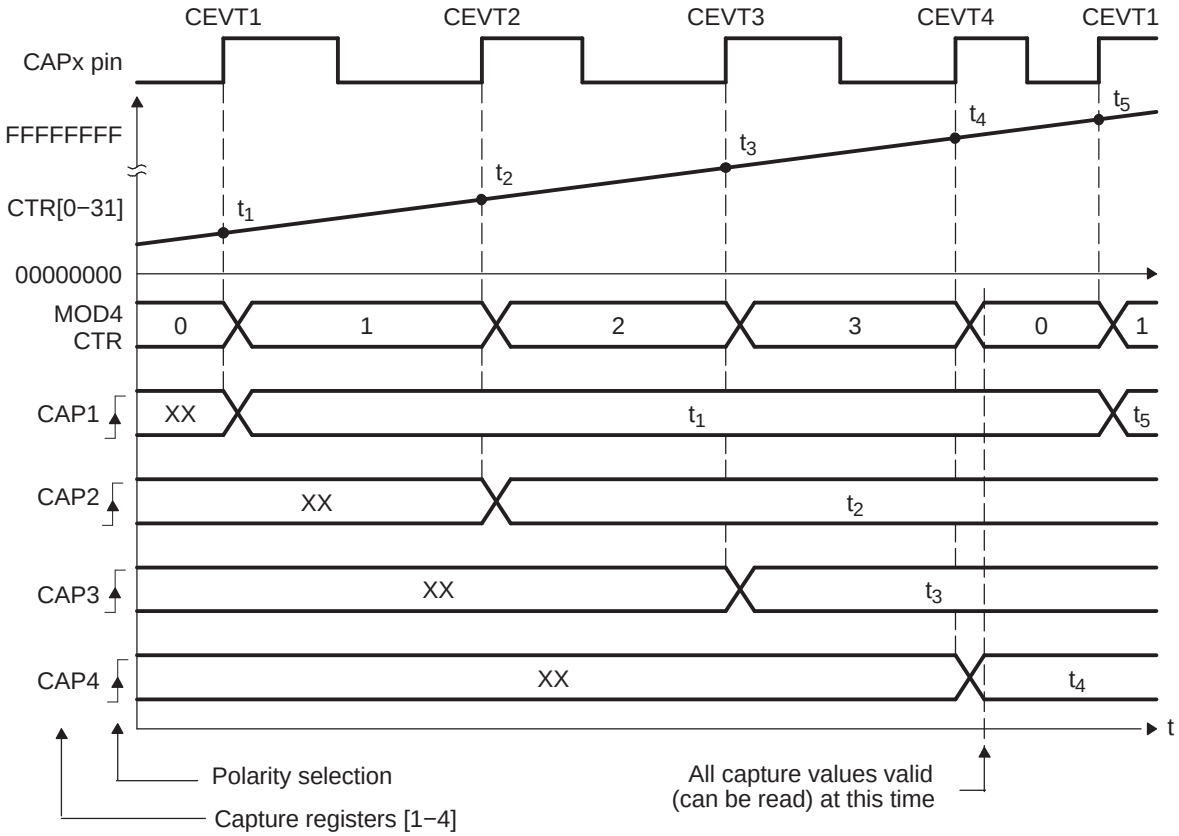
```
// ECCTL1 (ECAP Control Reg 1)

 //==========================

// CAPxPOL bits      #define EC_RISING 0x0    #define EC_FALLING 0x1
// CTRRSTx bits      #define EC_ABS_MODE 0x0  #define EC_DELTA_MODE 0x1
// PRESCALE bits     #define EC_BYPASS 0x0    #define EC_DIV1 0x0 #define EC_DIV2 0x1
                                              #define EC_DIV4 0x2 #define EC_DIV6 0x3
                                              #define EC_DIV8 0x4 #define EC_DIV10 0x5
// ECCTL2 ( ECAP Control Reg 2)
//==========================
// CONT/ONESHOT bit  #define EC_CONTINUOUS 0x0 #define EC_ONESHOT 0x1
// STOPVALUE bit     #define EC_EVENT1 0x0     #define EC_EVENT2 0x1 #define EC_EVENT3 0x2
                                               #define EC_EVENT4 0x3
// RE-ARM bit #define EC_ARM 0x1
// TSCTRSTOP bit     #define EC_FREEZE 0x0     #define EC_RUN 0x1
// SYNCO_SEL bit     #define EC_SYNCIN 0x0     #define EC_CTR_PRD 0x1
                                               #define EC_SYNCO_DIS 0x2
// CAP/APWM mode bit #define EC_CAP_MODE 0x0   #define EC_APWM_MODE 0x1
// APWMPOL bit       #define EC_ACTV_HI 0x0    #define EC_ACTV_LO 0x1
// Generic           #define EC_DISABLE 0x0    #define EC_ENABLE 0x1 #define EC_FORCE 0x1
```

### *20.6.1 Example 1 - Absolute Time-Stamp Operation Rising Edge Trigger*

Figure 20-12 shows an example of continuous capture operation (Mod4 counter wraps around). In this figure, TSCTR counts-up without resetting and capture events are qualified on the rising edge only, this gives period (and frequency) information.

On an event, the TSCTR contents (time-stamp) is first captured, then Mod4 counter is incremented to the next state. When the TSCTR reaches FFFFFFFF (maximum value), it wraps around to 00000000 (not shown in Figure 20-12), if this occurs, the CTROVF (counter overflow) flag is set, and an interrupt (if enabled) occurs, CTROVF (counter overflow) Flag is set, and an Interrupt (if enabled) occurs. Captured Time-stamps are valid at the point indicated by the diagram (after the 4th event), hence event CEVT4 can conveniently be used to trigger an interrupt and the CPU can read data from the CAPx registers.

**Figure 20-12. Capture Sequence for Absolute Time-stamp and Rising Edge Detect**



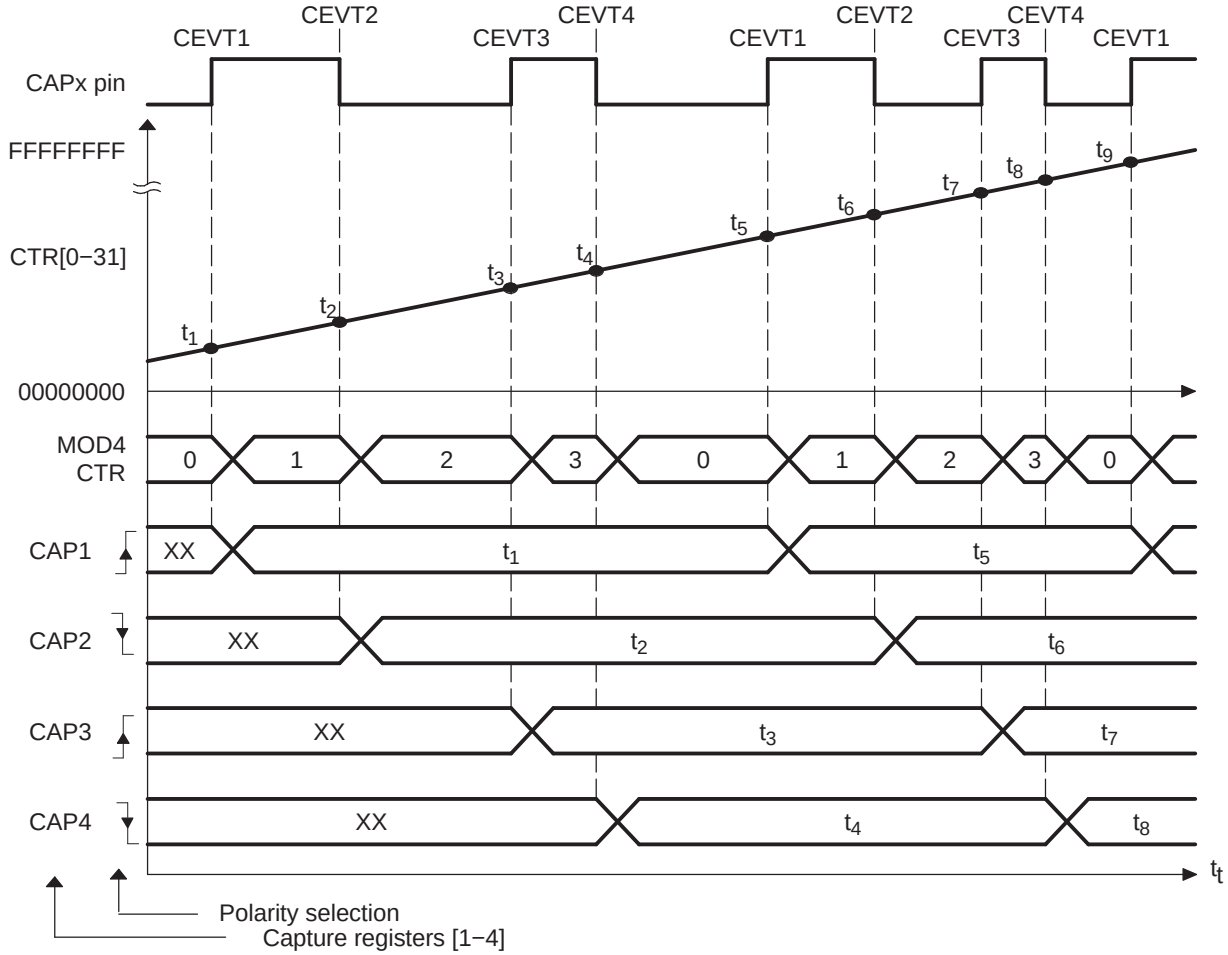
### 20.6.1.1 Code snippet for CAP mode Absolute Time, Rising Edge Trigger

```
// Code snippet for CAP mode Absolute Time, Rising edge trigger

// Initialization Time
//=======================

// ECAP module 1 config ECap1Regs.ECCTL1.bit.CAP1POL = EC_RISING;

   ECap1Regs.ECCTL1.bit.CAP2POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CAP3POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CAP4POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CTRRST1 = EC_ABS_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST2 = EC_ABS_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST3 = EC_ABS_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST4 = EC_ABS_MODE;
   ECap1Regs.ECCTL1.bit.CAPLDEN = EC_ENABLE;
   ECap1Regs.ECCTL1.bit.PRESCALE = EC_DIV1;
   ECap1Regs.ECCTL2.bit.CAP_APWM = EC_CAP_MODE;
   ECap1Regs.ECCTL2.bit.CONT_ONESHT = EC_CONTINUOUS;
   ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;
   ECap1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE;
   ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN;

// Allow TSCTR to run

// Run Time (CEVT4 triggered ISR call)
//=======================================

TSt1 = ECap1Regs.CAP1;
// Fetch Time-Stamp captured at t1 TSt2 = ECap1Regs.CAP2;
// Fetch Time-Stamp captured at t2 TSt3 = ECap1Regs.CAP3;
// Fetch Time-Stamp captured at t3 TSt4 = ECap1Regs.CAP4;
// Fetch Time-Stamp captured at t4 Period1 = TSt2-TSt1;
// Calculate 1st period Period2 = TSt3-TSt2;
// Calculate 2nd period Period3 = TSt4-TSt3;
// Calculate 3rd period
```

### 20.6.2 Example 2 - Absolute Time-Stamp Operation Rising and Falling Edge Trigger

In Figure 20-13 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, i.e: Period1 = $t_3 - t_1$, Period2 = $t_5 - t_3$, …etc. Duty Cycle1 (on-time %) = ($t_2 - t_1$) / Period1 x 100%, etc. Duty Cycle1 (off-time %) = ($t_3 - t_2$) / Period1 x 100%, etc.

**Figure 20-13. Capture Sequence for Absolute Time-stamp With Rising and Falling Edge Detect**



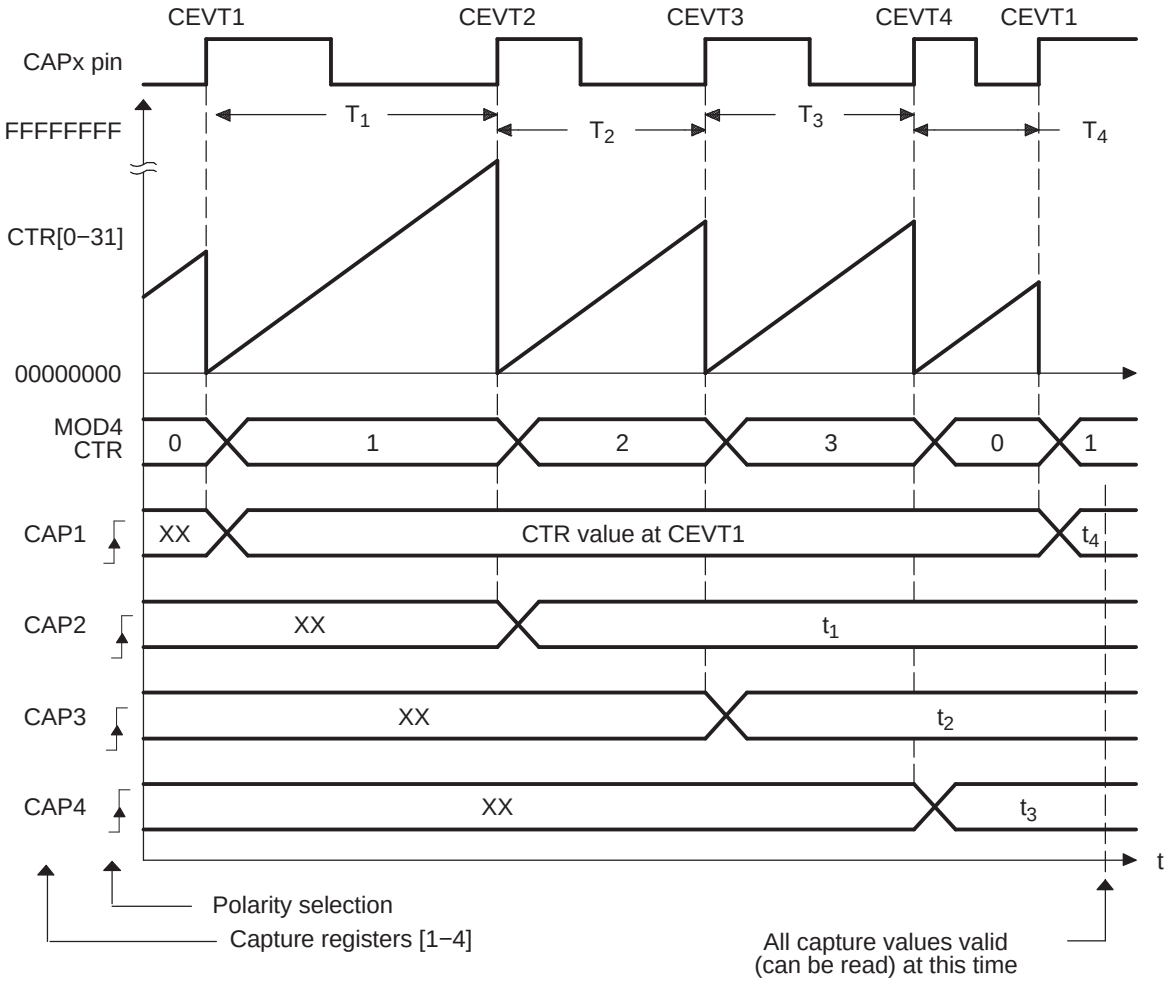
### 20.6.2.1 Code Snippet for CAP mode Absolute Time, Rising and Falling Edge Triggers

```
// Code snippet for CAP mode Absolute Time, Rising and Falling

// edge triggers // Initialization Time

//=======================

// ECAP module 1 config ECap1Regs.ECCTL1.bit.CAP1POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CAP2POL = EC_FALLING;
   ECap1Regs.ECCTL1.bit.CAP3POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CAP4POL = EC_FALLING;
   ECap1Regs.ECCTL1.bit.CTRRST1 = EC_ABS_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST2 = EC_ABS_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST3 = EC_ABS_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST4 = EC_ABS_MODE;
   ECap1Regs.ECCTL1.bit.CAPLDEN = EC_ENABLE;
   ECap1Regs.ECCTL1.bit.PRESCALE = EC_DIV1;
   ECap1Regs.ECCTL2.bit.CAP_APWM = EC_CAP_MODE;
   ECap1Regs.ECCTL2.bit.CONT_ONESHT = EC_CONTINUOUS;
   ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;
   ECap1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE;
   ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN;

// Allow TSCTR to run

// Run Time (CEVT4 triggered ISR call)

//=========================================

TSt1 = ECap1Regs.CAP1;
// Fetch Time-Stamp captured at t1 TSt2 = ECap1Regs.CAP2;
// Fetch Time-Stamp captured at t2 TSt3 = ECap1Regs.CAP3;
// Fetch Time-Stamp captured at t3 TSt4 = ECap1Regs.CAP4;
// Fetch Time-Stamp captured at t4 Period1 = TSt3-TSt1;
// Calculate 1st period DutyOnTime1 = TSt2-TSt1;
// Calculate On time DutyOffTime1 = TSt3-TSt2;
// Calculate Off time
```

### 20.6.3 Example 3 - Time Difference (Delta) Operation Rising Edge Trigger

This example Figure 20-14 shows how the eCAP module can be used to collect Delta timing data from pulse train waveforms. Here Continuous Capture mode (TSCTR counts-up without resetting, and Mod4 counter wraps around) is used. In Delta-time mode, TSCTR is Reset back to Zero on every valid event. Here Capture events are qualified as Rising edge only. On an event, TSCTR contents (Time-Stamp) is captured first, and then TSCTR is reset to Zero. The Mod4 counter then increments to the next state. If TSCTR reaches FFFFFFFF (Max value), before the next event, it wraps around to 00000000 and continues, a 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 = $T_2$,…etc. As shown in the diagram, the CEVT1 event is a good trigger point to read the timing data, $T_1$, $T_2$, $T_3$, $T_4$ are all valid here.

**Figure 20-14. Capture Sequence for Delta Mode Time-stamp and Rising Edge Detect**



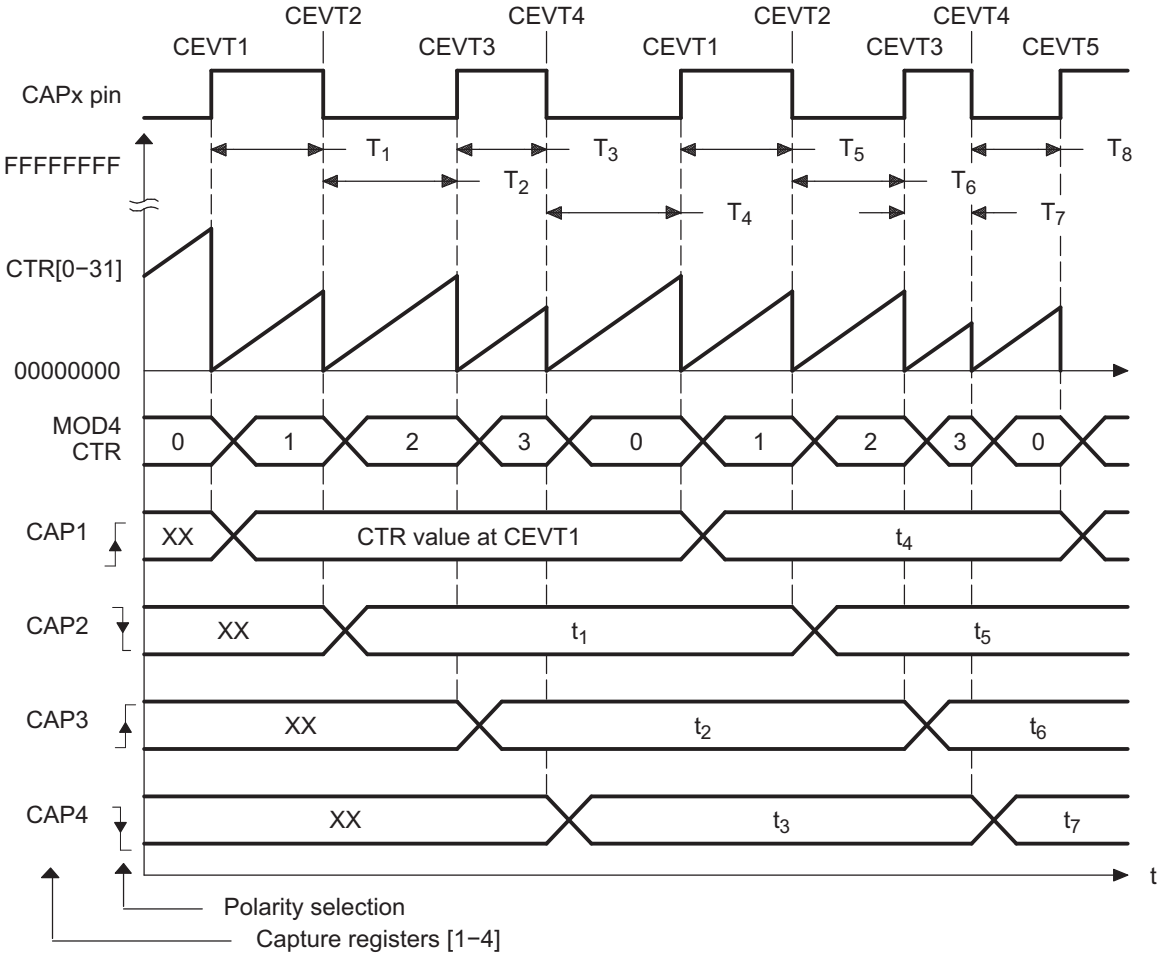
### 20.6.3.1 Code snippet for CAP mode Delta Time, Rising Edge Trigger

```
// Code snippet for CAP mode Delta Time, Rising edge trigger

// Initialization Time

//=======================

// ECAP module 1 config ECap1Regs.ECCTL1.bit.CAP1POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CAP2POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CAP3POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CAP4POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CTRRST1 = EC_DELTA_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST2 = EC_DELTA_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST3 = EC_DELTA_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST4 = EC_DELTA_MODE;
   ECap1Regs.ECCTL1.bit.CAPLDEN = EC_ENABLE;
   ECap1Regs.ECCTL1.bit.PRESCALE = EC_DIV1;
   ECap1Regs.ECCTL2.bit.CAP_APWM = EC_CAP_MODE;
   ECap1Regs.ECCTL2.bit.CONT_ONESHT = EC_CONTINUOUS;
   ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;
   ECap1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE;
   ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN;

// Allow TSCTR to run

// Run Time (CEVT1 triggered ISR call)

//=========================================

// Note: here Time-stamp directly represents the Period value. Period4 = ECap1Regs.CAP1;
// Fetch Time-Stamp captured at T1 Period1 = ECap1Regs.CAP2;
// Fetch Time-Stamp captured at T2 Period2 = ECap1Regs.CAP3;
// Fetch Time-Stamp captured at T3 Period3 = ECap1Regs.CAP4;
// Fetch Time-Stamp captured at T4
```

### 20.6.4 Example 4 - Time Difference (Delta) Operation Rising and Falling Edge Trigger

In Figure 20-15 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, i.e: Period1 = $T_1$+$T_2$, Period2 = $T_3$+$T_4$, …etc Duty Cycle1 (on-time %) = $T_1$ / Period1 x 100%, etc Duty Cycle1 (off-time %) = $T_2$ / Period1 x 100%, etc

**Figure 20-15. Capture Sequence for Delta Mode Time-stamp With Rising and Falling Edge Detect**




During initialization, you must write to the active registers for both period and compare. This will then automatically copy the init values into the shadow values. For subsequent compare updates, during run-time, only the shadow registers must be used.

### 20.6.4.1 Code snippet for CAP mode Delta Time, Rising and Falling Edge Triggers

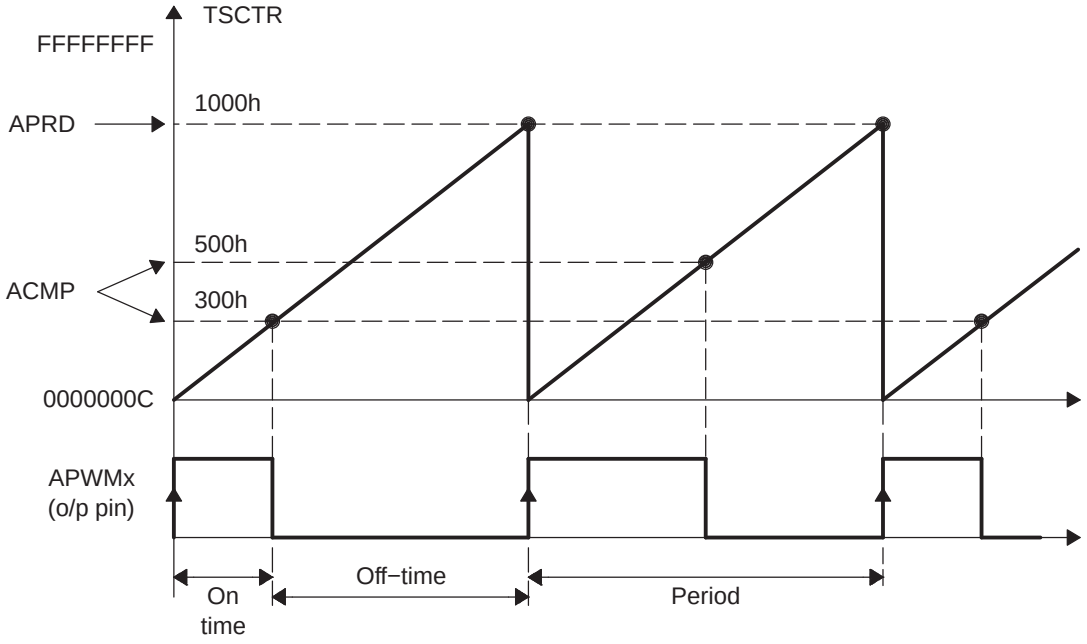
```
// Code snippet for CAP mode Delta Time, Rising and Falling
// edge triggers
// Initialization Time

//=======================

// ECAP module 1 config ECap1Regs.ECCTL1.bit.CAP1POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CAP2POL = EC_FALLING;
   ECap1Regs.ECCTL1.bit.CAP3POL = EC_RISING;
   ECap1Regs.ECCTL1.bit.CAP4POL = EC_FALLING;
   ECap1Regs.ECCTL1.bit.CTRRST1 = EC_DELTA_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST2 = EC_DELTA_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST3 = EC_DELTA_MODE;
   ECap1Regs.ECCTL1.bit.CTRRST4 = EC_DELTA_MODE;
   ECap1Regs.ECCTL1.bit.CAPLDEN = EC_ENABLE;
   ECap1Regs.ECCTL1.bit.PRESCALE = EC_DIV1;
   ECap1Regs.ECCTL2.bit.CAP_APWM = EC_CAP_MODE;
   ECap1Regs.ECCTL2.bit.CONT_ONESHT = EC_CONTINUOUS;
   ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;
   ECap1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE;
   ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN;

// Allow TSCTR to run

// Run Time (CEVT1 triggered ISR call)

//=========================================
//
Note: here Time-stamp directly represents the Duty cycle values. DutyOnTime1 = ECap1Regs.CAP2;

// Fetch Time-Stamp captured at T2 DutyOffTime1 = ECap1Regs.CAP3;
// Fetch Time-Stamp captured at T3 DutyOnTime2 = ECap1Regs.CAP4;
// Fetch Time-Stamp captured at T4 DutyOffTime2 = ECap1Regs.CAP1;
// Fetch Time-
Stamp captured at T1 Period1 = DutyOnTime1 + DutyOffTime1; Period2 = DutyOnTime2 + DutyOffTime2;
```

## 20.7 Application of the APWM Mode

In this example, the eCAP module is configured to operate as a PWM generator. Here a very simple single channel PWM waveform is generated from output pin APWMx. The PWM polarity is active high, which means that the compare value (CAP2 reg is now a compare register) represents the on-time (high level) of the period. Alternatively, if the APWMPOL bit is configured for active low, then the compare value represents the off-time. Note here values are in hexadecimal (“h”) notation.

### *20.7.1 Example 1 - Simple PWM Generation (Independent Channel/s)*

**Figure 20-16. PWM Waveform Details of APWM Mode Operation**




***Example 20-1. Code Snippet for APWM Mode***

```
// Code snippet for APWM mode Example 1

// Initialization Time

//=======================

// ECAP module 1 config ECap1Regs.CAP1 = 0x1000;

// Set period value ECap1Regs.CTRPHS = 0x0;

// make phase zero ECap1Regs.ECCTL2.bit.CAP_APWM = EC_APWM_MODE;
                   ECap1Regs.ECCTL2.bit.APWMPOL = EC_ACTV_HI;

// Active high ECap1Regs.ECCTL2.bit.SYNCI_EN = EC_DISABLE;

// Synch not used ECap1Regs.ECCTL2.bit.SYNCO_SEL = EC_SYNCO_DIS;

// Synch not used ECap1Regs.ECCTL2.bit.TSCTRSTOP = EC_RUN;

// Allow TSCTR to run

// Run Time (Instant 1, for example, ISR call)

//======================
   ECap1Regs.CAP2 = 0x300;
```

***Example 20-1. Code Snippet for APWM Mode (continued)***

```
// Set Duty cycle, that  is,  compare value

// Run Time (Instant 2, for example, another ISR call)

//======================\
   ECap1Regs.CAP2 = 0x500;

// Set Duty cycle, that is, compare value
```

## 20.8 Registers

### 20.8.1 Enhanced Capture Base Addresses

**Table 20-2. eCAP Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| ECap1Regs | ECAP_REGS | 0x0000_5200 | 0x0000_521F |
| ECap2Regs | ECAP_REGS | 0x0000_5240 | 0x0000_525F |
| ECap3Regs | ECAP_REGS | 0x0000_5280 | 0x0000_529F |
| ECap4Regs | ECAP_REGS | 0x0000_52C0 | 0x0000_52DF |
| ECap5Regs | ECAP_REGS | 0x0000_5300 | 0x0000_531F |
| ECap6Regs | ECAP_REGS | 0x0000_5340 | 0x0000_535F |
| ECap7Regs | ECAP_REGS | 0x0000_5380 | 0x0000_539F |

### 20.8.1.1 ECAP_REGS Registers

Table 20-3 lists the memory-mapped registers for the ECAP_REGS. All register offset addresses not listed in Table 20-3 should be considered as reserved locations and the register contents should not be modified.

**Table 20-3. 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 |
| 12h | ECCTL0 | Capture Control Register 0 | EALLOW | Go |
| 14h | ECCTL1 | Capture Control Register 1 | EALLOW | Go |
| 15h | ECCTL2 | Capture Control Register 2 | EALLOW | Go |
| 16h | ECEINT | Capture Interrupt Enable Register | EALLOW | Go |
| 17h | ECFLG | Capture Interrupt Flag Register | | Go |
| 18h | ECCLR | Capture Interrupt Clear Register | | Go |
| 19h | ECFRC | Capture Interrupt Force Register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 20-4 shows the codes that are used for access types in this section.

**Table 20-4. ECAP_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| W=1 | 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. |

#### 20.8.1.1.1 TSCTR Register (Offset = 0h) [reset = 0h]

TSCTR is shown in Figure 20-17 and described in Table 20-5.

Return to Summary Table.

Time-Stamp Counter

**Figure 20-17. 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 20-5. TSCTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | TSCTR | R/W | 0h | Active 32-bit counter register that is used as the capture time-base<br>HR mode :<br>1) This register reads HRCOUNTER value and is not writable<br>2) can be reset using CTRFILTRESET<br>3) Its not synchronized to SYSCLK domain so reads may not be accurate<br>Reset type: SYSRSn |

#### 20.8.1.1.2 CTRPHS Register (Offset = 2h) [reset = 0h]

CTRPHS is shown in Figure 20-18 and described in Table 20-6.

Return to Summary Table.

Counter Phase Offset Value Register

**Figure 20-18. 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 20-6. CTRPHS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | CTRPHS | R/W | 0h | Counter phase value register that can be programmed for phase lag/lead. This register CTRPHS is loaded into TSCTR upon either a SYNCI event or S/W force via a control bit. Used to achieve phase control synchronization with respect to other eCAP and EPWM time-bases.<br>This register is not applicable in HR mode.<br>Reset type: SYSRSn |

#### 20.8.1.1.3 CAP1 Register (Offset = 4h) [reset = 0h]

CAP1 is shown in Figure 20-19 and described in Table 20-7.

Return to Summary Table.

Capture 1 Register

**Figure 20-19. 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 20-7. 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) during a capture event<br>- Software - may be useful for test purposes<br>- APRD shadow register (CAP3) when used in APWM mode<br>Reset type: SYSRSn |

#### 20.8.1.1.4 CAP2 Register (Offset = 6h) [reset = 0h]

CAP2 is shown in Figure 20-20 and described in Table 20-8.

Return to Summary Table.

Capture 2 Register

**Figure 20-20. 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 20-8. 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>- APRD shadow register (CAP4) when used in APWM mode<br>Reset type: SYSRSn |

#### 20.8.1.1.5 CAP3 Register (Offset = 8h) [reset = 0h]

CAP3 is shown in Figure 20-21 and described in Table 20-9.

Return to Summary Table.

Capture 3 Register

**Figure 20-21. 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 20-9. CAP3 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | CAP3 | R/W | 0h | In CMP mode, this is a time-stamp capture register.<br>In APWM mode, this is the period shadow (APRD) register. You update the PWM period value through this register. In this mode, CAP3 (APRD) shadows CAP1.<br>Reset type: SYSRSn |

#### 20.8.1.1.6 CAP4 Register (Offset = Ah) [reset = 0h]

CAP4 is shown in Figure 20-22 and described in Table 20-10.

Return to Summary Table.

Capture 4 Register

**Figure 20-22. 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 20-10. CAP4 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | CAP4 | R/W | 0h | In CMP mode, this is a time-stamp capture register.<br>In APWM mode, this is the compare shadow (ACMP) register. You update the PWM compare value via this register. In this mode, CAP4 (ACMP) shadows CAP2.<br>Reset type: SYSRSn |

#### 20.8.1.1.7 ECCTL0 Register (Offset = 12h) [reset = 7Fh]

ECCTL0 is shown in Figure 20-23 and described in Table 20-11.

Return to Summary Table.

Capture Control Register 0

**Figure 20-23. ECCTL0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | |
| R=0-0h | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | | | INPUTSEL | | | | | | |
| R=0-0h | | | | | | | | | R/W-7Fh | | | | | | |

**Table 20-11. ECCTL0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-7 | RESERVED | R=0 | 0h | Reserved |
| 6-0 | INPUTSEL | R/W | 7Fh | Capture input source select bits<br>0000000 capture input is ECAPxINPUT[0]<br>0000001 capture input is ECAPxINPUT[1]<br>0000010 capture input is ECAPxINPUT[2]<br>...<br>1111111 capture input is ECAPxINPUT[127]<br>Reset type: CPU1.SYSRSn |

#### 20.8.1.1.8 ECCTL1 Register (Offset = 14h) [reset = 0h]

ECCTL1 is shown in Figure 20-24 and described in Table 20-12.

Return to Summary Table.

Capture Control Register 1

**Figure 20-24. ECCTL1 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| FREE_SOFT | | PRESCALE | | | | | CAPLDEN |
| R/W-0h | | R/W-0h | | | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CTRRST4 | CAP4POL | CTRRST3 | CAP3POL | CTRRST2 | CAP2POL | CTRRST1 | CAP1POL |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 20-12. 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) |

**Table 20-12. 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) |

#### 20.8.1.1.9 ECCTL2 Register (Offset = 15h) [reset = 6h]

ECCTL2 is shown in Figure 20-25 and described in Table 20-13.

Return to Summary Table.

Capture Control Register 2

**Figure 20-25. ECCTL2 Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| MODCNTRSTS | | DMAEVTSEL | | CTRFILTRESET | APWMPOL | CAP_APWM | SWSYNC |
| R-0h | | R/W-0h | | R=0/W=1-0h | R/W-0h | R/W-0h | R/W-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/W-0h | R/W-3h | | R/W-0h |

**Table 20-13. ECCTL2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-14 | MODCNTRSTS | R | 0h | This bit field reads current status on modulo counter<br>00b (R) = CAP1 register gets loaded on next capture event.<br>01b (R) = CAP2 register gets loaded on next capture event.<br>10b (R) = CAP3 register gets loaded on next capture event.<br>11b (R) = CAP4 register gets loaded on next capture event.<br>Reset type: CPU1.SYSRSn |
| 13-12 | DMAEVTSEL | R/W | 0h | DMA event select<br>00b (R/W) = DMA interrupt source is CEVT1<br>01b (R/W) = DMA interrupt source is CEVT2<br>10b (R/W) = DMA interrupt source is CEVT3<br>11b (R/W) = DMA interrupt source is CEVT4<br>Reset type: CPU1.SYSRSn |
| 11 | CTRFILTRESET | R=0/W=1 | 0h | Reset Bit<br>0h (R) = No effect<br>1h (W) = Resets event filter, counter, modulo counter and CEVT[1,2,3,4] and CNTOVF , HRERROR flags<br>Note: This provides an ability start capture module from known state in case spurious inputs are captured while ECAP is configured.<br>Reset type: CPU1.SYSRSn |
| 10 | APWMPOL | R/W | 0h | APWM output polarity select.<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) |

**Table 20-13. ECCTL2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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<br>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<br>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/W | 0h | Software-forced Counter (TSCTR) Synchronizing.<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<br>ECAP modules down-stream providing the SYNCO_SEL bits are 0,0. After writing<br>a 1, this bit returns to a zero.<br>Note: Selection CTR = PRD is meaningful only in APWM mode<br>however, you can<br>choose it in CAP mode if you find doing so useful. |
| 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<br>2h (R/W) = Disable sync out signal<br>3h (R/W) = Disable sync out signal |
| 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/W | 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 |

**Table 20-13. ECCTL2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |

#### 20.8.1.1.10 ECEINT Register (Offset = 16h) [reset = 0h]

ECEINT is shown in Figure 20-26 and described in Table 20-14.

Return to 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 20-26. ECEINT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | RESERVED |
| R/W-0h | | | | | | | 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/W-0h |

**Table 20-14. ECEINT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-9 | RESERVED | R/W | 0h | Reserved |
| 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 |

**Table 20-14. 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/W | 0h | Reserved |

#### 20.8.1.1.11 ECFLG Register (Offset = 17h) [reset = 0h]

ECFLG is shown in Figure 20-27 and described in Table 20-15.

Return to Summary Table.

Capture Interrupt Flag Register

**Figure 20-27. ECFLG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | RESERVED |
| R-0h | | | | | | | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CTR_CMP | CTR_PRD | CTROVF | CEVT4 | CEVT3 | CEVT2 | CEVT1 | INT |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 20-15. ECFLG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 7 | CTR_CMP | R | 0h | Compare Equal Compare Status Flag. This flag is active only in APWM mode.<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. |

**Table 20-15. ECFLG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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. |

#### 20.8.1.1.12 ECCLR Register (Offset = 18h) [reset = 0h]

ECCLR is shown in Figure 20-28 and described in Table 20-16.

Return to Summary Table.

Capture Interrupt Clear Register

**Figure 20-28. ECCLR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | RESERVED |
| R-0h | | | | | | | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CTR_CMP | CTR_PRD | CTROVF | CEVT4 | CEVT3 | CEVT2 | CEVT1 | INT |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 20-16. ECCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 7 | CTR_CMP | R=0/W=1 | 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 condition |
| 6 | CTR_PRD | R=0/W=1 | 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 condition |
| 5 | CTROVF | R=0/W=1 | 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 condition |
| 4 | CEVT4 | R=0/W=1 | 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 condition. |
| 3 | CEVT3 | R=0/W=1 | 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 condition. |
| 2 | CEVT2 | R=0/W=1 | 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 condition. |
| 1 | CEVT1 | R=0/W=1 | 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 condition. |
| 0 | INT | R=0/W=1 | 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 |

#### 20.8.1.1.13 ECFRC Register (Offset = 19h) [reset = 0h]

ECFRC is shown in Figure 20-29 and described in Table 20-17.

Return to Summary Table.

Capture Interrupt Force Register

**Figure 20-29. ECFRC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | RESERVED |
| R-0h | | | | | | | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CTR_CMP | CTR_PRD | CTROVF | CEVT4 | CEVT3 | CEVT2 | CEVT1 | 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 20-17. ECFRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-9 | RESERVED | R | 0h | Reserved |
| 8 | RESERVED | R | 0h | Reserved |
| 7 | CTR_CMP | R/W | 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 bit. |
| 6 | CTR_PRD | R/W | 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 bit. |
| 5 | CTROVF | R/W | 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 bit. |
| 4 | CEVT4 | R/W | 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 bit |
| 3 | CEVT3 | R/W | 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 bit |
| 2 | CEVT2 | R/W | 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 bit. |
| 1 | CEVT1 | R/W | 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 bit. |
| 0 | RESERVED | R | 0h | Reserved |

# High Resolution Capture (HRCAP)

This chapter describes the operation of the high resolution capture (HRCAP) module on the TMS320xF2804x devices. The HRCAP submodule described here is part of the Type 1 eCAP. HRCAP measures the width of external pulses to a higher degree of accuracy than the eCAP module. See the TMS320x28xx, 28xxx DSP Peripheral Reference Guide for a list of all devices with an HRCAP module of the same type, to determine the differences between types, and for a list of device-specific differences within a type). All referenced functions come from C2000Ware; a detailed description of each can be found in the documentation included with C2000Ware.

**Topic** **Page**

## 21.1 Introduction

Uses for the HRCAP include:

- Capacitive touch applications
- High-resolution period and duty cycle measurements of pulse train cycles
- Instantaneous speed measurements
- Instantaneous frequency measurements
- Voltage measurements across an isolation boundary
- Distance/sonar measurement and scanning
- Measuring flow

The HRCAP submodule includes the following features:

- Pulse-width capture in either non-high-resolution or high-resolution modes
- Absolute mode pulse-width capture
- Continuous or one-shot capture
- Interrupt on either falling or rising edge
- Continuous mode capture of pulse widths in 4-deep buffer
- Hardware calibration logic for precision high-resolution capture

All of the above resources are available on any pin using the Input X-BAR

Improvements to the Type 0 HRCAP are as follows:

- Simplified calibration scheme
  - HRCAP is always functional; never offline to perform calibration
  - Calibration is always running in the background; drastically reduced software overhead to calibrate
- Reduced software overhead to compute fractional bits
- Fractional and integer portions are packed into 32 bits
- All eCAP hardware is accessible when using the HRCAP enhancements. See Known Exceptions for practical considerations.
- Usage of the HRCAP is now unified with the eCAP

## 21.2 Description

The HRCAP enhancement has been added to eCAP 6 and eCAP 7 to allow signals to be captured asynchronously to SYSCLK. Each HRCAP submodule includes one capture channel in addition to a hardware calibration block. All eCAP hardware is accessible when using the HRCAP enhancements; however, using the Event Filter or the Input Qualifier is not valid, as these are synchronous to SYSCLK.

Each HRCAP-capable channel has the following independent key resources:

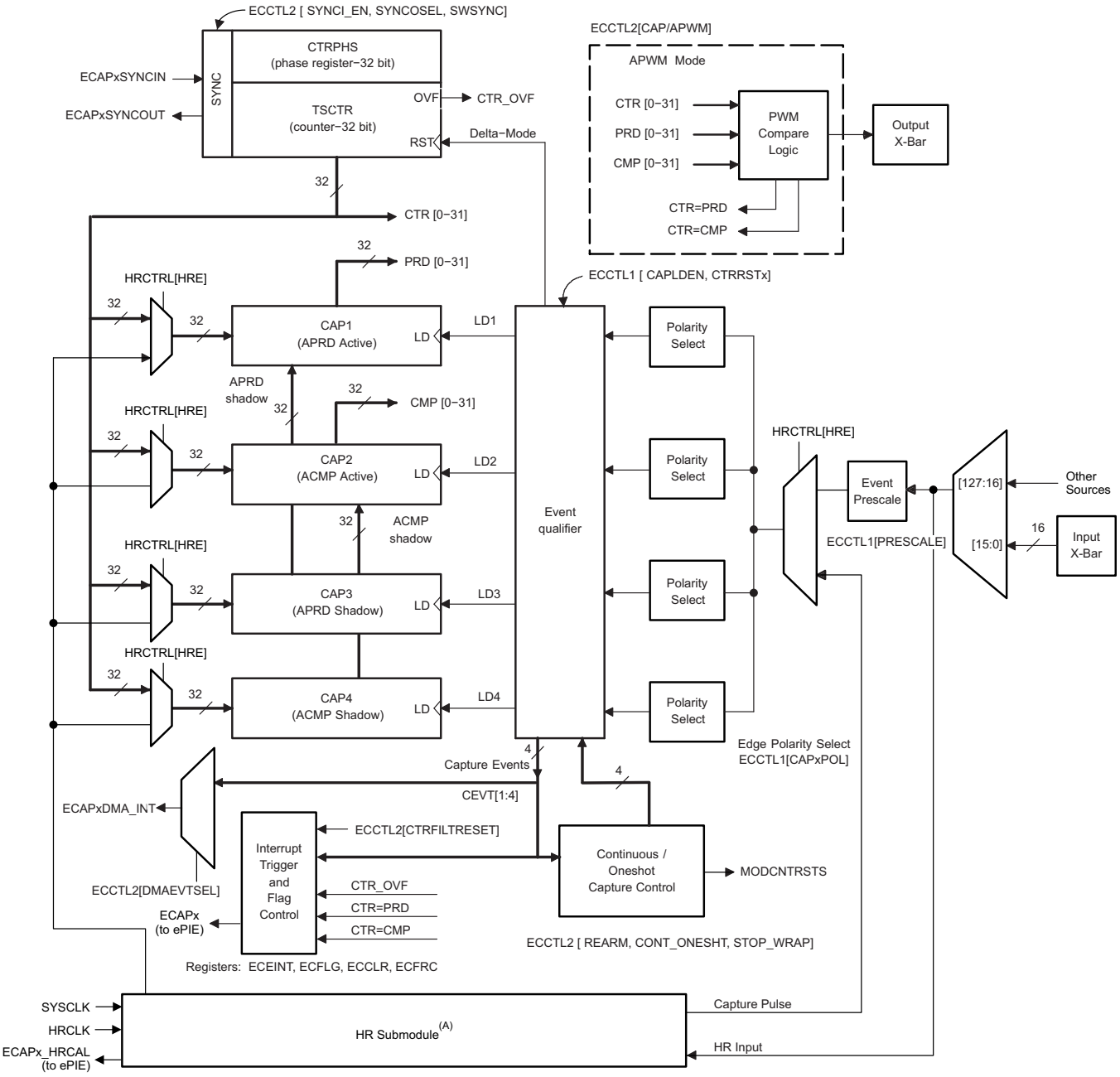
- All hardware of the respective eCAP
- High-resolution calibration logic
- Dedicated calibration interrupt

## 21.3 Operational Details

Figure 21-1 shows the various components that implement the high-resolution capture functionality of the eCAP module. Please note that existing eCAP resources are reused, which requires that the eCAP module is set up before using the HRCAP enhancements. For simplicity, absolute timestamp measurements are recommended. See Known Exceptions for more details.

All HRCAP measurements are relative-time measures, in terms of minimum step size. Calibration hardware as well as software functions have been provided to convert relative-time measurements to 'time-converted' measures in terms of seconds. The calibration hardware and software is only required if time-converted measurements are required.

**Figure 21-1. HRCAP Operations Block Diagram**




A The HRCAP submodule is available on all eCAP modules; in this case, the high-resolution muxes and hardware are not implemented.

### 21.3.1 HRCAP Clocking

Unlike previous Type0 HRCAP modules, the Type1 eCAP, with HRCAP functionality, does not require a second PLL. However, the module still requires both SYSCLK and a second asynchronous clock source called HRCLK. The HRCLK is sensitive to changes in both temperature and voltage. For this reason, when using time-converted measurements, it is required to make periodic continuous calibrations.

### 21.3.2 HRCAP Initialization Sequence

To set up the HRCAP to make time-converted measurements:

1. Enable HRCLK using HRCAP_enableHighResolutionClock()
2. Delay 1uS
3. Configure the eCAP module as desired, including interrupts
4. Enable HR mode using HRCAP_enableHighResolution()
5. Set calibration period using HRCAP_setCalibrationPeriod()
6. Enable continuous calibration using HRCAP_setCalibrationMode()
7. Enable interrupts using HRCAP_enableCalibrationInterrupt()
8. Start calibration using HRCAP_startCalibration()

Steps 5-8 only apply for time-converted measurements. When using the HRCAP to take relative-time measurements only steps 1-4 are required.

### 21.3.3 HRCAP Interrupts

The HRCAP enhancements leverage the existing eCAP interrupts (see Section 20.5.7) in addition to HRCALINT which is used exclusively by the hardware calibration block. HRCALINT can be triggered by the following conditions:

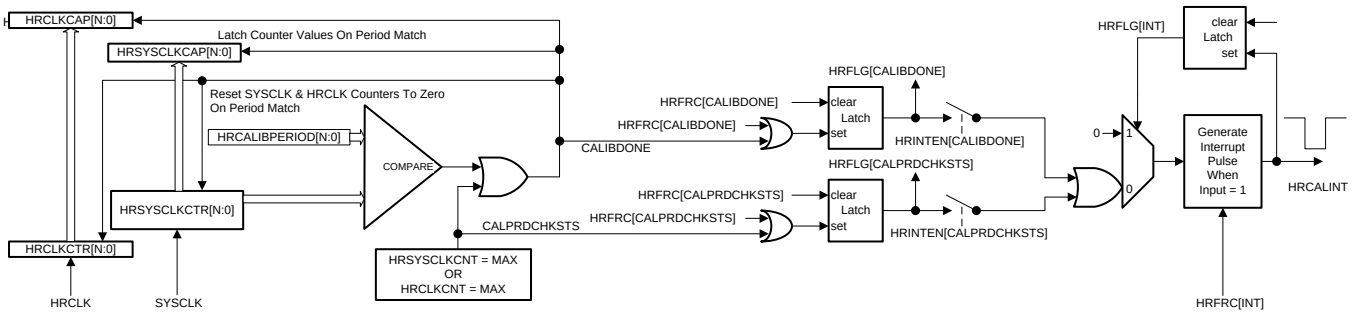
1. SYSCLKCTR = HRCALIBPERIOD
2. SYSCLKCTR or HRCLKCTR experience an overflow condition

Figure 21-2 shows this logic.

### 21.3.4 HRCAP Calibration

The following section only applies to time-converted measurements; calibration for relative-time measurements is not required. All values captured by the HRCAP submodule are in number of HRCLK cycles. The HRCLK speed varies widely with temperature and voltage, thus a scale factor is required to convert the capture value to the SYSCLK domain. For the same reason, it is required to periodically recalculate the scale factor. The HRCAP submodule has a calibration block to reduce software overhead when calculating a scale factor between HRCLK and SYSCLK.

**Figure 21-2. HRCAP Calibration**




The calibration block contains the following key resources:

- HRSYSCLKCNT
  - A 32-bit counter connected to SYSCLK. The counter will start counting when CALIBSTART is set.
- HRCLKCNT
  - A 32-bit counter connected to HRCLK. The counter will start counting when CALIBSTART is set.
- HRCALIBPERIOD
  - Calibration period, calibration is stopped when HRSYSCLKCNT is equal to the value in this register.
- HRSYSCLKCAP
  - On a calibration period match, the value of HRSYSCLKCNT is captured into HRSYSCLKCAP
- HRCLKCAP
  - On a calibration period match, the value of HRCLKCNT is captured into HRCLKCAP
- HRCALINT
  - An interrupt that occurs on a calibration period match, or when one of the counter registers experiences an overflow condition.

The calibration logic consists of two free-running counters; one clocked by HRCLK(HRCLKCTR) and the other clocked by SYSCLK(HRSYSCLKCTR). When HRSYSCLKCTR is equal to HRCALIBPERIOD, the calibration block will capture and reset both counter values, then trigger an interrupt indicating a new scale factor is ready to be calculated. The scale factor can be found by dividing HRSYSCLKCAP by HRCLKCAP, see Equation 1. A DriverLib function, HRCAP_getScaleFactor, has been provided to determine the scale factor. This function should be called inside of the calibration interrupt service routine. If one of the counters experiences overflow, the CALPRDCHKSTS flag will be set. The full details of the calibration block are described in Figure 21-2.

$$ScaleFactor = \frac{HRSYSCLKCAP}{HRCLKCAP} \quad (2)$$

**NOTE:** Even with calibration, noise on the 1.2V VDD supply will negatively affect the standard deviation of the HRCAP submodule. Care should be taken to ensure that the 1.2V supply is clean, and that noisy internal events such as enabling and disabling clock trees have been minimized while using the HRCAP.

#### 21.3.4.1 Applying the Scale Factor

A DriverLib function has been provided to apply the scale factor to a capture value,HRCAP_getEventTimeStampNanoseconds(). Equation 2 shows how to convert a raw count to seconds without using the DriverLib function.

$$Measurement(nS) = \frac{RawCount * scaleFactor}{128 * SysClk\,\mathrm{Prd}(nS)} \quad (3)$$

**Table 21-1. Scale Factor**

| Parameter | Typical value | Explanation |
|---|---|---|
| RawCount | 9300 | Capture value as read from ECAP_REGS_CAP1-4 |
| ScaleFactor | 2.75 | The Scale factor as calculated from Equation 2 |
| 128 | 128 | Constant determined by the hardware of the HRCAP submodule |
| SysClkPrd(nS) | 10 | Period of the system clock |
| Measurement(nS) | 1998.04 | Signal converted to nS |

### *21.3.5 DriverLib Functions*

## 21.4 Known Exceptions

In HRCAP mode:

- Enabling and disabling core clocks negatively affects the standard deviation of the HRCAP submodule. Care should be taken to avoid enabling and disabling core clocks while taking measurements.
- TSCTR is not writable, however it can be reset using ECCTL2[CTRFILTRESET]
- Input synchronization is not applicable when using the HRCAP enhancements, as the HRCAP submodule is asychronous to SYSCLK.
- The Event Filter functionality is not applicable for HRCAP, this defeats the purpose of HRCAP as the Event Filter's output is synchronous to SYSCLK.
- It is recommended to use absolute time mode for high resolution mode. If time difference mode is used, it may lead to inaccurate results if the fractional value is not taken into consideration for capture events which have reset the time base counter.
  - Actual Capture Value = (Capture Value) – (fractional value of reference event which reset the counter)
- For high frequency input signals the CPU may not be able cope with the speed of the captures. In such a case One-Shot mode is recommended, this allows the device to capture up to four edges before waiting to be serviced when the CPU is ready. This is applicable for the eCAP as well; however in that case the event filter can be used to reduce the rate of captures.

## 21.5 Registers

### 21.5.1 High-Resolution Capture Base Addresses

**Table 21-2. HRCAP Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| Hrcap6Regs | HRCAP_REGS | 0x0000_5360 | 0x0000_537F |
| Hrcap7Regs | HRCAP_REGS | 0x0000_53A0 | 0x0000_53BF |

### 21.5.1.1 HRCAP_REGS Registers

Table 21-3 lists the memory-mapped registers for the HRCAP_REGS. All register offset addresses not listed in Table 21-3 should be considered as reserved locations and the register contents should not be modified.

**Table 21-3. HRCAP_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | HRCTL | High-Res Control Register | EALLOW | Go |
| 4h | HRINTEN | High-Res Calibration Interrupt Enable Register | EALLOW | Go |
| 6h | HRFLG | High-Res Calibration Interrupt Flag Register | | Go |
| 8h | HRCLR | High-Res Calibration Interrupt Clear Register | | Go |
| Ah | HRFRC | High-Res Calibration Interrupt Force Register | EALLOW | Go |
| Ch | HRCALPRD | High-Res Calibration Period Register | EALLOW | Go |
| Eh | HRSYSCLKCTR | High-Res Calibration SYSCLK Counter Register | | Go |
| 10h | HRSYSCLKCAP | High-Res Calibration SYSCLK Capture Register | | Go |
| 12h | HRCLKCTR | High-Res Calibration HRCLK Counter Register | | Go |
| 14h | HRCLKCAP | High-Res Calibration HRCLK Capture Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 21-4 shows the codes that are used for access types in this section.

**Table 21-4. HRCAP_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| RW=1 | R<br>W=1 | Read |
| **Write Type** | | |
| W | W | Write |
| W=1 | W | Write |
| WR=0 | R=0<br>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. |

#### 21.5.1.1.1 HRCTL Register (Offset = 0h) [reset = 0h]

HRCTL is shown in Figure 21-3 and described in Table 21-5.

Return to Summary Table.

High-Res Control Register

**Figure 21-3. HRCTL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R/W-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R/W-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | CALIBCONT | CALIBSTS | CALIBSTART | PRDSEL | HRCLKE | HRE |
| R/W-0h | | R/W-0h | R=0-0h | RW=1/WR=0-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 21-5. HRCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R/W | 0h | Reserved |
| 5 | CALIBCONT | R/W | 0h | Continuous mode Calibration Select Bit:<br>0 Continuous mode disabled.<br>1 Continuous mode enabled. Calibration automatically restarts at end of current calibration cycle.<br>Reset type: CPU1.SYSRSn |
| 4 | CALIBSTS | R=0 | 0h | Calibration status Bit:<br>0 No active calibration cycle<br>1 Calibration cycle in progress<br>Reset type: CPU1.SYSRSn |
| 3 | CALIBSTART | RW=1/WR=0 | 0h | Calibration start Bit:<br>0 No effect<br>1 Starts the calibration cycle<br>Reset type: CPU1.SYSRSn |
| 2 | PRDSEL | R/W | 0h | Calibration Period Match Select Bit:<br>0 Use SYSCLK Counter For Period Match (default at reset)<br>1 Reserved<br>Reset type: CPU1.SYSRSn |
| 1 | HRCLKE | R/W | 0h | High Resolution Clock Enable Bit:<br>0 High resolution clock disabled (default at reset)<br>1 High resolution clock enabled. The clock should be enabled before enabling the high res function via the HRE bit.<br>Reset type: CPU1.SYSRSn |

**Table 21-5. HRCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | HRE | R/W | 0h | High Resolution Enable Bit:<br>0 High resolution mode disabled (default at reset)<br>1 High resolution mode enabled. Enabling this mode will connect the capture registers and edge event modes of the ECAP to be accessed by the High Res function.<br>Note: The High Res clock needs to be enabled (using the HRCLKE bit) first before enabling the module. Allow a certain start up stabilization period before enabling the module.<br>Reset type: CPU1.SYSRSn |

#### 21.5.1.1.2 HRINTEN Register (Offset = 4h) [reset = 0h]

HRINTEN is shown in Figure 21-4 and described in Table 21-6.

Return to Summary Table.

High-Res Calibration Interrupt Enable Register

**Figure 21-4. HRINTEN Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | CALPRDCHKSTS | CALIBDONE | RESERVED |
| R=0-0h | | | | | R/W-0h | R/W-0h | R=0-0h |

**Table 21-6. HRINTEN Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-3 | RESERVED | R=0 | 0h | Reserved |
| 2 | CALPRDCHKSTS | R/W | 0h | Calibration Period Check status Interrupt Enable:<br>0 Disable Calibration Period Check interrupt status<br>1 Enable Calibration Period Check interrupt status<br>Reset type: CPU1.SYSRSn |
| 1 | CALIBDONE | R/W | 0h | Calibration done Interrupt Enable:<br>0 Disable Calibration done Interrupt<br>1 Enable Calibration done Interrupt<br>Reset type: CPU1.SYSRSn |
| 0 | RESERVED | R=0 | 0h | Reserved |

#### 21.5.1.1.3 HRFLG Register (Offset = 6h) [reset = 0h]

HRFLG is shown in Figure 21-5 and described in Table 21-7.

Return to Summary Table.

High-Res Calibration Interrupt Flag Register

**Figure 21-5. HRFLG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | CALPRDCHKSTS | CALIBDONE | CALIBINT |
| R=0-0h | | | | | R-0h | R-0h | R-0h |

**Table 21-7. HRFLG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-3 | RESERVED | R=0 | 0h | Reserved |
| 2 | CALPRDCHKSTS | R | 0h | Calibration period check status Flag Bit:<br>1 Indicates that calibration ended before PRDCHK due to overflow on one of the counters.<br>0 Indicates no event occurred.<br>Note: This bit remains latched until cleared by the user using the HRCLR [CALPRDCHKSTS] bit.<br>Reset type: CPU1.SYSRSn |
| 1 | CALIBDONE | R | 0h | Calibration Done Interrupt Flag Bit:<br>1 Indicates calibration cycle is completed<br>0 Indicates calibration cycle has not completed.<br>Note: This bit remains latched until cleared by the user using the HRCLR [CALIBDONE] bit.<br>Reset type: CPU1.SYSRSn |
| 0 | CALIBINT | R | 0h | Global calibration Interrupt Status Flag:<br>1 Indicates that an interrupt was generated from CALIBDONE or CALPRDCHKSTS.<br>0 Indicates no interrupt generated.<br>Reset type: CPU1.SYSRSn |

#### 21.5.1.1.4 HRCLR Register (Offset = 8h) [reset = 0h]

HRCLR is shown in Figure 21-6 and described in Table 21-8.

Return to Summary Table.

High-Res Calibration Interrupt Clear Register

**Figure 21-6. HRCLR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | CALPRDCHKSTS | CALIBDONE | CALIBINT |
| R=0-0h | | | | | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 21-8. HRCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-3 | RESERVED | R=0 | 0h | Reserved |
| 2 | CALPRDCHKSTS | R=0/W=1 | 0h | Clear Calibration period check status Flag Bit:<br>1 Clears the CALPRDCHKSTS flag register bit.<br>0 No effect.<br>Note: H/W has priority over CPU writes if the user tries to clear a flag bit and an event occurs on the same cycle that tries to set the flag for the selected bit.<br>Reset type: CPU1.SYSRSn |
| 1 | CALIBDONE | R=0/W=1 | 0h | Clear Calibration Done Interrupt Flag Bit:<br>1 Clears the CALIBDONE interrupt flag register bit.<br>0 No effect.<br>Note: H/W has priority over CPU writes if the user tries to clear a flag bit and an event occurs on the same cycle that tries to set the flag for the selected bit.<br>Reset type: CPU1.SYSRSn |
| 0 | CALIBINT | R=0/W=1 | 0h | Clear Global calibration Interrupt Flag<br>1 Clears the Global interrupt flag and enables further interrupts to be generated if any of the event flags are set.<br>0 No effect.<br>Reset type: CPU1.SYSRSn |

#### 21.5.1.1.5 HRFRC Register (Offset = Ah) [reset = 0h]

HRFRC is shown in Figure 21-7 and described in Table 21-9.

Return to Summary Table.

High-Res Calibration Interrupt Force Register

**Figure 21-7. HRFRC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | CALPRDCHKSTS | CALIBDONE | RESERVED |
| R=0-0h | | | | | R=0/W=1-0h | R=0/W=1-0h | R=0-0h |

**Table 21-9. HRFRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-3 | RESERVED | R=0 | 0h | Reserved |
| 2 | CALPRDCHKSTS | R=0/W=1 | 0h | Force CALPRDCHKSTS flag:<br>0 No effect<br>1 Sets the CALPRDCHKSTS flag.<br>Reset type: CPU1.SYSRSn |
| 1 | CALIBDONE | R=0/W=1 | 0h | Force CALIBDONE flag:<br>0 No effect<br>1 Sets the CALIBDONE flag.<br>Reset type: CPU1.SYSRSn |
| 0 | RESERVED | R=0 | 0h | Reserved |

#### 21.5.1.1.6 HRCALPRD Register (Offset = Ch) [reset = 003FFFFFh]

HRCALPRD is shown in Figure 21-8 and described in Table 21-10.

Return to Summary Table.

High-Res Calibration Period Register

**Figure 21-8. HRCALPRD Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRD | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-003FFFFFh | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 21-10. HRCALPRD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | PRD | R/W | 003FFFFFh | Register to program calibration period. The period value is matched against HRSYSCLKCTR. On a match an interrupt is generated and the counter registers values are captured.<br>Reset type: CPU1.SYSRSn |

#### 21.5.1.1.7 HRSYSCLKCTR Register (Offset = Eh) [reset = 0h]

HRSYSCLKCTR is shown in Figure 21-9 and described in Table 21-11.

Return to Summary Table.

High-Res Calibration SYSCLK Counter Register

**Figure 21-9. HRSYSCLKCTR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HRSYSCLKCTR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 21-11. HRSYSCLKCTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | HRSYSCLKCTR | R | 0h | Current SYSCLK counter value<br>Reset type: CPU1.SYSRSn |

#### 21.5.1.1.8 HRSYSCLKCAP Register (Offset = 10h) [reset = 0h]

HRSYSCLKCAP is shown in Figure 21-10 and described in Table 21-12.

Return to Summary Table.

High-Res Calibration SYSCLK Capture Register

**Figure 21-10. HRSYSCLKCAP Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HRSYSCLKCAP | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 21-12. HRSYSCLKCAP Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | HRSYSCLKCAP | R/W | 0h | HRSYSCLKCTR is captures into this register at end of calibration cycle.<br>Reset type: CPU1.SYSRSn |

#### 21.5.1.1.9 HRCLKCTR Register (Offset = 12h) [reset = 0h]

HRCLKCTR is shown in Figure 21-11 and described in Table 21-13.

Return to Summary Table.

High-Res Calibration HRCLK Counter Register

**Figure 21-11. HRCLKCTR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HRCLKCTR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 21-13. HRCLKCTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | HRCLKCTR | R | 0h | Current HRCLK counter value<br>Reset type: CPU1.SYSRSn |

#### 21.5.1.1.10 HRCLKCAP Register (Offset = 14h) [reset = 0h]

HRCLKCAP is shown in Figure 21-12 and described in Table 21-14.

Return to Summary Table.

High-Res Calibration HRCLK Capture Register

**Figure 21-12. HRCLKCAP Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HRCLKCAP | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 21-14. HRCLKCAP Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | HRCLKCAP | R/W | 0h | HRCLKCTR is captures into this register at end of calibration cycle.<br>Reset type: CPU1.SYSRSn |

# Enhanced QEP (eQEP)

The enhanced QEP (eQEP) module described here is a Type-1 eQEP. See the *TMS320x28xx, 28xxx DSP Peripheral Reference Guide* (SPRU566) for a list of all devices with a module of the same type to determine the differences between types and for a list of device-specific differences within a type.

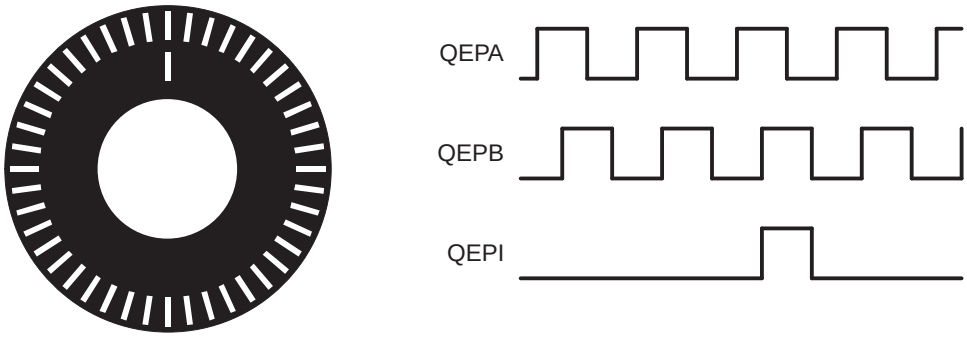
The enhanced quadrature encoder pulse (eQEP) module is used for direct interface with a linear or rotary incremental encoder to get position, direction, and speed information from a rotating machine for use in a high-performance motion and position-control system.

**Topic** **Page**

## 22.1 Introduction

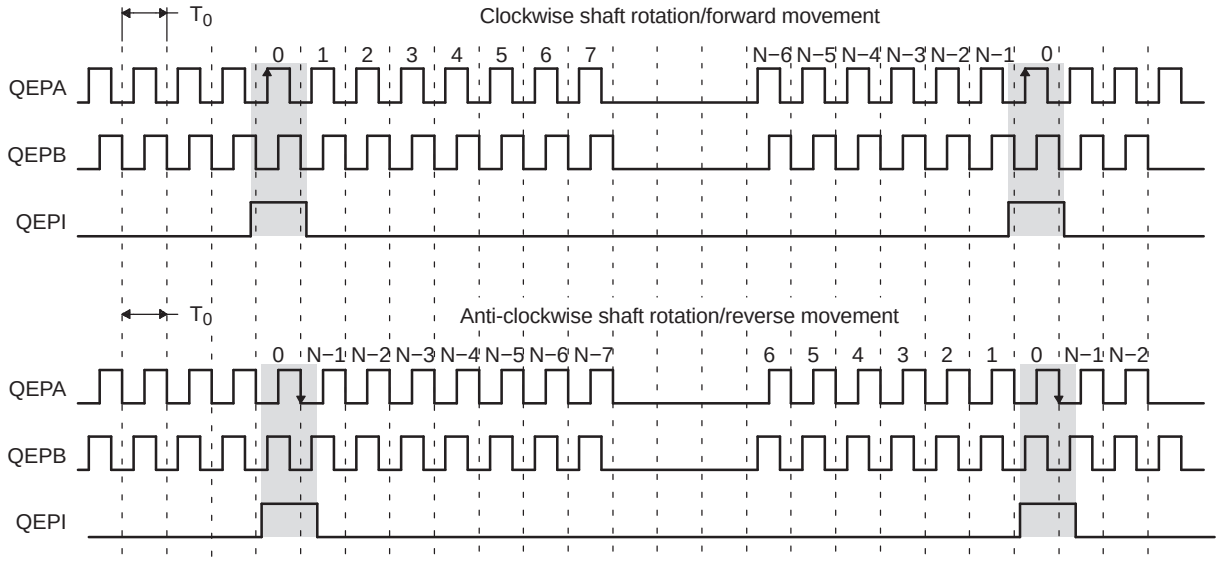
A single track of slots patterns the periphery of an incremental encoder disk, as shown in Figure 22-1. These slots create an alternating pattern of dark and light lines. The disk count is defined as the number of dark/light line pairs that occur per revolution (lines per revolution). As a rule, a second track is added to generate a signal that occurs once per revolution (index signal: QEPI), which can be used to indicate an absolute position. Encoder manufacturers identify the index pulse using different terms such as index, marker, home position, and zero reference

**Figure 22-1. Optical Encoder Disk**




To derive direction information, the lines on the disk are read out by two different photo-elements that "look" at the disk pattern with a mechanical shift of 1/4 the pitch of a line pair between them. This shift is realized with a reticle or mask that restricts the view of the photo-element to the desired part of the disk lines. As the disk rotates, the two photo-elements generate signals that are shifted 90° out of phase from each other. These are commonly called the quadrature QEPA and QEPB signals. The clockwise direction for most encoders is defined as the QEPA channel going positive before the QEPB channel and vise versa as shown in Figure 22-2.

**Figure 22-2. QEP Encoder Output Signal for Forward/Reverse Movement**

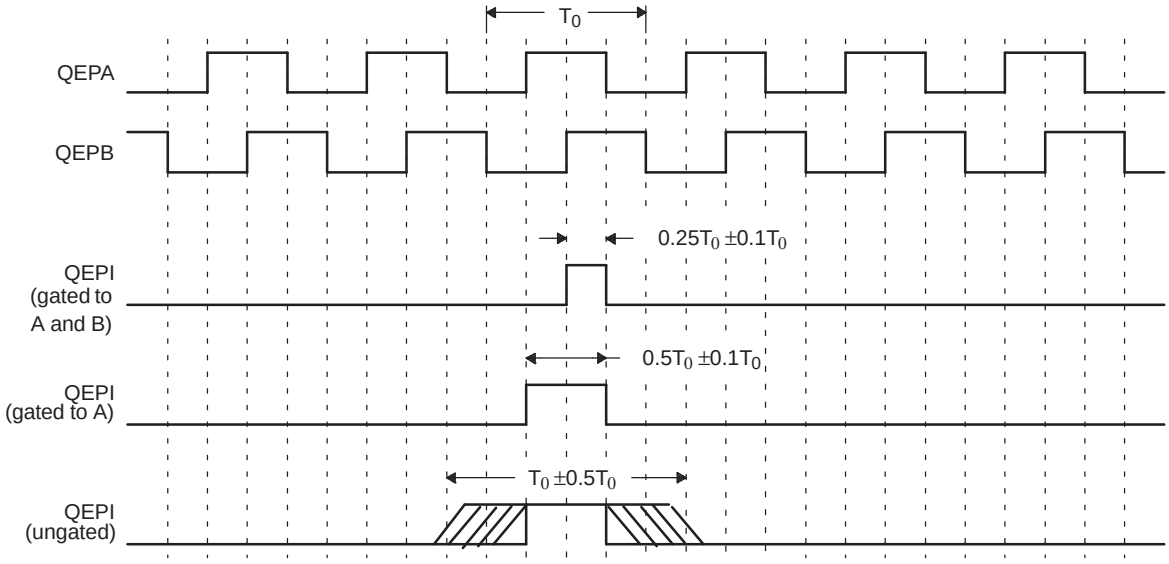



**Legend:** N = lines per revolution

The encoder wheel typically makes one revolution for every revolution of the motor or the wheel may be at a geared rotation ratio with respect to the motor. Therefore, the frequency of the digital signal coming from the QEPA and QEPB outputs varies proportionally with the velocity of the motor. For example, a 2000-line encoder directly coupled to a motor running at 5000 revolutions per minute (rpm) results in a frequency of 166.6 KHz, so by measuring the frequency of either the QEPA or QEPB output, the processor can determine the velocity of the motor.

Quadrature encoders from different manufacturers come with two forms of index pulse (gated index pulse or ungated index pulse) as shown in Figure 22-3. 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 22-3. Index Pulse Example**




Some typical applications of shaft encoders include robotics and even computer input in the form of a mouse. Inside your mouse you can see where the mouse ball spins a pair of axles (a left/right, and an up/down axle). These axles are connected to optical shaft encoders that effectively tell the computer how fast and in what direction the mouse is moving.

**General Issues:** Estimating velocity from a digital position sensor is a cost-effective strategy in motor control. Two different first order approximations for velocity may be written as:

$$v(k) \approx \frac{x(k) - x(k-1)}{T} = \frac{\Delta X}{T} \quad (4)$$

$$v(k) \approx \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T} \quad (5)$$

where

v(k): Velocity at time instant k

x(k): Position at time instant k

x(k-1): Position at time instant k-1

T: Fixed unit time or inverse of velocity calculation rate

ΔX: Incremental position movement in unit time

t(k): Time instant "k"

t(k-1): Time instant "k-1"

X: Fixed unit position

ΔT: Incremental time elapsed for unit position movement.

Equation 4 is the conventional approach to velocity estimation and it requires a time base to provide unit time event for velocity calculation. Unit time is basically the inverse of the velocity calculation rate.

The encoder count (position) is read once during each unit time event. The quantity [x(k) - x(k-1)] is formed by subtracting the previous reading from the current reading. Then the velocity estimate is computed by multiplying by the known constant 1/T (where T is the constant time between unit time events and is known in advance).

Estimation based on Equation 4 has an inherent accuracy limit directly related to the resolution of the position sensor and the unit time period T. For example, consider a 500-line per revolution quadrature encoder with a velocity calculation rate of 400 Hz. When used for position the quadrature encoder gives a four-fold increase in resolution, in this case, 2000 counts per revolution. The minimum rotation that can be detected is therefore 0.0005 revolutions, which gives a velocity resolution of 12 rpm when sampled at 400 Hz. While this resolution may be satisfactory at moderate or high speeds, e.g. 1% error at 1200 rpm, it would clearly prove inadequate at low speeds. In fact, at speeds below 12 rpm, the speed estimate would erroneously be zero much of the time.

At low speed, Equation 5 provides a more accurate approach. It requires a position sensor that outputs a fixed interval pulse train, such as the aforementioned quadrature encoder. The width of each pulse is defined by motor speed for a given sensor resolution. Equation 5 can be used to calculate motor speed by measuring the elapsed time between successive quadrature pulse edges. However, this method suffers from the opposite limitation, as does Equation 4. A combination of relatively large motor speeds and high sensor resolution makes the time interval $\Delta T$ small, and thus more greatly influenced by the timer resolution. This can introduce considerable error into high-speed estimates.

For systems with a large speed range (that is, speed estimation is needed at both low and high speeds), one approach is to use Equation 5 at low speed and have the DSP software switch over to Equation 4 when the motor speed rises above some specified threshold.

## 22.2 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

For proper operation of the eQEP module, input GPIO pins must be configured via the GPxQSELn registers for synchronous input mode (with or without qualification). The asynchronous mode should not be used for eQEP input pins. The internal pullups can be configured in the GPyPUD register.

See the *GPIO* chapter for more details on GPIO mux and settings.

## 22.3 Description

This section provides the eQEP inputs, memory map, and functional description.

### *22.3.1 EQEP Inputs*

The eQEP inputs include two pins for quadrature-clock mode or direction-count mode, an index (or 0 marker), and a strobe input. The eQEP module requires that the QEPA, QEPB, and QEPI inputs are synchronized to SYSCLK prior to entering the module. The application code should enable the synchronous GPIO input feature on any eQEP-enabled GPIO pins (see the *System Control and Interrupts*chapter for more details).

- *QEPA/XCLK and QEPB/XDIR*

  These two pins can be used in quadrature-clock mode or direction-count mode.

  – *Quadrature-clock Mode*

    The eQEP encoders provide two square wave signals (A and B) 90 electrical degrees out of phase whose phase relationship is used to determine the direction of rotation of the input shaft and number of eQEP pulses from the index position to derive the relative position information. For forward or clockwise rotation, QEPA signal leads QEPB signal and vice versa. The quadrature decoder uses these two inputs to generate quadrature-clock and direction signals.

  – *Direction-count Mode*

    In direction-count mode, direction and clock signals are provided directly from the external source. Some position encoders have this type of output instead of quadrature output. The QEPA pin provides the clock input and the QEPB pin provides the direction input.

- *QEPI: Index or Zero Marker*

  The eQEP encoder uses an index signal to assign an absolute start position from which position information is incrementally encoded using quadrature pulses. This pin is connected to the index output of the eQEP encoder to optionally reset the position counter for each revolution. This signal can be used to initialize or latch the position counter on the occurrence of a desired event on the index pin.

- QEPS: *Strobe Input*

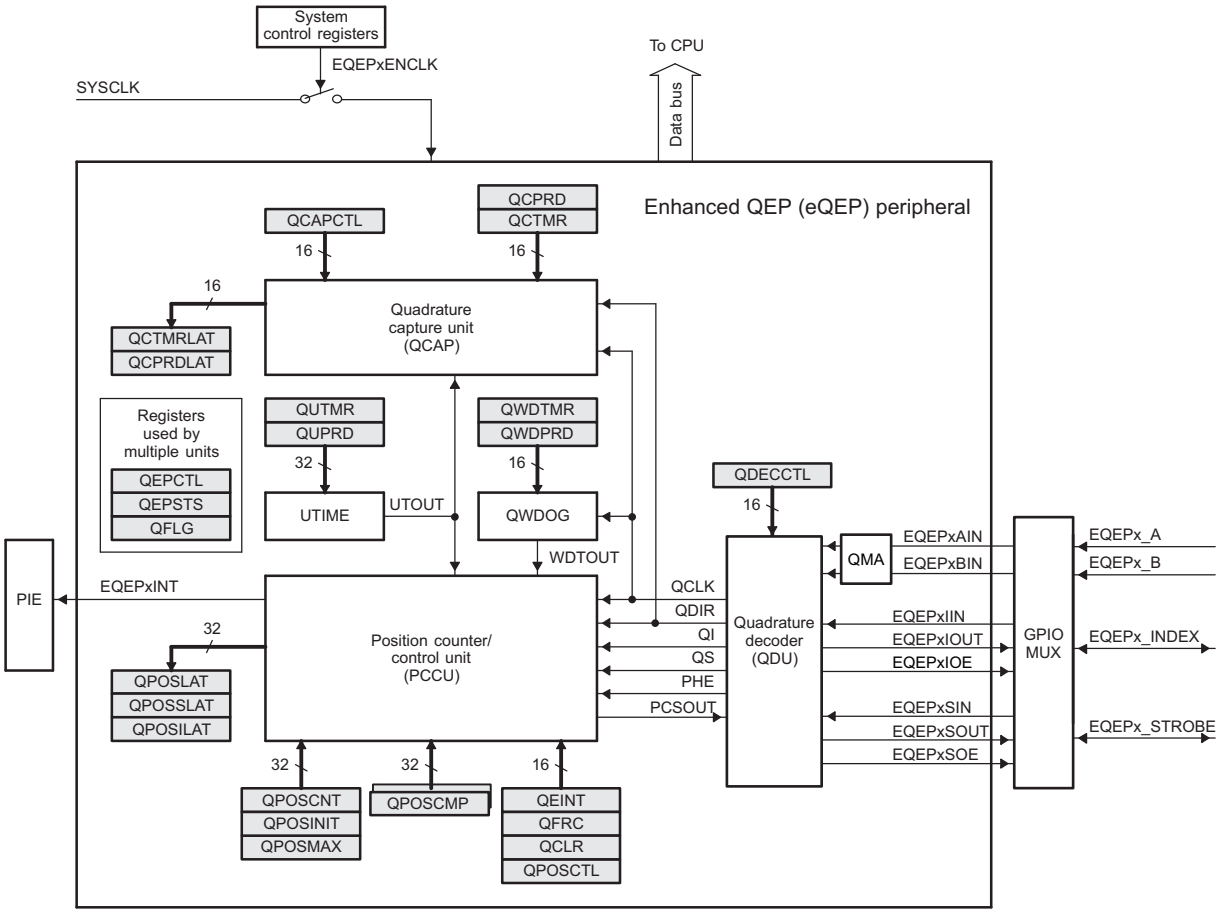
  This general-purpose strobe signal can initialize or latch the position counter on the occurrence of a desired event on the strobe pin. This signal is typically connected to a sensor or limit switch to notify that the motor has reached a defined position.

### 22.3.2 Functional Description

The eQEP peripheral contains the following major functional units (as shown in Figure 22-4):

- 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 22-4. Functional Block Diagram of the eQEP Peripheral**



### 22.3.3 eQEP Memory Map

Table 22-1 lists the registers with their memory locations, sizes, and reset values.

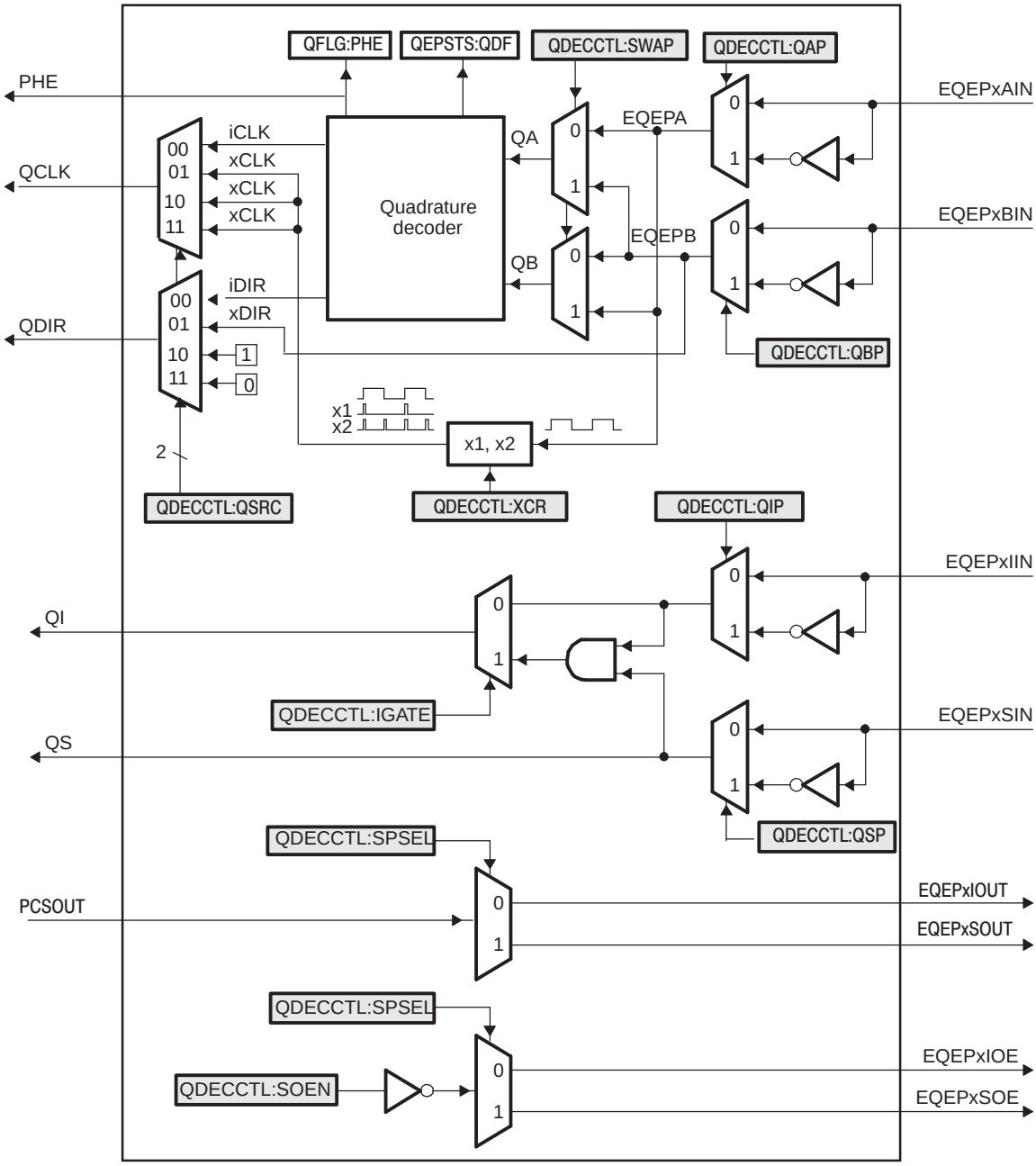
**Table 22-1. EQEP Memory Map**

| Name | Offset | Size(x16)/ #shadow | Reset | Register Description |
|---|---|---|---|---|
| QPOSCNT | 0x00 | 2/0 | 0x00000000 | eQEP Position Counter |
| QPOSINIT | 0x02 | 2/0 | 0x00000000 | eQEP Initialization Position Count |
| QPOSMAX | 0x04 | 2/0 | 0x00000000 | eQEP Maximum Position Count |
| QPOSCMP | 0x06 | 2/1 | 0x00000000 | eQEP Position-compare |
| QPOSILAT | 0x08 | 2/0 | 0x00000000 | eQEP Index Position Latch |
| QPOSSLAT | 0x0A | 2/0 | 0x00000000 | eQEP Strobe Position Latch |
| QPOSLAT | 0x0C | 2/0 | 0x00000000 | eQEP Position Latch |
| QUTMR | 0x0E | 2/0 | 0x00000000 | QEP Unit Timer |
| QUPRD | 0x10 | 2/0 | 0x00000000 | eQEP Unit Period Register |
| QWDTMR | 0x12 | 1/0 | 0x0000 | eQEP Watchdog Timer |
| QWDPRD | 0x13 | 1/0 | 0x0000 | eQEP Watchdog Period Register |
| QDECCTL | 0x14 | 1/0 | 0x0000 | eQEP Decoder Control Register |
| QEPCTL | 0x15 | 1/0 | 0x0000 | eQEP Control Register |
| QCAPCTL | 0x16 | 1/0 | 0x0000 | eQEP Capture Control Register |
| QPOSCTL | 0x17 | 1/0 | 0x00000 | eQEP Position-compare Control Register |
| QEINT | 0x18 | 1/0 | 0x0000 | eQEP Interrupt Enable Register |
| QFLG | 0x19 | 1/0 | 0x0000 | eQEP Interrupt Flag Register |
| QCLR | 0x1A | 1/0 | 0x0000 | eQEP Interrupt Clear Register |
| QFRC | 0x1B | 1/0 | 0x0000 | eQEP Interrupt Force Register |
| QEPSTS | 0x1C | 1/0 | 0x0000 | eQEP Status Register |
| QCTMR | 0x1D | 1/0 | 0x0000 | eQEP Capture Timer |
| QCPRD | 0x1E | 1/0 | 0x0000 | eQEP Capture Period Register |
| QCTMRLAT | 0x1F | 1/0 | 0x0000 | eQEP Capture Timer Latch |
| QCPRDLAT | 0x20 | 1/0 | 0x0000 | eQEP Capture Period Latch |
| reserved | 0x21 to 0x3F | 31/0 | | |

## 22.4 Quadrature Decoder Unit (QDU)

Figure 22-5 shows a functional block diagram of the QDU.

**Figure 22-5. Functional Block Diagram of Decoder Unit**




### 22.4.1 Position Counter Input Modes

Clock and direction input to position counter is selected using QDECCTL[QSRC] bits, based on interface input requirement as follows:

- Quadrature-count mode
- Direction-count mode
- UP-count mode
- DOWN-count mode

### 22.4.1.1 Quadrature Count Mode

The quadrature decoder generates the direction and clock to the position counter in quadrature count mode.

**Direction Decoding—** The direction decoding logic of the eQEP circuit determines which one of the sequences (QEPA, QEPB) is the leading sequence and accordingly updates the direction information in QEPSTS[QDF] bit. Table 22-2 and Figure 22-6 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 22-7 shows the direction decoding and clock generation from the eQEP input signals.

**Table 22-2. Quadrature Decoder Truth Table**

| . | | | |
|---|---|---|---|
| **Previous Edge** | **Present Edge** | **QDIR** | **QPOSCNT** |
| QA↑ | QB↑ | UP | Increment |
| | QB↓ | DOWN | Decrement |
| | QA↓ | TOGGLE | Increment or Decrement |
| QA↓ | QB↓ | UP | Increment |
| | QB↑ | DOWN | Decrement |
| | QA↑ | TOGGLE | Increment or Decrement |
| QB↑ | QA↑ | DOWN | Increment |
| | QA↓ | UP | Decrement |
| | QB↓ | TOGGLE | Increment or Decrement |
| QB↓ | QA↓ | DOWN | Increment |
| | QA↑ | UP | Decrement |
| | QB↑ | TOGGLE | Increment or Decrement |

**Figure 22-6. Quadrature Decoder State Machine**

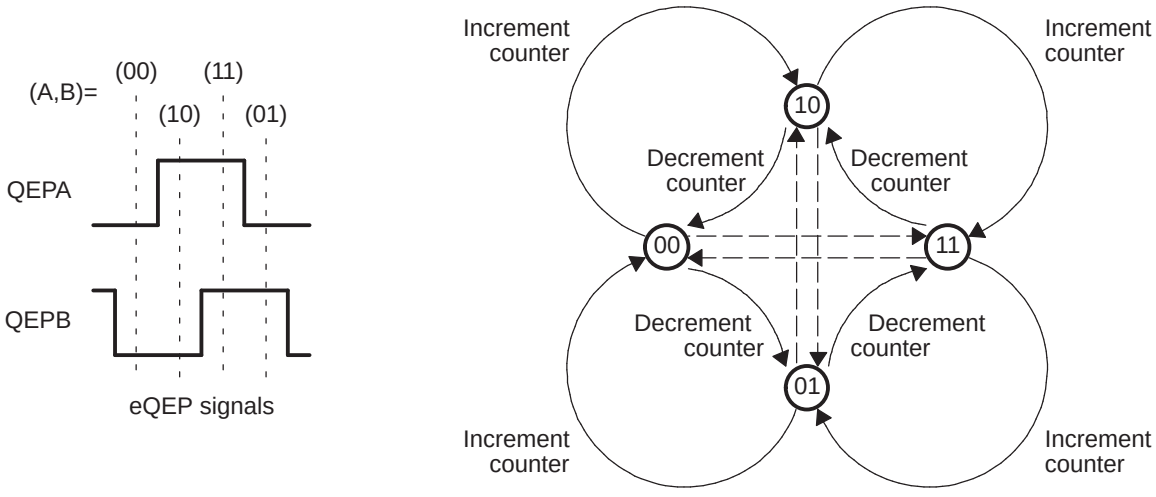
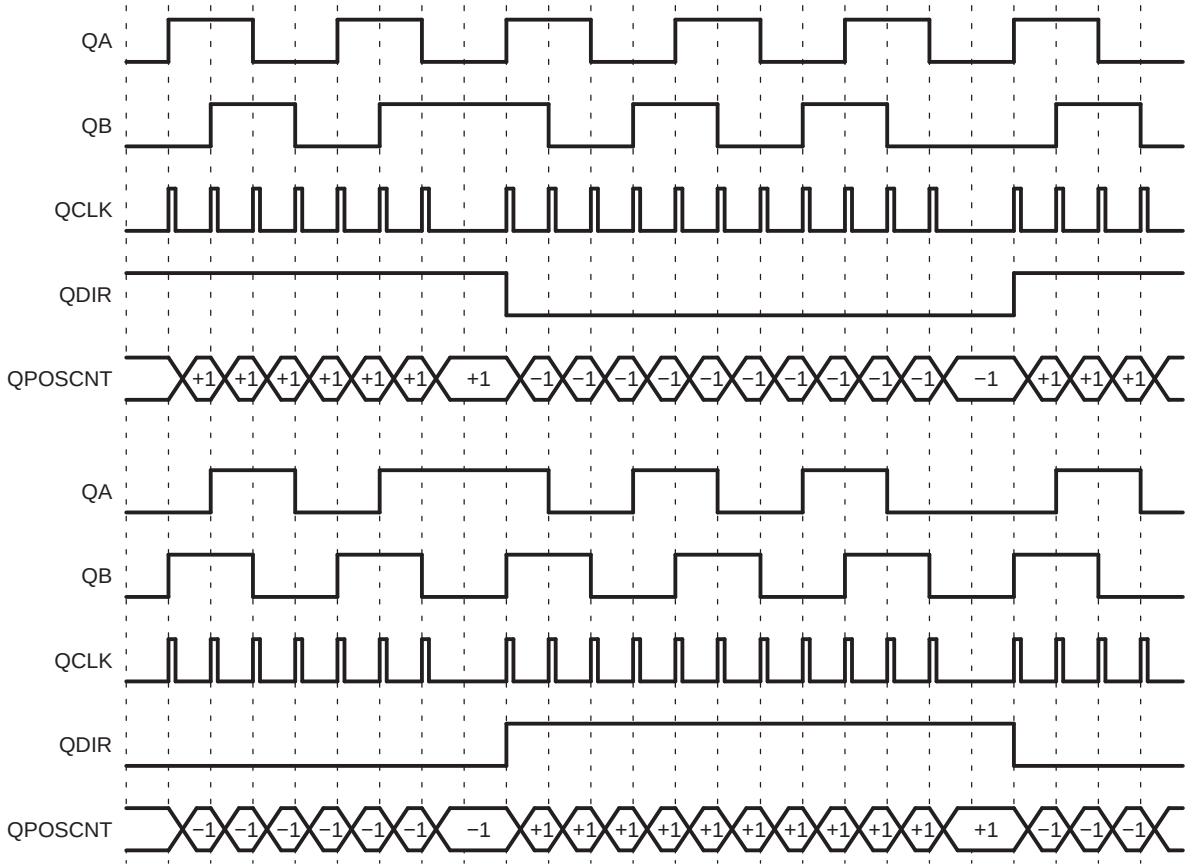


**Figure 22-7. Quadrature-clock and Direction Decoding**




**Phase Error Flag—** In normal operating conditions, quadrature inputs QEPA and QEPB will be 90 degrees out of phase. The phase error flag (PHE) is set in the QFLG register and QPOSCNT value can be incorrect and offset by multiples of 1 or 3, when edge transition is detected simultaneously on the QEPA and QEPB signals to optionally generate interrupts. State transitions marked by dashed lines in Figure 22-6 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 22-7 .

**Reverse Count—** In normal quadrature count operation, QEPA input is fed to the QA input of the quadrature decoder and the QEPB input is fed to the QB input of the quadrature decoder. Reverse counting is enabled by setting the SWAP bit in the QDECCTL register. This will swap the input to the quadrature decoder thereby reversing the counting direction.

### 22.4.1.2 Direction-count Mode

Some position encoders provide direction and clock outputs, instead of quadrature outputs. In such cases, direction-count mode can be used. QEPA input will provide the clock for position counter and the QEPB input will have the direction information. The position counter is incremented on every rising edge of a QEPA input when the direction input is high and decremented when the direction input is low.

#### 22.4.1.3 Up-Count Mode

The counter direction signal is hard-wired for up count and the position counter is used to measure the frequency of the QEPA input. Clearing the QDECCTL[XCR] bit enables clock generation to the position counter on both edges of the QEPA input, thereby increasing the measurement resolution by 2x factor. In up-count mode, it is recommended that the application not configure QEPB as a GPIO mux option, or ensure that a signal edge is not generated on the QEPB input.

#### 22.4.1.4 Down-Count Mode

The counter direction signal is hardwired for a down count and the position counter is used to measure the frequency of the QEPA input. Setting of the QDECCTL[XCR] bit enables clock generation to the position counter on both edges of a QEPA input, thereby increasing the measurement resolution by 2x factor. In down-count mode, it is recommended that the application not configure QEPB as a GPIO mux option, or ensure that a signal edge is not generated on the QEPB input.

### *22.4.2 eQEP Input Polarity Selection*

Each eQEP input can be inverted using QDECCTL[8:5] control bits. As an example, setting of QDECCTL[QIP] bit will invert the index input.

### *22.4.3 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.

## 22.5 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.

### *22.5.1 Position Counter Operating Modes*

Position counter data may be captured in different manners. In some systems, the position counter is accumulated continuously for multiple revolutions and the position counter value provides the position information with respect to the known reference. An example of this is the quadrature encoder mounted on the motor controlling the print head in the printer. Here the position counter is reset by moving the print head to the home position and then position counter provides absolute position information with respect to home position.

In other systems, the position counter is reset on every revolution using index pulse and position counter provides rotor angle with respect to index pulse position.

Position counter can be configured to operate in following four modes

- Position Counter Reset on Index Event
- Position Counter Reset on Maximum Position
- Position Counter Reset on the first Index Event
- Position Counter Reset on Unit Time Out Event (Frequency Measurement)

In all the above operating modes, position counter is reset to 0 on overflow and to QPOSMAX register value on underflow. Overflow occurs when the position counter counts up after QPOSMAX value. Underflow occurs when position counter counts down after "0". Interrupt flag is set to indicate overflow/underflow in QFLG register.

### 22.5.1.1 Position Counter Reset on Index Event (QEPCTL[PCRM]=00)

If the index event occurs during the forward movement, then position counter is reset to 0 on the next eQEP clock. If the index event occurs during the reverse movement, then the position counter is reset to the value in the QPOSMAX register on the next eQEP clock.

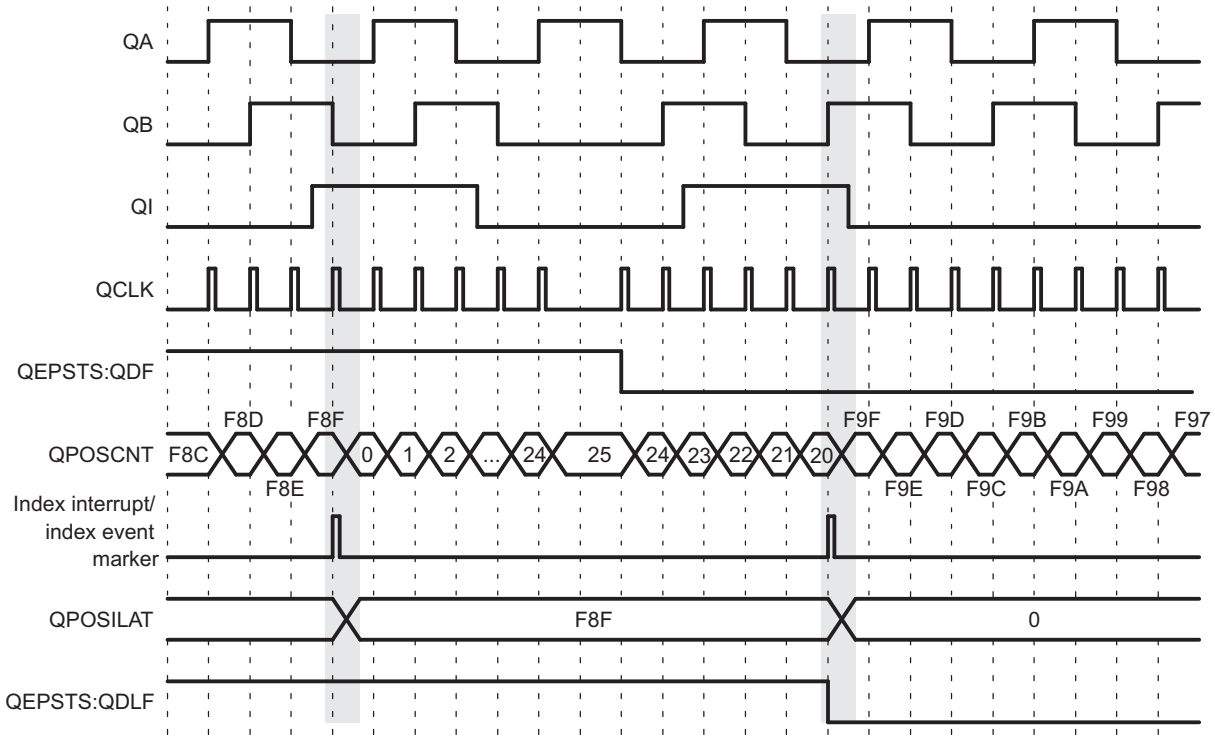
First index marker is defined as the quadrature edge following the first index edge. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and direction on the first index event marker (QEPSTS[FIDF]) in QEPSTS registers, it also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for index event reset operation.

For example, if the first reset operation occurs on the falling edge of QEPB during the forward direction, then all the subsequent reset must be aligned with the falling edge of QEPB for the forward rotation and on the rising edge of QEPB for the reverse rotation as shown in Figure 22-8.

The position-counter value is latched to the QPOSILAT register and direction information is recorded in the QEPSTS[QDLF] bit on every index event marker. The position-counter error flag (QEPSTS[PCEF]) and error interrupt flag (QFLG[PCE]) are set if the latched value is not equal to 0 or QPOSMAX. The position-counter error flag (QEPSTS[PCEF]) is updated on every index event marker and an interrupt flag (QFLG[PCE]) will be set on error that can be cleared only through software.

The index event latch configuration QEPCTL[IEL] must be configured to '00' or '11' when pcrm=0 and 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 22-8. Position Counter Reset by Index Pulse for 1000 Line Encoder (QPOSMAX = 3999 or 0xF9F)**

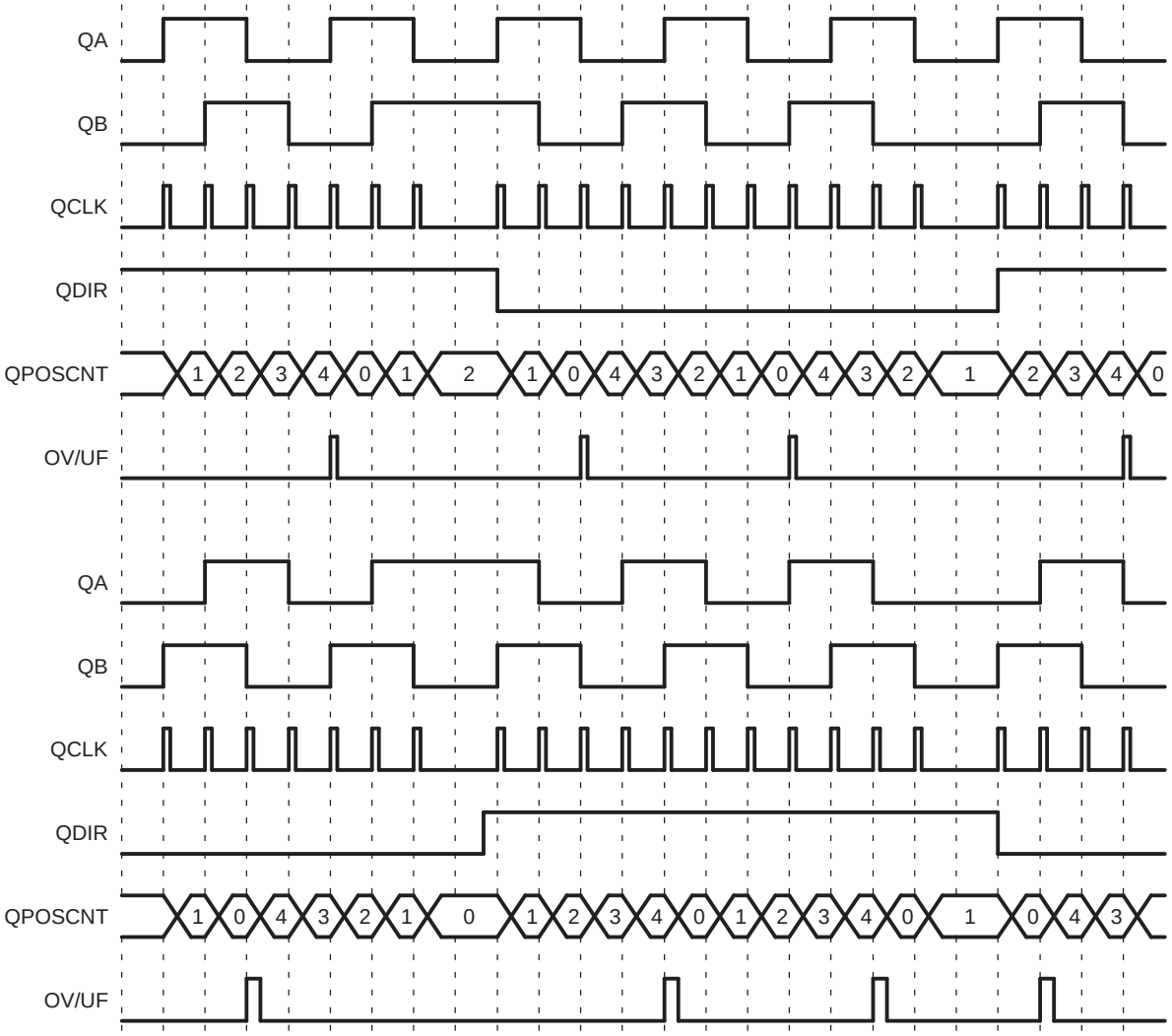



### 22.5.1.2 Position Counter Reset on Maximum Position (QEPCTL[PCRM]=01)

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 22-9 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 22-9. Position Counter Underflow/Overflow (QPOSMAX = 4)**




#### 22.5.1.3 Position Counter Reset on the First Index Event (QEPCTL[PCRM] = 10)

If the index event occurs during forward movement, then the position counter is reset to 0 on the next eQEP clock. If the index event occurs during the reverse movement, then the position counter is reset to the value in the QPOSMAX register on the next eQEP clock. Note that this is done only on the first occurrence and subsequently the position counter value is not reset on an index event; rather, it is reset based on maximum position as described in Section Section 22.5.1.2.

The first index marker fields (QEPSTS[FIDF] and QEPSTS[FIMF]) are not applicable in this mode.

#### 22.5.1.4 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.

### *22.5.2 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.

### 22.5.2.1 Index Event Latch

In some applications, it may not be desirable to reset the position counter on every index event and instead it may be required to operate the position counter in full 32-bit mode (QEPCTL[PCRM] = 01 and QEPCTL[PCRM] = 10 modes).

In such cases, the eQEP position counter can be configured to latch on the following events and direction information is recorded in the QEPSTS[QDLF] bit on every index event marker.

- Latch on Rising edge (QEPCTL[IEL]=01)
- Latch on Falling edge (QEPCTL[IEL]=10)
- Latch on Index Event Marker (QEPCTL[IEL]=11)

This is particularly useful as an error checking mechanism to check if the position counter accumulated the correct number of counts between index events. As an example, the 1000-line encoder must count 4000 times when moving in the same direction between the index events.

The index event latch interrupt flag (QFLG[IEL]) is set when the position counter is latched to the QPOSILAT register.

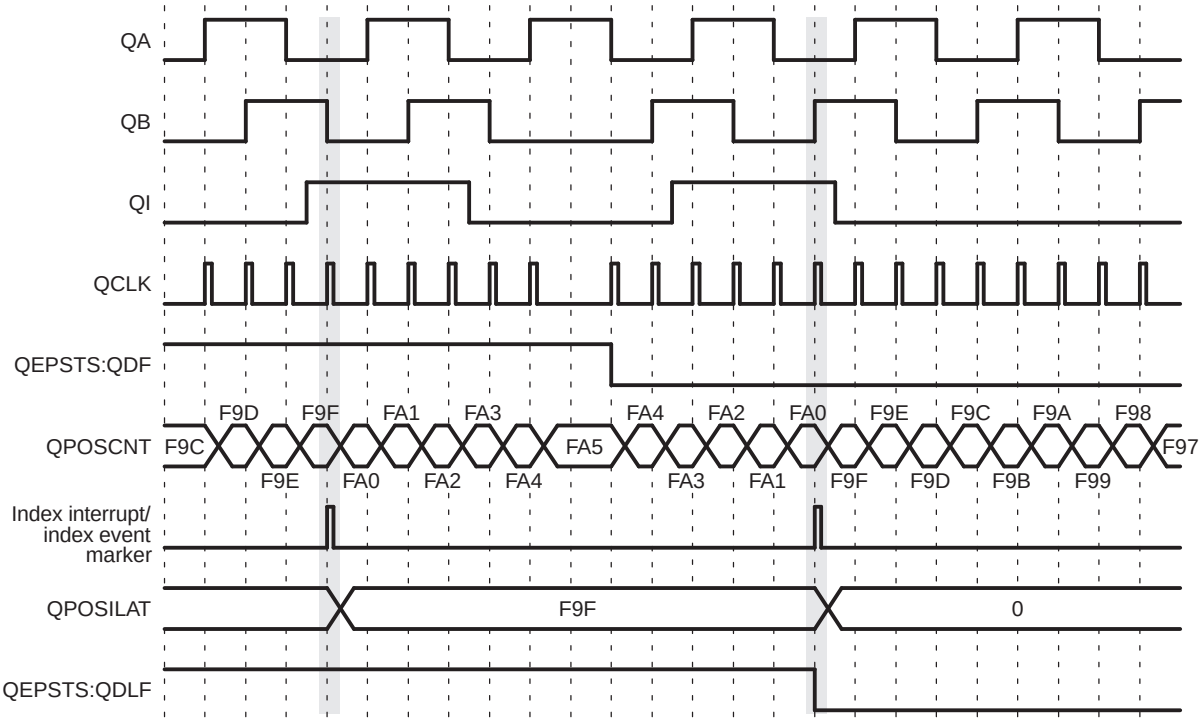
**Latch on Rising Edge (QEPCTL[IEL]=01)—** The position counter value (QPOSCNT) is latched to the QPOSILAT register on every rising edge of an index input.

**Latch on Falling Edge (QEPCTL[IEL] = 10)—** The position counter value (QPOSCNT) is latched to the QPOSILAT register on every falling edge of index input.

**Latch on Index Event Marker/Software Index Marker (QEPCTL[IEL] = 11—** The first index marker is defined as the quadrature edge following the first index edge. The eQEP peripheral records the occurrence of the first index marker (QEPSTS[FIMF]) and direction on the first index event marker (QEPSTS[FIDF]) in the QEPSTS registers. It also remembers the quadrature edge on the first index marker so that same relative quadrature transition is used for latching the position counter (QEPCTL[IEL]=11).

Figure 22-10 shows the position counter latch using an index event marker.

**Figure 22-10. Software Index Marker for 1000-line Encoder (QEPCTL[IEL] = 1)**



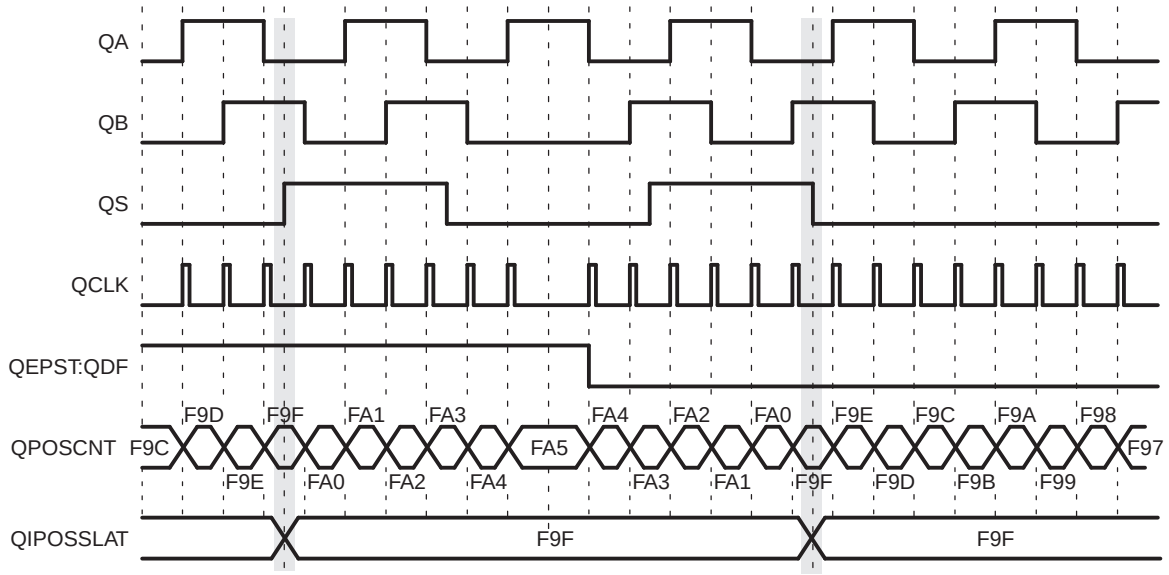
#### 22.5.2.2 Strobe Event Latch

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 22-11.

The strobe event latch interrupt flag (QFLG[SEL) is set when the position counter is latched to the QPOSSLAT register.

**Figure 22-11. Strobe Event Latch (QEPCTL[SEL] = 1)**




### *22.5.3 Position Counter Initialization*

The position counter can be initialized using following events:

- Index event
- Strobe event
- Software initialization

**Index Event Initialization (IEI)—** The QEPI index input can be used to trigger the initialization of the position counter at the rising or falling edge of the index input. If the QEPCTL[IEI] bits are 10, then the position counter (QPOSCNT) is initialized with a value in the QPOSINIT register on the rising edge of index input. Conversely, if the QEPCTL[IEI] bits are 11, initialization will be on the falling edge of the index input.

**Strobe Event Initialization (SEI)—** If the QEPCTL[SEI] bits are 10, then the position counter is initialized with a value in the QPOSINIT register on the rising edge of strobe input.

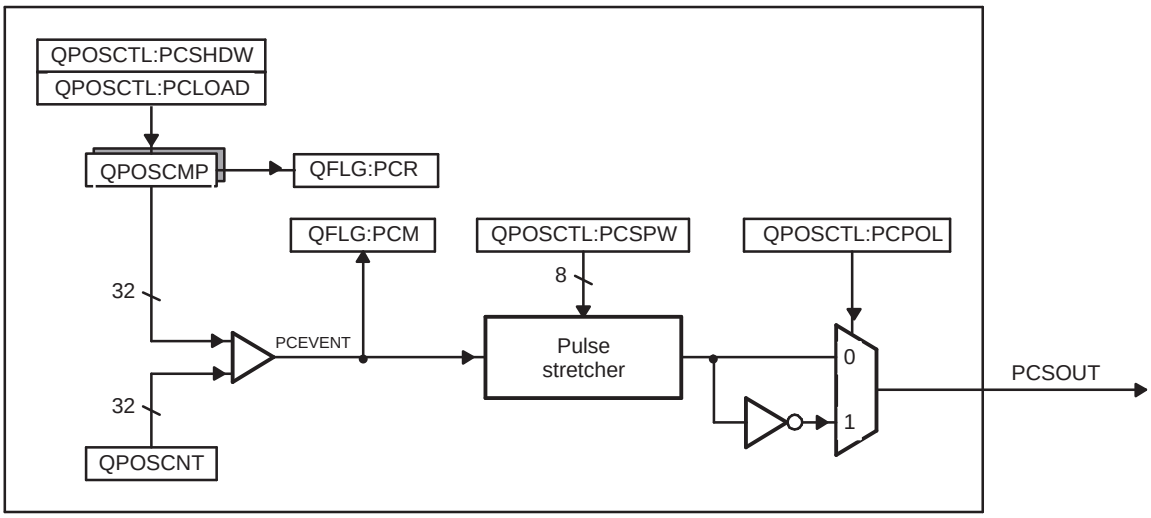
If QEPCTL[SEL] bits are 11, then the position counter is initialized with a value in the QPOSINIT register on the rising edge of strobe input for forward direction and on the falling edge of strobe input for reverse direction.

**Software Initialization (SWI)—** The position counter can be initialized in software by writing a 1 to the QEPCTL[SWI] bit. This bit is not automatically cleared. While the bit is still set, if a 1 is written to it again, the position counter will be re-initialized.

### 22.5.4 eQEP Position-compare Unit

The eQEP peripheral includes a position-compare unit that is used to generate a sync output and/or interrupt on a position-compare match. Figure 22-12 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 22-12. 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.

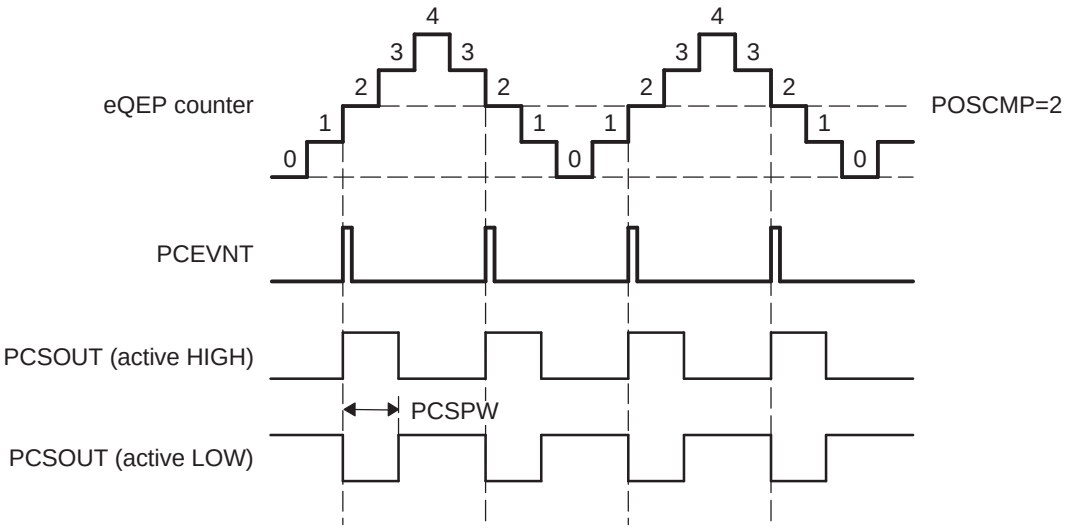
- Load on compare match
- Load on position-counter zero event

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 22-13).

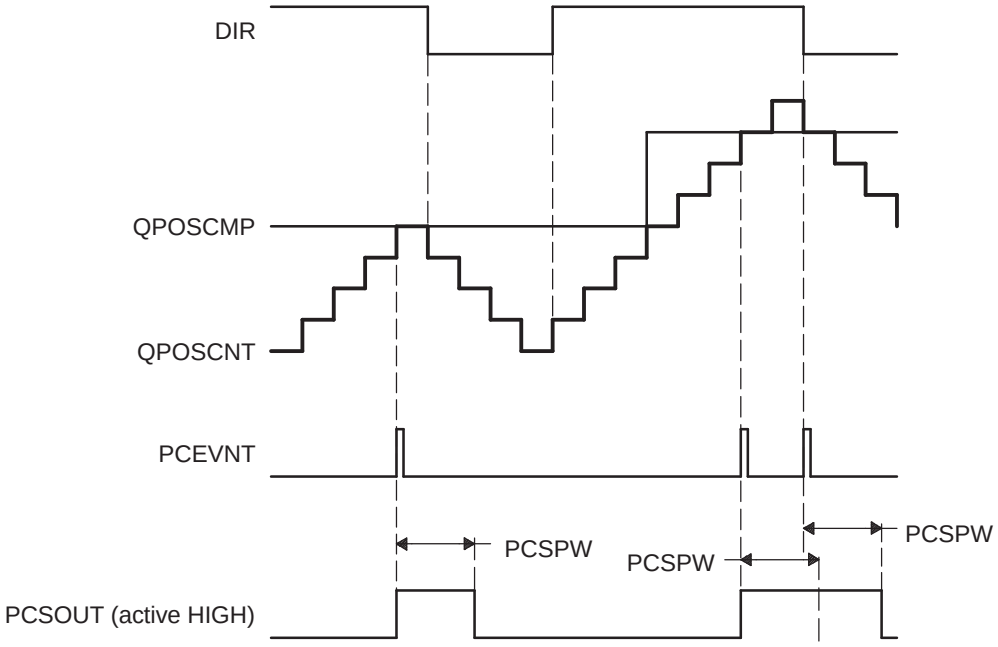
See the register section for the layout of the eQEP Position-Compare Control Register (QPOSCTL) and description of the QPOSCTL bit fields.

**Figure 22-13. 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 22-14.

**Figure 22-14. eQEP Position-compare Sync Output Pulse Stretcher**




## 22.6 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 22-15. This feature is typically used for low speed measurement using the following equation:

$$v(k) = \frac{X}{t(k) - t(k-1)} = \frac{X}{\Delta T} \quad (6)$$

where,

- X - Unit position is defined by integer multiple of quadrature edges (see Figure 22-16)
- ΔT - Elapsed time between unit position events
- v(k) - Velocity at time instant "k"

The eQEP capture timer (QCTMR) runs from prescaled SYSCLKOUT and the prescaler is programmed by the QCAPCTL[CCPS] bits. The capture timer (QCTMR) value is latched into the capture period register (QCPRD) on every unit position event and then the capture timer is reset, a flag is set in QEPSTS:UPEVNT to indicate that new value is latched into the QCPRD register. Software can check this status flag before reading the period register for low speed measurement and clear the flag by writing 1.

Time measurement (ΔT) between unit position events will be correct if the following conditions are met:

- No more than 65,535 counts have occurred between unit position events.
- No direction change between unit position events.

The capture unit sets the eQEP overflow error flag (QEPSTS[COEF]) in the event of capture timer overflow between unit position events. If a direction change occurs between the unit position events, then an error flag is set in the status register (QEPSTS[CDEF]).

Capture Timer (QCTMR) and Capture period register (QCPRD) can be configured to latch on following events.

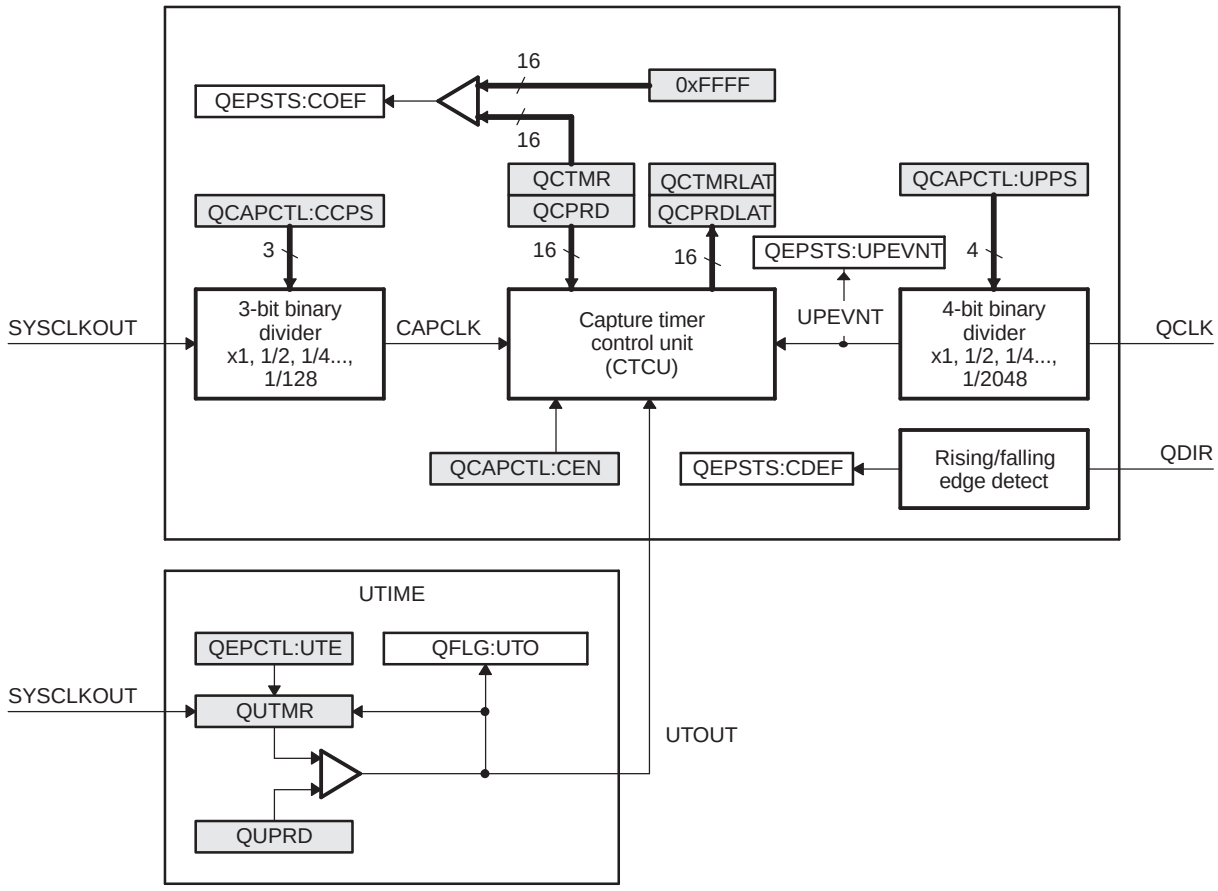
- CPU read of QPOSCNT register
- Unit time-out event

If the QEPCTL[QCLM] bit is cleared, then the capture timer and capture period values are latched into the QCTMRLAT and QCPRDLAT registers, respectively, when the CPU reads the position counter (QPOSCNT).

If the QEPCTL[QCLM] bit is set, then the position counter, capture timer, and capture period values are latched into the QPOSLAT, QCTMRLAT and QCPRDLAT registers, respectively, on unit time out.

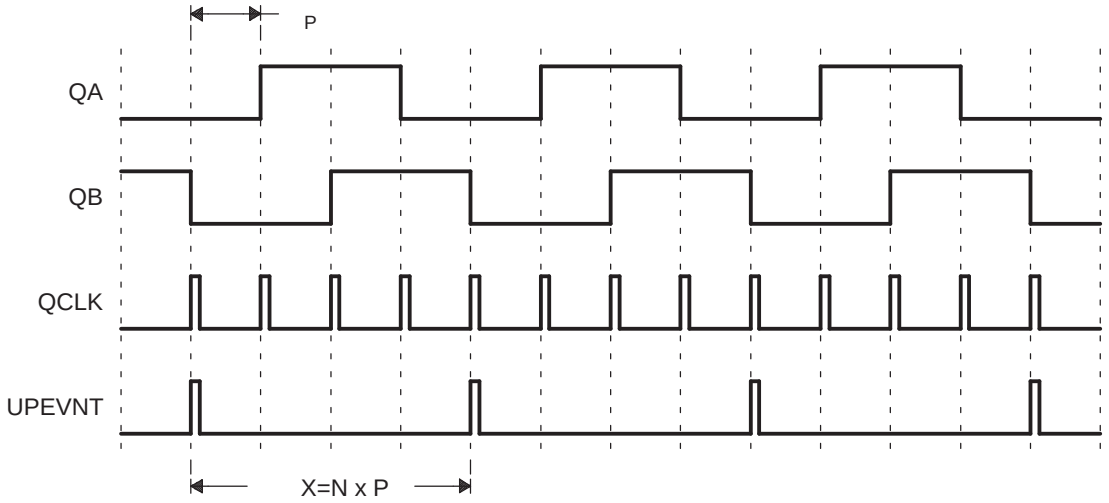
Figure 22-17 shows the capture unit operation along with the position counter.

**Figure 22-15. eQEP Edge Capture Unit**



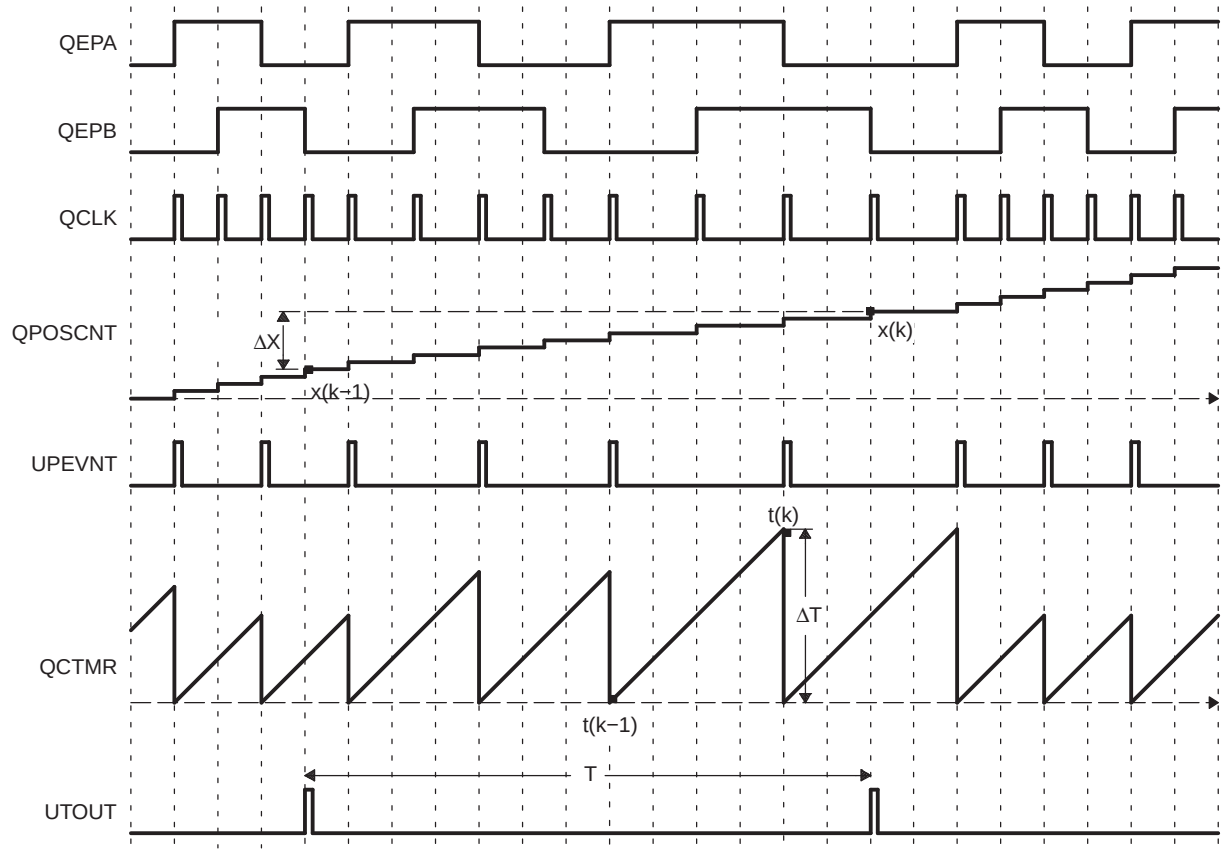

**NOTE:** The QCAPCTL[UPPS] prescaler should not be modified dynamically (such as switching the unit event prescaler from QCLK/4 to QCLK/8). Doing so may result in undefined behavior. The QCAPCTL[CPPS] prescaler can be modified dynamically (such as switching CAPCLK prescaling mode from SYSCLK/4 to SYSCLK/8) only after the capture unit is disabled.

**Figure 22-16. Unit Position Event for Low Speed Measurement (QCAPCTL[UPPS] = 0010)**




A N - Number of quadrature periods selected using QCAPCTL[UPPS] bits

**Figure 22-17. eQEP Edge Capture Unit - Timing Details**




Velocity Calculation Equations:

$$v(k) = \frac{x(k) - x(k-1)}{T} = \frac{\Delta X}{T} \quad (7)$$

where

v(k): Velocity at time instant k

x(k): Position at time instant k

x(k-1): Position at time instant k-1

T: Fixed unit time or inverse of velocity calculation rate

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"

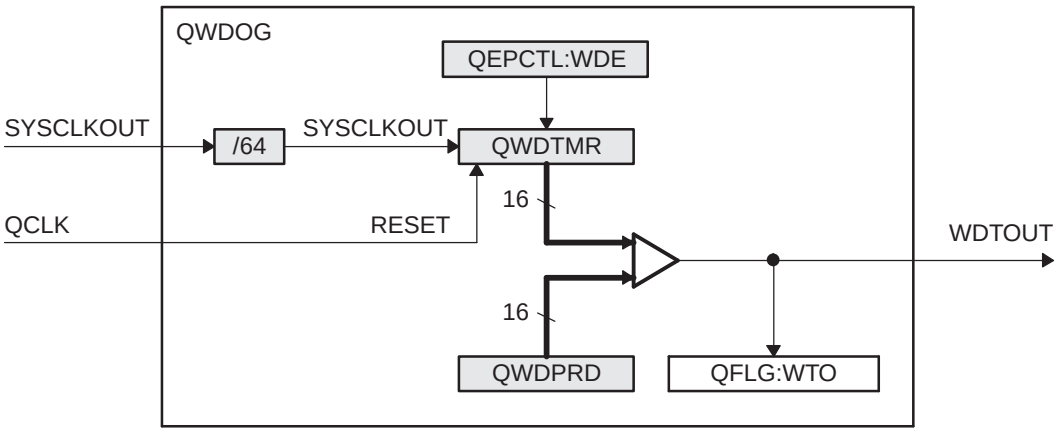
Unit time (T) and unit period(X) are configured using the QUPRD and QCAPCTL[UPPS] registers. Incremental position output and incremental time output is available in the QPOSLAT and QCPRDLAT registers.

| 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 ZCAPCTL[UPPS] bits |
| ΔT | Capture Period Latch (QCPRDLAT) |

## 22.7 eQEP Watchdog

The eQEP peripheral contains a 16-bit watchdog timer that monitors the quadrature-clock to indicate proper operation of the motion-control system. The eQEP watchdog timer is clocked from SYSCLKOUT/64 and the quadrate clock event (pulse) resets the watchdog timer. If no quadrature-clock event is detected until a period match (QWDPRD = QWDTMR), then the watchdog timer will time out and the watchdog interrupt flag will be set (QFLG[WTO]). The time-out value is programmable through the watchdog period register (QWDPRD).

**Figure 22-18. eQEP Watchdog Timer**

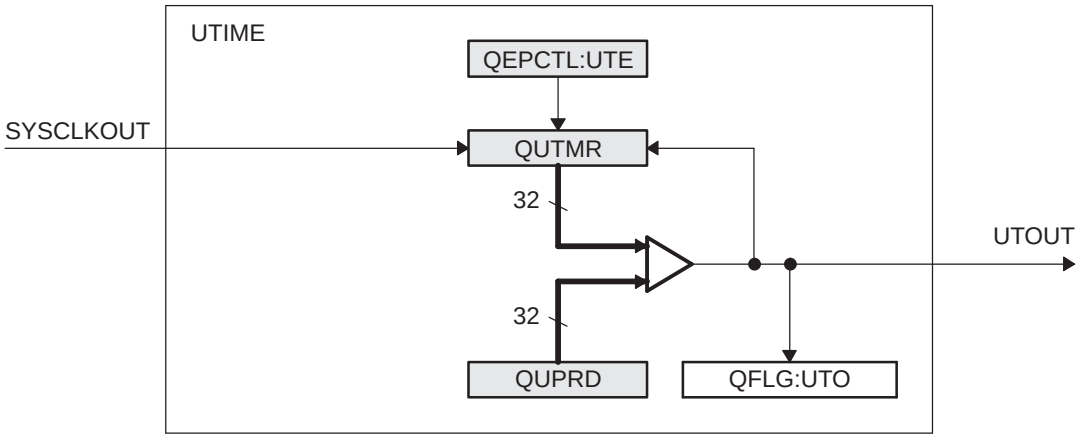



## 22.8 Unit Timer Base

The eQEP peripheral includes a 32-bit timer (QUTMR) that is clocked by SYSCLKOUT to generate periodic interrupts for velocity calculations. The unit time out interrupt is set (QFLG[UTO]) when the unit timer (QUTMR) matches the unit period register (QUPRD). The 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 22.6.

**Figure 22-19. eQEP Unit Time Base**




## 22.9 QMA Module

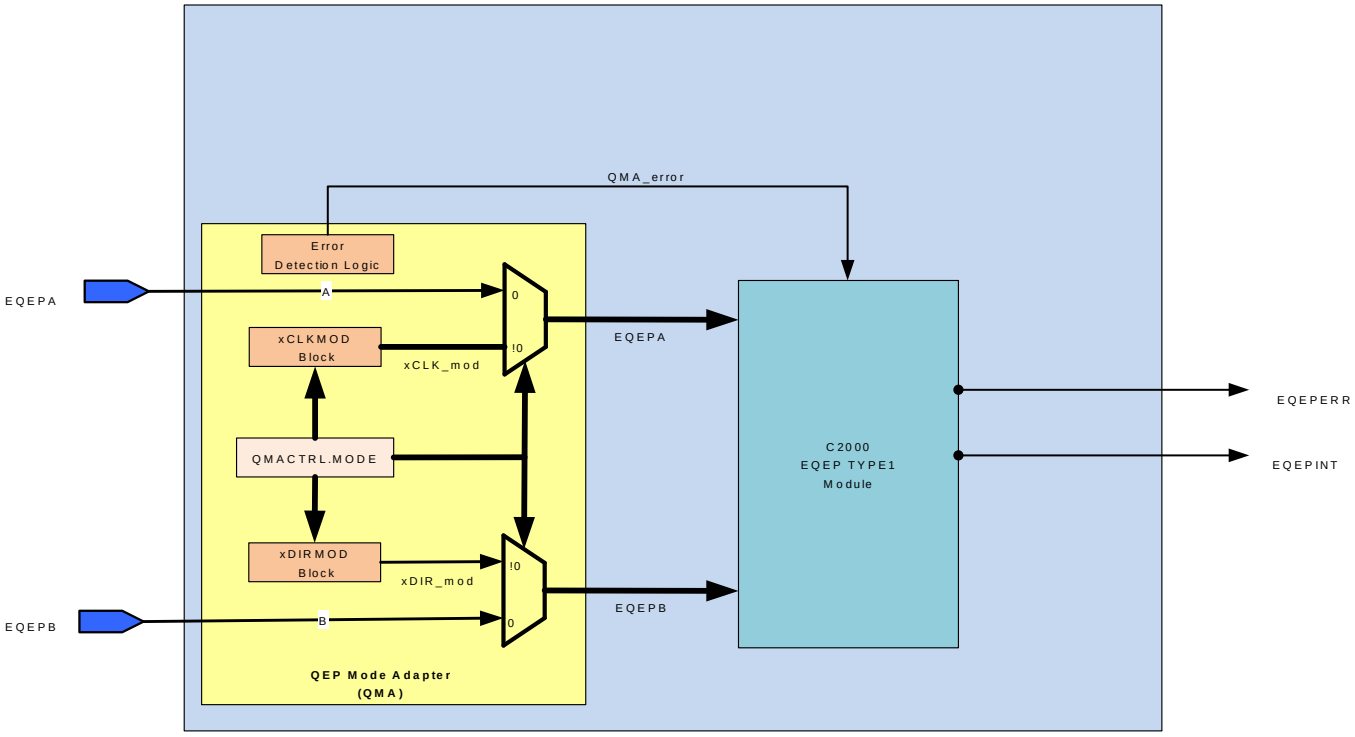
The QEP Mode Adapter (QMA) is designed to extend the C2000 eQEP module capabilities to support the additional modes described below. Figure 22-20 depicts how the QMA module is integrated into to the C2000 eQEP module.

At reset, by default QMA logic is bypassed and the EQEPA and EQEPB inputs from the pins go directly into the eQEP module. When QMA module is enabled by configuring the QMACTRL[MODE] register, the EQEPA and EQEPB input are processed by this module and modified version of EQEPA and EQEPB signals are sent to the eQEP module. The QMA module requires the eQEP module to be configured in the Direction-Count mode and generates a clock signal on EQEPA input and direction signal on EQEPB input as needed for the proper operation of the intended mode.

- The xCLKMOD block inside the QMA module looks at the transitions on external EQEPA and EQEPB signals to generate the clock signal on the EQEPA input to the eQEP module.
- The xDIRMOD block inside the QMA module looks at the transitions on external EQEPA and EQEPB signals to generate the direction signal on the EQEPB input to the eQEP module.

The QMA module has error detection logic to detect illegal transitions on EQEPA and EQEPB input signals. The QMA module's error and interrupt are integrated inside the eQEP module as described in the eQEP Interrupt Structure section. In addition QMACTRL register configuration can be locked using QMALOCK register. Refer to the register description for more details.

**Figure 22-20. QMA Module Block Diagram**




## 22.9.1 Modes of Operation

The QMA module can be operated in the following modes by configuring the QMACTRL register.

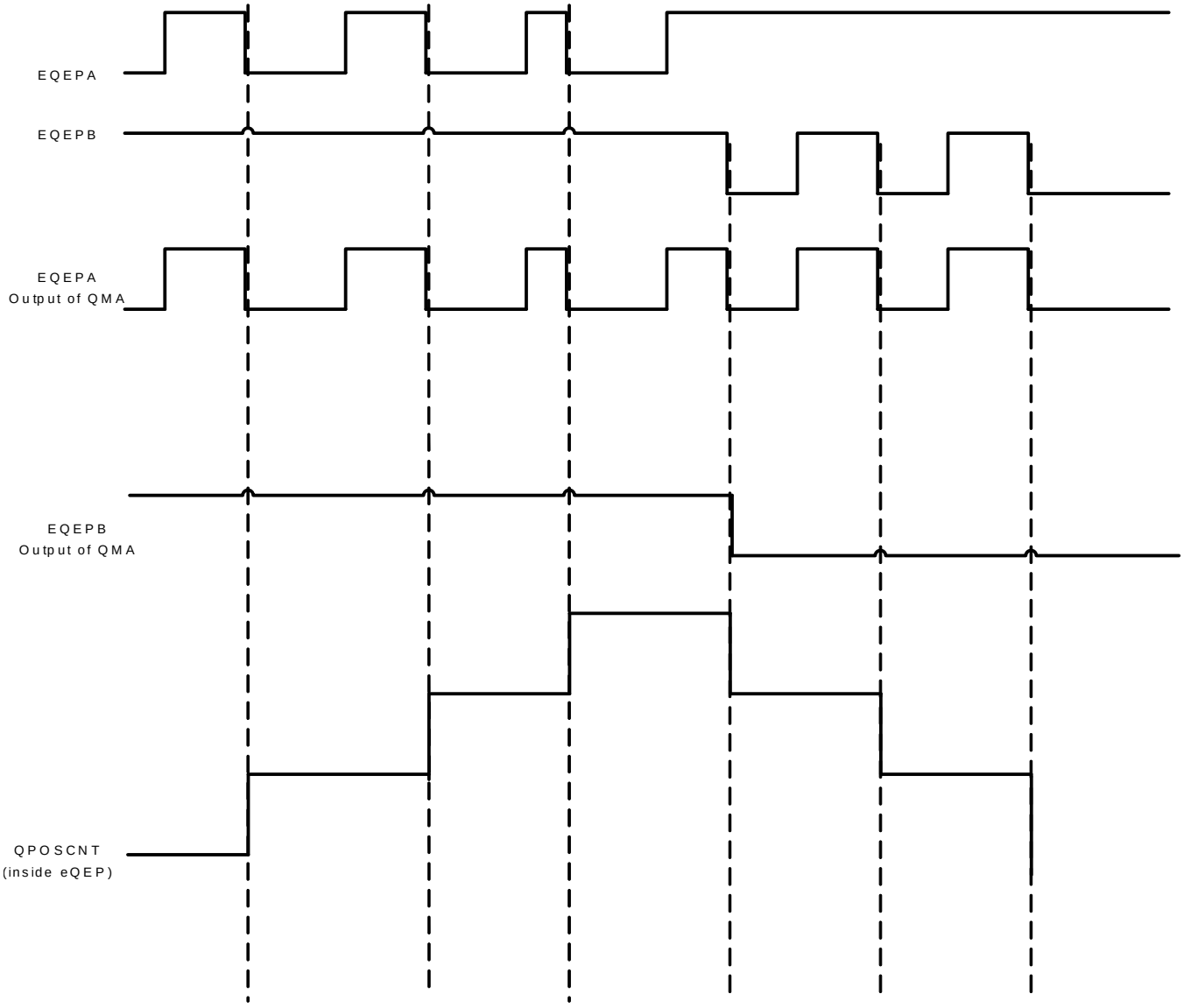
### 22.9.1.1 QMA Mode-1(QMACTRL[MODE]=1)

In this mode outputs of QMA correspond to the following as shown in Figure 22-21:

- EQEPA Output of QMA is AND of EQEPA and EQEPB inputs coming from the pin
- EQEPB Output of QMA is direction signal generated by QMA based on EQEPA and EQEPB inputs

This mode is used when the default state of EQEPA and EQEPB inputs is high

**Figure 22-21. QMA Mode-1**



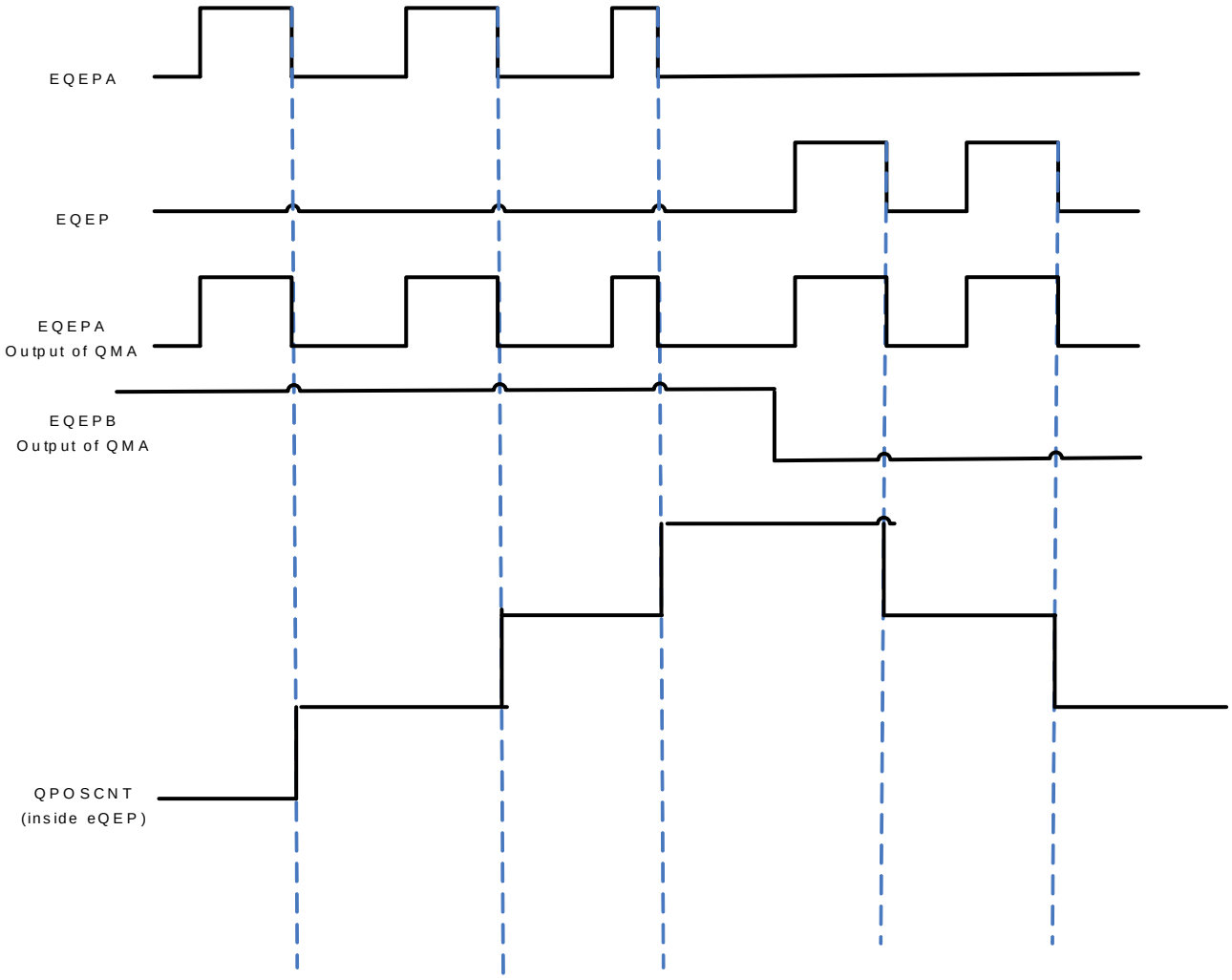

### 22.9.1.2 QMA Mode-2(QMACTRL[MODE]=2)

In this mode, outputs of QMA correspond to the following as shown in Figure 22-22:

- EQEPA Output of QMA is OR of EQEPA and EQEPB inputs coming from the pin
- EQEPB Output of QMA is direction signal generated by QMA based on EQEPA and EQEPB inputs

This mode is used when the default state of EQEPA and EQEPB inputs is low

**Figure 22-22. QMA Mode-2**



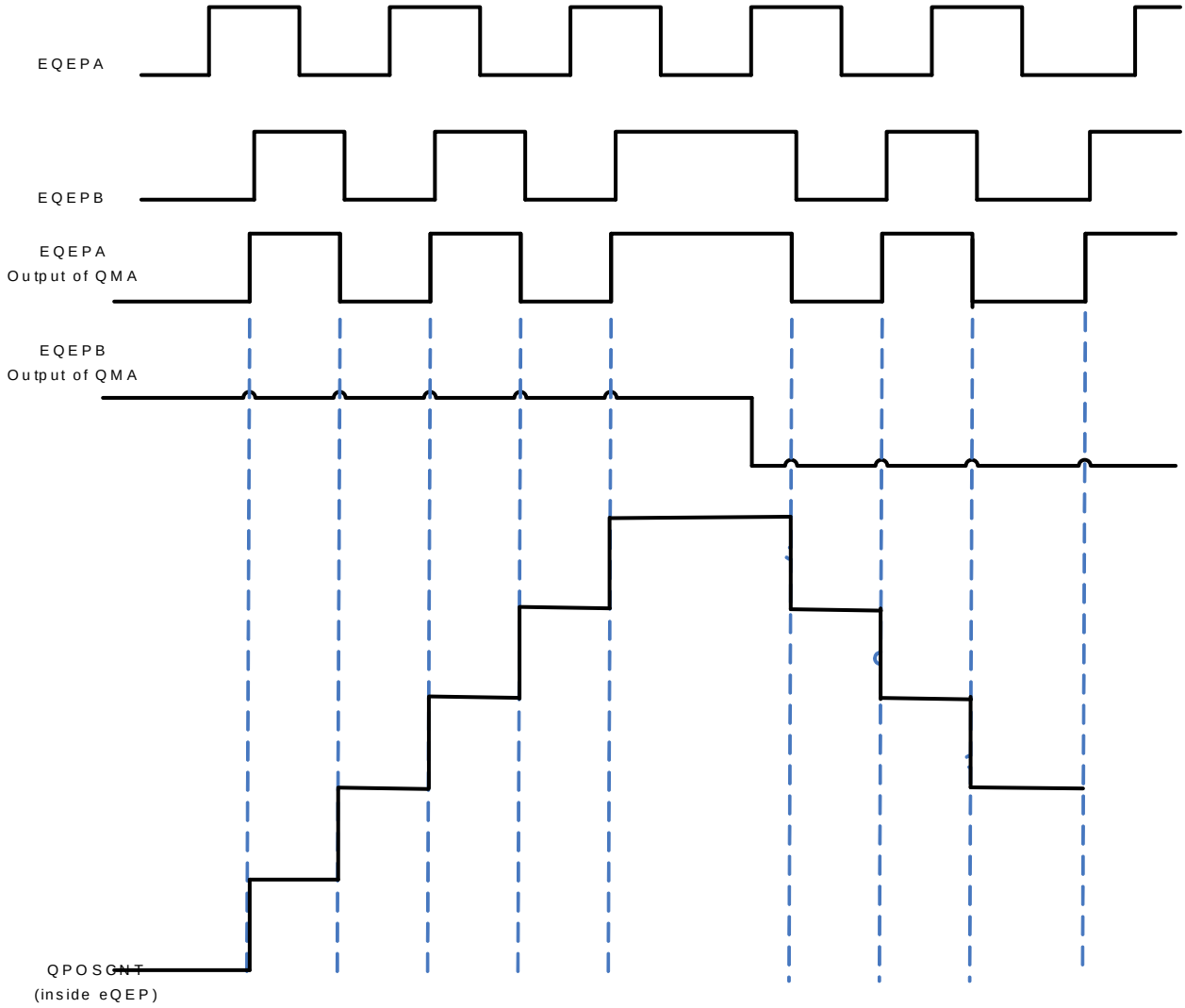

### 22.9.1.3 QMA Mode-3(QMACTRL[MODE]=3)

In this mode outputs of QMA correspond to the following as shown in Figure 22-23

- EQEPA Output of QMA is the same as EQEPB input signal coming from the pin
- EQEPB Output of QMA is direction signal generated by QMA based on EQEPA and EQEPB inputs

This mode is used when the default state of EQEPA and EQEPB inputs are in quadrature but EQEPB input alone is used for pulse counting.

**Figure 22-23. QMA Mode-3**



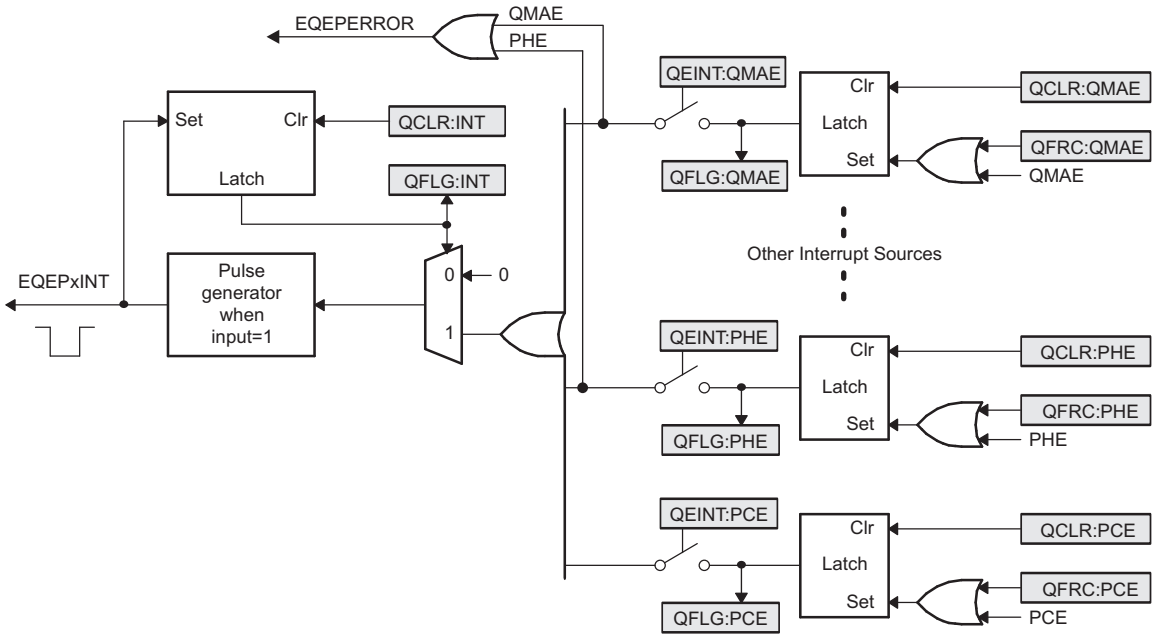

### 22.9.2 Interrupt and Error Generation

The error detection logic detects illegal transitions on EQEPA and EQEPB signals and generates an error signal. This error signal can be used to generate eQEP interrupt and error output. Refer to Section 22.10 for details.

## 22.10 eQEP Interrupt Structure

Figure 22-24 shows how the interrupt mechanism works in the EQEP module.

**Figure 22-24. EQEP Interrupt Generation**




Eleven interrupt events (PCE, PHE, QDC, WTO, PCU, PCO, PCR, PCM, SEL, IEL and UTO) can be generated. The interrupt control register (QEINT) is used to enable/disable individual interrupt event sources. The interrupt flag register (QFLG) indicates if any interrupt event has been latched and contains the global interrupt flag bit (INT).

An Interrupt pulse is generated to PIE when:

a. Interrupt is enabled for eQEP event inside QEINT register

b. Interrupt flag for eQEP event inside QFLG register is set, and

c. Global interrupt status flag bit QFLG[INT] had been cleared for previously generated interrupt event. The interrupt service routine will need to clear the global interrupt flag bit and the serviced event, via the interrupt clear register (QCLR), before any other interrupt pulses are generated.If either flags inside the QFLG register are not cleared, further interrupt event will not generate interrupt to PIE. You can force an interrupt event by way of the interrupt force register (QFRC), which is useful for test purposes.

## 22.11 Registers

### *22.11.1 Enhanced QEP Base Addresses*

**Table 22-3. 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 |

### 22.11.1.1 EQEP_REGS Registers

Table 22-4 lists the memory-mapped registers for the EQEP_REGS. All register offset addresses not listed in Table 22-4 should be considered as reserved locations and the register contents should not be modified.

**Table 22-4. 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 |
| 30h | REV | QEP Revision Number | | Go |
| 32h | QEPSTROBESEL | QEP Strobe select register | | Go |
| 34h | QMACTRL | QMA Control register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 22-5 shows the codes that are used for access types in this section.

**Table 22-5. EQEP_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| W=1 | W | Write |
| **Reset or Default Value** | | |
| *-n* | | Value after reset or the default value |
| **Register Array Variables** | | |

**Table 22-5. EQEP_REGS Access Type Codes (continued)**

| Access Type | Code | Description |
|---|---|---|
| i,j,k,l,m,n | | When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula. |
| y | | When this variable is used in a register name, an offset, or an address it refers to the value of a register array. |

#### 22.11.1.1.1 QPOSCNT Register (Offset = 0h) [reset = 0h]

QPOSCNT is shown in Figure 22-25 and described in Table 22-6.

Return to Summary Table.

Position Counter

**Figure 22-25. QPOSCNT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QPOSCNT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 22-6. 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 given reference point. This Register acts as a Read ONLY register while counter is counting up/down.<br>Reset type: SYSRSn |

#### 22.11.1.1.2 QPOSINIT Register (Offset = 2h) [reset = 0h]

QPOSINIT is shown in Figure 22-26 and described in Table 22-7.

Return to Summary Table.

Position Counter Init

**Figure 22-26. QPOSINIT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QPOSINIT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 22-7. 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 |

#### 22.11.1.1.3 QPOSMAX Register (Offset = 4h) [reset = 0h]

QPOSMAX is shown in Figure 22-27 and described in Table 22-8.

Return to Summary Table.

Maximum Position Count

**Figure 22-27. QPOSMAX Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QPOSMAX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 22-8. 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 |

#### 22.11.1.1.4 QPOSCMP Register (Offset = 6h) [reset = 0h]

QPOSCMP is shown in Figure 22-28 and described in Table 22-9.

Return to Summary Table.

Position Compare

**Figure 22-28. QPOSCMP Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QPOSCMP | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 22-9. 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.<br>Reset type: SYSRSn |

#### 22.11.1.1.5 QPOSILAT Register (Offset = 8h) [reset = 0h]

QPOSILAT is shown in Figure 22-29 and described in Table 22-10.

Return to Summary Table.

Index Position Latch

**Figure 22-29. QPOSILAT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QPOSILAT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 22-10. 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 |

#### 22.11.1.1.6 QPOSSLAT Register (Offset = Ah) [reset = 0h]

QPOSSLAT is shown in Figure 22-30 and described in Table 22-11.

Return to Summary Table.

Strobe Position Latch

**Figure 22-30. 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 22-11. 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 |

#### 22.11.1.1.7 QPOSLAT Register (Offset = Ch) [reset = 0h]

QPOSLAT is shown in Figure 22-31 and described in Table 22-12.

Return to Summary Table.

Position Latch

**Figure 22-31. QPOSLAT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QPOSLAT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 22-12. 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 |

#### 22.11.1.1.8 QUTMR Register (Offset = Eh) [reset = 0h]

QUTMR is shown in Figure 22-32 and described in Table 22-13.

Return to Summary Table.

QEP Unit Timer

**Figure 22-32. QUTMR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QUTMR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 22-13. 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.<br>Reset type: SYSRSn |

#### 22.11.1.1.9 QUPRD Register (Offset = 10h) [reset = 0h]

QUPRD is shown in Figure 22-33 and described in Table 22-14.

Return to Summary Table.

QEP Unit Period

**Figure 22-33. QUPRD Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QUPRD | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 22-14. 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 |

#### 22.11.1.1.10 QWDTMR Register (Offset = 12h) [reset = 0h]

QWDTMR is shown in Figure 22-34 and described in Table 22-15.

Return to Summary Table.

QEP Watchdog Timer

**Figure 22-34. QWDTMR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QWDTMR | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 22-15. 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 |

#### 22.11.1.1.11 QWDPRD Register (Offset = 13h) [reset = 0h]

QWDPRD is shown in Figure 22-35 and described in Table 22-16.

Return to Summary Table.

QEP Watchdog Period

**Figure 22-35. QWDPRD Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QWDPRD | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 22-16. 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 |

#### 22.11.1.1.12 QDECCTL Register (Offset = 14h) [reset = 0h]

QDECCTL is shown in Figure 22-36 and described in Table 22-17.

Return to Summary Table.

Quadrature Decoder Control

**Figure 22-36. QDECCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| QSRC | | SOEN | SPSEL | XCR | SWAP | IGATE | QAP |
| R/W-0h | | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| QBP | QIP | QSP | RESERVED | | | | |
| R/W-0h | R/W-0h | R/W-0h | R-0h | | | | |

**Table 22-17. 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 |

**Table 22-17. QDECCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 22.11.1.1.13 QEPCTL Register (Offset = 15h) [reset = 0h]

QEPCTL is shown in Figure 22-37 and described in Table 22-18.

Return to Summary Table.

QEP Control

**Figure 22-37. QEPCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| FREE_SOFT | | PCRM | | SEI | | IEI | |
| R/W-0h | | R/W-0h | | R/W-0h | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SWI | SEL | IEL | | QPEN | QCLM | UTE | WDE |
| R/W-0h | R/W-0h | R/W-0h | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 22-18. 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>QWDTMR behavior<br>Watchdog counter stops immediately<br>QUTMR behavior<br>Unit timer stops immediately<br>QCTMR behavior<br>Capture Timer stops immediately<br>1h (R/W) = QPOSCNT behavior<br>Position counter continues to count until the rollover<br>QWDTMR behavior<br>Watchdog counter counts until WD period match roll over<br>QUTMR behavior<br>Unit timer counts until period rollover<br>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>QWDTMR behavior<br>Watchdog counter is unaffected by emulation suspend<br>QUTMR behavior<br>Unit timer is unaffected by emulation suspend<br>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 |

**Table 22-18. QEPCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11-10 | SEI | R/W | 0h | Strobe event init<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 |
| 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 QFLAG register do not show value as 0 and show the actual state of that flag.<br>1h (R/W) = eQEP position counter is enabled |

**Table 22-18. QEPCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |

#### 22.11.1.1.14 QCAPCTL Register (Offset = 16h) [reset = 0h]

QCAPCTL is shown in Figure 22-38 and described in Table 22-19.

Return to Summary Table.

Qaudrature Capture Control

**Figure 22-38. QCAPCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| CEN | RESERVED | | | | | | |
| R/W-0h | R-0h | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | CCPS | | | UPPS | | | |
| R-0h | R/W-0h | | | R/W-0h | | | |

**Table 22-19. 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 |

#### 22.11.1.1.15 QPOSCTL Register (Offset = 17h) [reset = 0h]

QPOSCTL is shown in Figure 22-39 and described in Table 22-20.

Return to Summary Table.

Position Compare Control

**Figure 22-39. QPOSCTL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| PCSHDW | PCLOAD | PCPOL | PCE | PCSPW | | | |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PCSPW | | | | | | | |
| R/W-0h | | | | | | | |

**Table 22-20. 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 |

#### 22.11.1.1.16 QEINT Register (Offset = 18h) [reset = 0h]

QEINT is shown in Figure 22-40 and described in Table 22-21.

Return to Summary Table.

QEP Interrupt Control

**Figure 22-40. QEINT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | QMAE | UTO | IEL | SEL | PCM |
| R-0h | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PCR | PCO | PCU | WTO | QDC | QPE | PCE | RESERVED |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R-0h |

**Table 22-21. QEINT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12 | QMAE | R/W | 0h | QMA Error Interrupt enable<br>Reset type: SYSRSn<br>0h (R/W) = Interrupt is disabled<br>1h (R/W) = Interrupt is enabled |
| 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 |

**Table 22-21. QEINT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 22.11.1.1.17 QFLG Register (Offset = 19h) [reset = 0h]

QFLG is shown in Figure 22-41 and described in Table 22-22.

Return to Summary Table.

QEP Interrupt Flag

**Figure 22-41. QFLG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | QMAE | UTO | IEL | SEL | PCM |
| R-0h | | | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PCR | PCO | PCU | WTO | QDC | PHE | PCE | INT |
| R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 22-22. QFLG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12 | QMAE | R | 0h | QMA Error interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = Interrupt was generated |
| 11 | UTO | R | 0h | Unit time out interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = Set by eQEP unit timer period match |
| 10 | IEL | R | 0h | Index event latch interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = This bit is set after latching the QPOSCNT to QPOSILAT |
| 9 | SEL | R | 0h | Strobe event latch interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = This bit is set after latching the QPOSCNT to QPOSSLAT |
| 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) = This bit is set on position-compare match |
| 7 | PCR | R | 0h | Position-compare ready interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = This bit is set after transferring the shadow register value to the active position compare register |
| 6 | PCO | R | 0h | Position counter overflow interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = This bit is set on position counter overflow. |
| 5 | PCU | R | 0h | Position counter underflow interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = This bit is set on position counter underflow. |

**Table 22-22. QFLG Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | WTO | R | 0h | Watchdog timeout interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = Set by watchdog timeout |
| 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 |
| 2 | PHE | R | 0h | Quadrature phase error interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = Set on simultaneous transition of QEPA and QEPB |
| 1 | PCE | R | 0h | Position counter error interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No interrupt generated<br>1h (R/W) = Position counter error |
| 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 |

#### 22.11.1.1.18 QCLR Register (Offset = 1Ah) [reset = 0h]

QCLR is shown in Figure 22-42 and described in Table 22-23.

Return to Summary Table.

QEP Interrupt Clear

**Figure 22-42. QCLR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | QMAE | UTO | IEL | SEL | PCM |
| R-0h | | | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PCR | PCO | PCU | WTO | QDC | PHE | PCE | INT |
| R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h | R=0/W=1-0h |

**Table 22-23. QCLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12 | QMAE | R=0/W=1 | 0h | Clear QMA Error interrupt flag<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Clears the interrupt flag |
| 11 | UTO | R=0/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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/W=1 | 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 |

**Table 22-23. QCLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | WTO | R=0/W=1 | 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/W=1 | 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 |
| 2 | PHE | R=0/W=1 | 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/W=1 | 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/W=1 | 0h | Global interrupt clear flag<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Clears the interrupt flag |

### 22.11.1.1.19 QFRC Register (Offset = 1Bh) [reset = 0h]

QFRC is shown in Figure 22-43 and described in Table 22-24.

Return to Summary Table.

QEP Interrupt Force

**Figure 22-43. QFRC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | QMAE | UTO | IEL | SEL | PCM |
| R-0h | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PCR | PCO | PCU | WTO | QDC | PHE | PCE | RESERVED |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R-0h |

**Table 22-24. QFRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12 | QMAE | R/W | 0h | Force QMA error interrupt<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Force the interrupt |
| 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 |

**Table 22-24. QFRC Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

#### 22.11.1.1.20 QEPSTS Register (Offset = 1Ch) [reset = 80h]

QEPSTS is shown in Figure 22-44 and described in Table 22-25.

Return to Summary Table.

QEP Status

**Figure 22-44. QEPSTS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| UPEVNT | FIDF | QDF | QDLF | COEF | CDEF | FIMF | PCEF |
| R/W=1-1h | R-0h | R-0h | R-0h | R/W=1-0h | R/W=1-0h | R/W=1-0h | R-0h |

**Table 22-25. QEPSTS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7 | UPEVNT | R/W=1 | 1h | 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=1 | 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=1 | 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=1 | 0h | First index marker flag<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'. |

**Table 22-25. 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 |

#### 22.11.1.1.21 QCTMR Register (Offset = 1Dh) [reset = 0h]

QCTMR is shown in Figure 22-45 and described in Table 22-26.

Return to Summary Table.

QEP Capture Timer

**Figure 22-45. QCTMR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QCTMR | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 22-26. 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 |

#### 22.11.1.1.22 QCPRD Register (Offset = 1Eh) [reset = 0h]

QCPRD is shown in Figure 22-46 and described in Table 22-27.

Return to Summary Table.

QEP Capture Period

**Figure 22-46. QCPRD Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QCPRD | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 22-27. 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 |

#### 22.11.1.1.23 QCTMRLAT Register (Offset = 1Fh) [reset = 0h]

QCTMRLAT is shown in Figure 22-47 and described in Table 22-28.

Return to Summary Table.

QEP Capture Latch

**Figure 22-47. QCTMRLAT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QCTMRLAT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 22-28. 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 |

#### 22.11.1.1.24 QCPRDLAT Register (Offset = 20h) [reset = 0h]

QCPRDLAT is shown in Figure 22-48 and described in Table 22-29.

Return to Summary Table.

QEP Capture Period Latch

**Figure 22-48. QCPRDLAT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QCPRDLAT | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 22-29. 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 |

#### 22.11.1.1.25 REV Register (Offset = 30h) [reset = 1h]

REV is shown in Figure 22-49 and described in Table 22-30.

Return to Summary Table.

QEP Revision Number

**Figure 22-49. REV Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | |
| R=0-0h | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | | | | MINOR | | | MAJOR | | |
| R=0-0h | | | | | | | | | | R-0h | | | R-1h | | |

**Table 22-30. REV Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-6 | RESERVED | R=0 | 0h | Reserved |
| 5-3 | MINOR | R | 0h | This field specifies the Minor Revision number for the eQEP IP.<br>Reset type: N/A |
| 2-0 | MAJOR | R | 1h | This field specifies the Major Revision number for the eQEP IP.<br>Reset type: N/A |

#### 22.11.1.1.26 QEPSTROBESEL Register (Offset = 32h) [reset = 0h]

QEPSTROBESEL is shown in Figure 22-50 and described in Table 22-31.

Return to Summary Table.

QEP Strobe select register

**Figure 22-50. QEPSTROBESEL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R=0-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | STROBESEL | |
| R=0-0h | | | | | | R/W-0h | |

**Table 22-31. QEPSTROBESEL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-2 | RESERVED | R=0 | 0h | Reserved |
| 1-0 | STROBESEL | R/W | 0h | Strobe source select:<br>Reset type: SYSRSn<br>0h (R/W) = QEP Strobe after polarity mux<br>1h (R/W) = QEP Strobe after polarity mux<br>2h (R/W) = QEP Strobe after polarity mux ORed with ADCSOCA<br>3h (R/W) = QEP Strobe after polarity mux ORed with ADCSOCB |

#### 22.11.1.1.27 QMACTRL Register (Offset = 34h) [reset = 0h]

QMACTRL is shown in Figure 22-51 and described in Table 22-32.

Return to Summary Table.

QMA Control register

**Figure 22-51. QMACTRL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | |
| R=0-0h | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | | | | | | | MODE | | |
| R=0-0h | | | | | | | | | | | | | R/W-0h | | |

**Table 22-32. QMACTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-3 | RESERVED | R=0 | 0h | Reserved |
| 2-0 | MODE | R/W | 0h | Select Mode for QMA mode:<br>000 : QMA Module is bypassed.<br>001 : QMA Mode-1 operation selected<br>010 : QMA Mode-2 operation selected<br>011 : QMA Module is bypassed (reserved)<br>1xx : QMA Module is bypassed (reserved)<br>Reset type: SYSRSn |

# Serial Peripheral Interface (SPI)

This chapter describes the serial peripheral interface (SPI) which is a high-speed synchronous serial input and output (I/O) port that allows a serial bit stream of programmed length (one to 16 bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI is normally used for communications between the MCU controller and external peripherals or another controller. Typical applications include external I/O or peripheral expansion via devices such as shift registers, display drivers, and analog-to-digital converters (ADCs). Multi-device communications are supported by the master or slave operation of the SPI. The port supports a 16-level, receive and transmit FIFO for reducing CPU servicing overhead.

**Topic** **Page**

## 23.1 SPI Module Overview

### 23.1.1 Features

The SPI module features include:

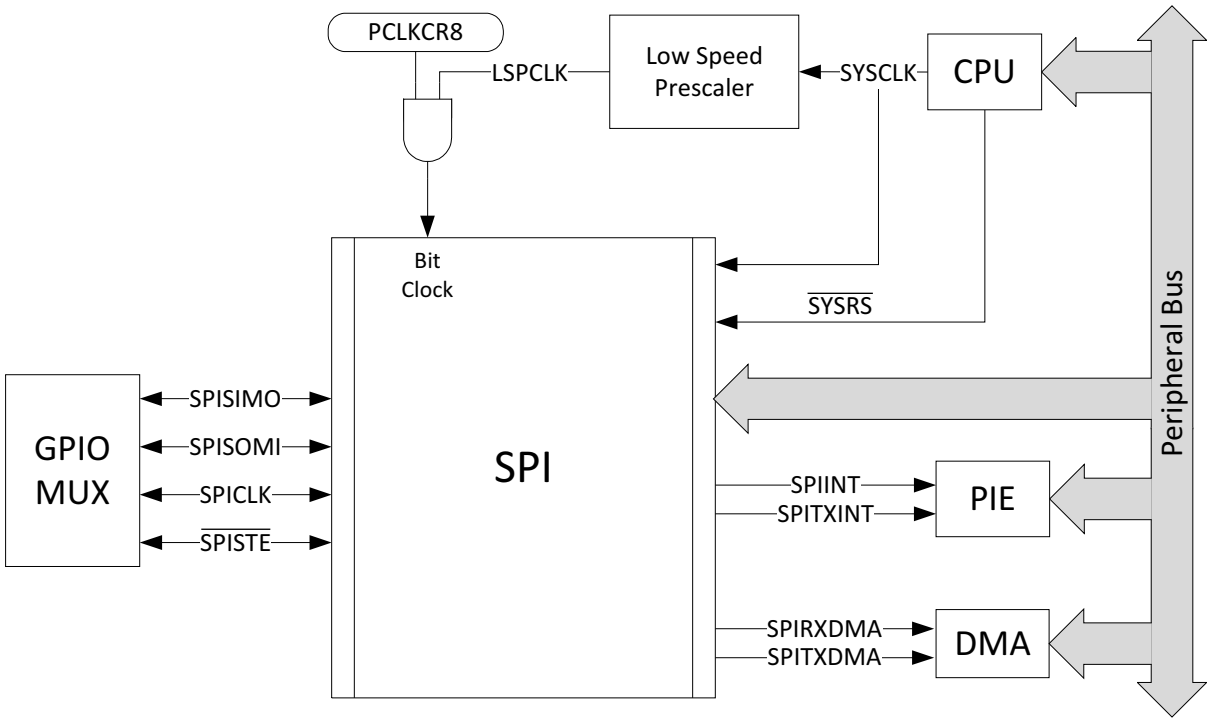
- SPISOMI: SPI slave-output/master-input pin
- SPISIMO: SPI slave-input/master-output pin
- $\overline{\text{SPISTE}}$: SPI slave transmit-enable pin
- SPICLK: SPI serial-clock pin

**NOTE:** All four pins can be used as GPIO, if the SPI module is not used.

- Two operational modes: Master and Slave
- Baud rate: 125 different programmable rates. The maximum baud rate that can be employed is limited by the maximum speed of the I/O buffers used on the SPI pins. See the device-specific data manual for more details.
- Data word length: one to sixteen data bits
- Four clocking schemes (controlled by clock polarity and clock phase bits) include:
  - Falling edge without phase delay: SPICLK active-high. SPI transmits data on the falling edge of the SPICLK signal and receives data on the rising edge of the SPICLK signal.
  - Falling edge with phase delay: SPICLK active-high. SPI transmits data one half-cycle ahead of the falling edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
  - Rising edge without phase delay: SPICLK inactive-low. SPI transmits data on the rising edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
  - Rising edge with phase delay: SPICLK inactive-low. SPI transmits data one half-cycle ahead of the falling edge of the SPICLK signal and receives data on the rising edge of the SPICLK signal.
- Simultaneous receive and transmit operation (transmit function can be disabled in software)
- Transmitter and receiver operations are accomplished through either interrupt- driven or polled algorithm
- 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

Figure 23-1 shows the SPI CPU interfaces.

**Figure 23-1. SPI CPU Interface**




## 23.2 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.

### *23.2.1 SPI Module Signals*

Table 23-1 classifies and provides a summary of the SPI module signals.

**Table 23-1. 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. An active $\overline{\text{SPISTE}}$ will not allow the slave to receive data. This prevents the SPI slave from losing synchronization with the master. It is this reason that TI does not recommend that the $\overline{\text{SPISTE}}$ always be tied to the active state.

If the SPI slave does ever lose synchronization with the master, toggling SPISWRESET will reset internal bit counter as well as the various status flags in the module. By resetting the bit counter, the SPI will interpret the next clock transition as the first bit of a new transmission. The register bit singular fields which are reset by SPISWRESET can be found in Section 23.5

**Configuring a GPIO to emulate $\overline{\text{SPISTE}}$**

In many systems, a SPI master may be connected to multiple SPI slaves using multiple instances of $\overline{\text{SPISTE}}$. Though this SPI module does not natively support multiple $\overline{\text{SPISTE}}$ signals, it is possible to emulate this behavior in software using GPIOs. In this configuration, the SPI must be configured as the master. Rather than using the GPIO Mux to select $\overline{\text{SPISTE}}$, the application would configure pins to be GPIO outputs, one GPIO per SPI slave. Before transmitting any data, the application would drive the desired GPIO to the active state. Immediately after the transmission has been completed, the GPIO chip select would be driven to the inactive state. This process can be repeated for many slaves which share the SPICLK, SPISIMO, and SPISOMI lines.

### *23.2.2 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 should be set to asynchronous mode by setting the appropriate GPxQSELn register bits to 11b. The internal pullups can be configured in the GPyPUD register.

See the *GPIO* chapter for more details on GPIO mux and settings.

#### 23.2.2.1 GPIOs Required for High-Speed Mode

The high-speed mode of the SPI is available on all GPIO mux options. To enable the high-speed enhancements, set SPICCR.HS_MODE to 1. Ensure that the capacitive loading on the pin does not exceed the value stated in the device Data Manual.

When not operating in high-speed mode, or if the capacitive loading on the pins exceed the value stated in the device Data Manual, SPICCR.HS_MODE should be set to 0.

### *23.2.3 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 without 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 will generate 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 will generate an interrupt on the SPIINT vector if the OVERRUNINTENA bit is set and OVERRUN_FLAG was previously cleared.

In FIFO mode, the SPI can interrupt the CPU upon a match condition between the current receive FIFO status (RXFFST) and the receive FIFO interrupt level (RXFFIL). If RXFFST is greater than or equal to RXFFIL, the receive FIFO interrupt flag (RXFFINT) will be set. SPIRXINT will be triggered in the PIE block if RXFFINT is set and the receive FIFO interrupt is enabled (RXFFIENA = 1).

**SPITXINT**

The SPITXINT interrupt is not available when the SPI is operating in non-FIFO mode.

In FIFO mode, the SPITXINT behavior is similar to the SPIRXINT. SPITXINT is generated upon a match condition between the current transmit FIFO status (TXFFST) and the transmit FIFO interrupt level (TXFFIL). If TXFFST is less than or equal to TXFFIL, the transmit FIFO interrupt flag (TXFFINT) will be set. SPITXINT will be triggered in the PIE block if TXFFINT is set and the transmit FIFO interrupt is enabled in the SPI module (TXFFIENA = 1).

Figure 23-2 and Table 23-2 show how these control bits influence the SPI interrupt generation.

**Figure 23-2. SPI Interrupt Flags and Enable Logic Generation**

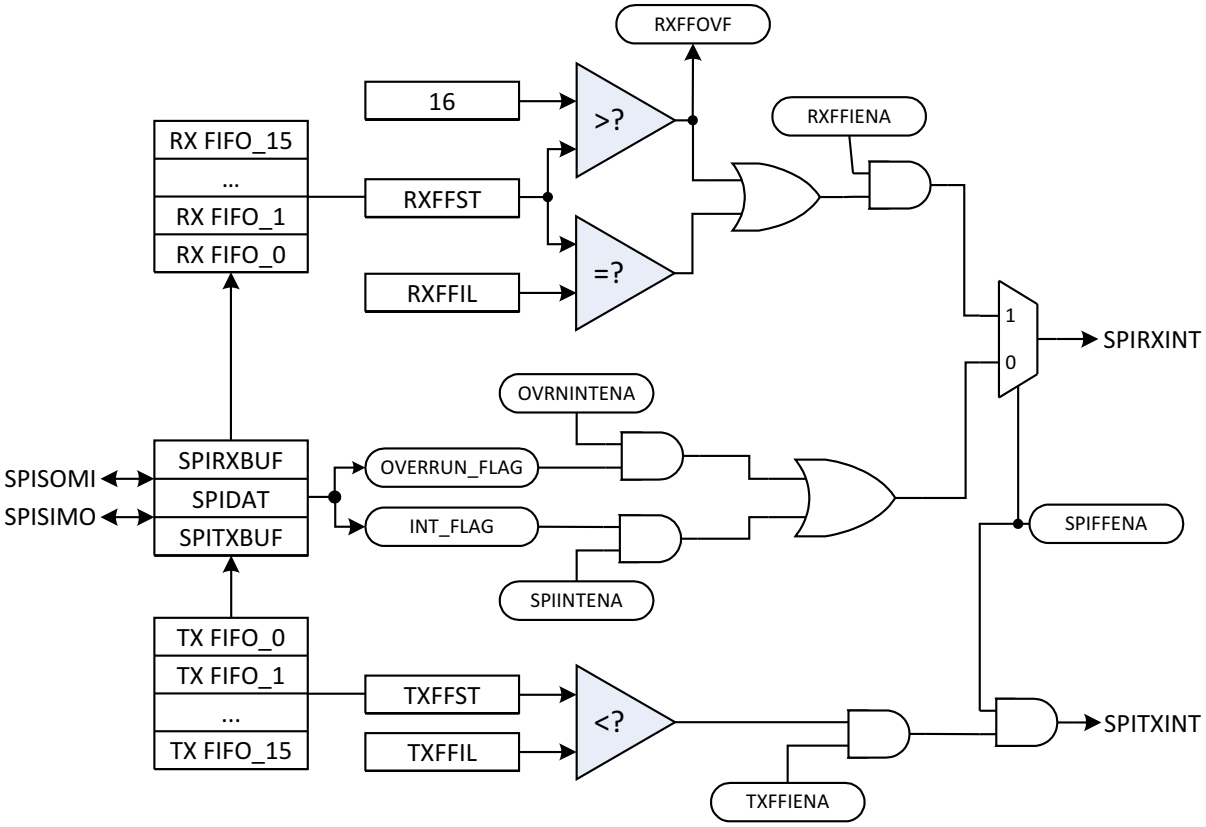


**Table 23-2. SPI Interrupt Flag Modes**

| FIFO Options | SPI interrupt Source | Interrupt Flags | Interrupt Enables | FIFO Enable (SPIFFENA) | Interrupt[1] Line |
|---|---|---|---|---|---|
| **SPI without FIFO** | | | | | |
| | Receive overrun | RXOVRN | OVRNINTENA | 0 | SPIRXINT |
| | Data receive | SPIINT | SPIINTENA | 0 | SPIRXINT |
| | Transmit empty | SPIINT | SPIINTENA | 0 | SPIRXINT |
| **SPI FIFO mode** | | | | | |
| | FIFO receive | RXFFIL | RXFFIENA | 1 | SPIRXINT |
| | Transmit empty | TXFFIL | TXFFIENA | 1 | SPITXINT |

(1) In non-FIFO mode, SPIRXINT is the same name as the SPIINT interrupt in 28x devices.

### 23.2.4 DMA Support

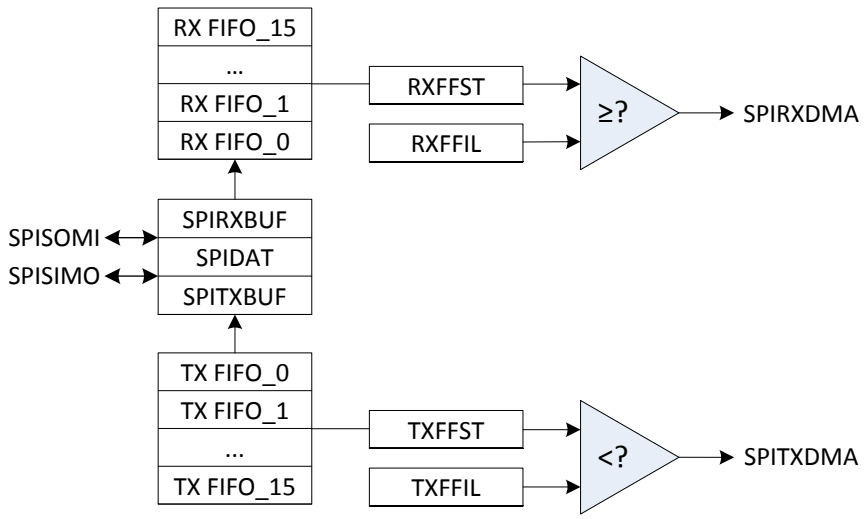
Both the CPU and DMA have access to the SPI data registers via the internal peripheral bus. This access is limited to 16-bit register read/writes. Each SPI module can generate two DMA events, SPITXDMA and SPIRXDMA. The DMA events are controlled by configuring the SPIFFTX.TXFFIL and SPIFFRX.RXFFIL appropriately. SPITXDMA activates when TXFFST is less than the interrupt level (TXFFIL). SPIRXDMA activates when RXFFST is greater than or equal to the interrupt level (RXFFIL).

The SPI must have FIFO enhancements enabled in order for the DMA triggers to be generated.

For more information on configuring the SPI for DMA transfers refer to Section 23.3.8.

Figure 23-3 is a block diagram showing the DMA trigger generation from the SPI module.

**Figure 23-3. SPI DMA Trigger Diagram**




## 23.3 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.

### 23.3.1 Introduction to Operation

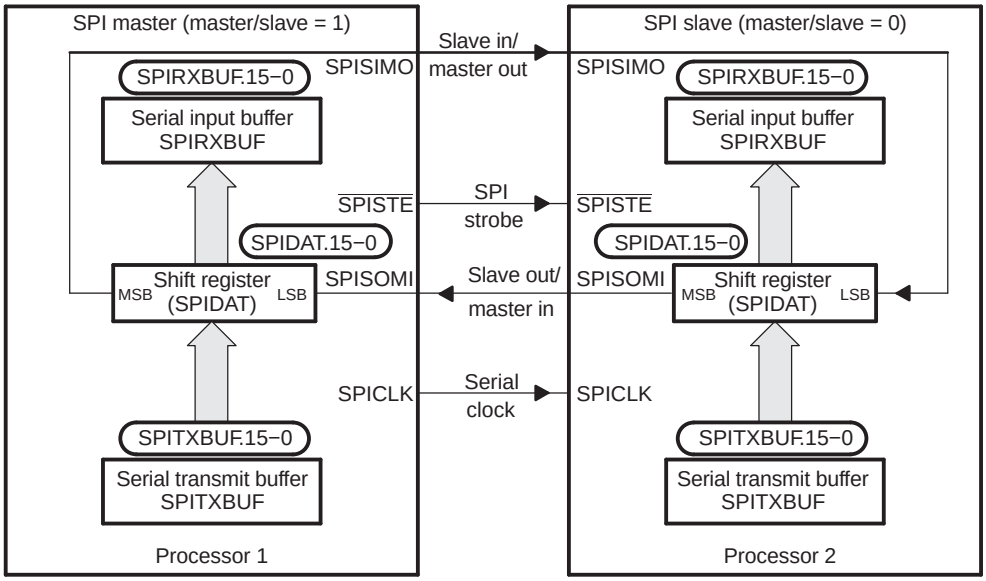
Figure 23-4 shows typical connections of the SPI for communications between two controllers: a master and a slave.

The master data transfer by sending the SPICLK signal. For both the slave and the master, data is shifted out of the shift registers on one edge of the SPICLK and latched into the shift register on the opposite SPICLK clock edge. If the CLK_PHASE bit is high, data is transmitted and received a half-cycle before the SPICLK transition. As a result, both controllers send and receive data simultaneously. The application software determines whether the data is meaningful or dummy data. There are three possible methods for data transmission:

- Master sends data; slave sends dummy data.
- Master sends data; slave sends data.
- Master sends dummy data; slave sends data.

The master can initiate data transfer at any time because it controls the SPICLK signal. The software, however, determines how the master detects when the slave is ready to broadcast data.

**Figure 23-4. SPI Master/Slave Connection**




The SPI can operate in master or slave mode. The MASTER_SLAVE bit selects the operating mode and the source of the SPICLK signal.

### 23.3.2 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.

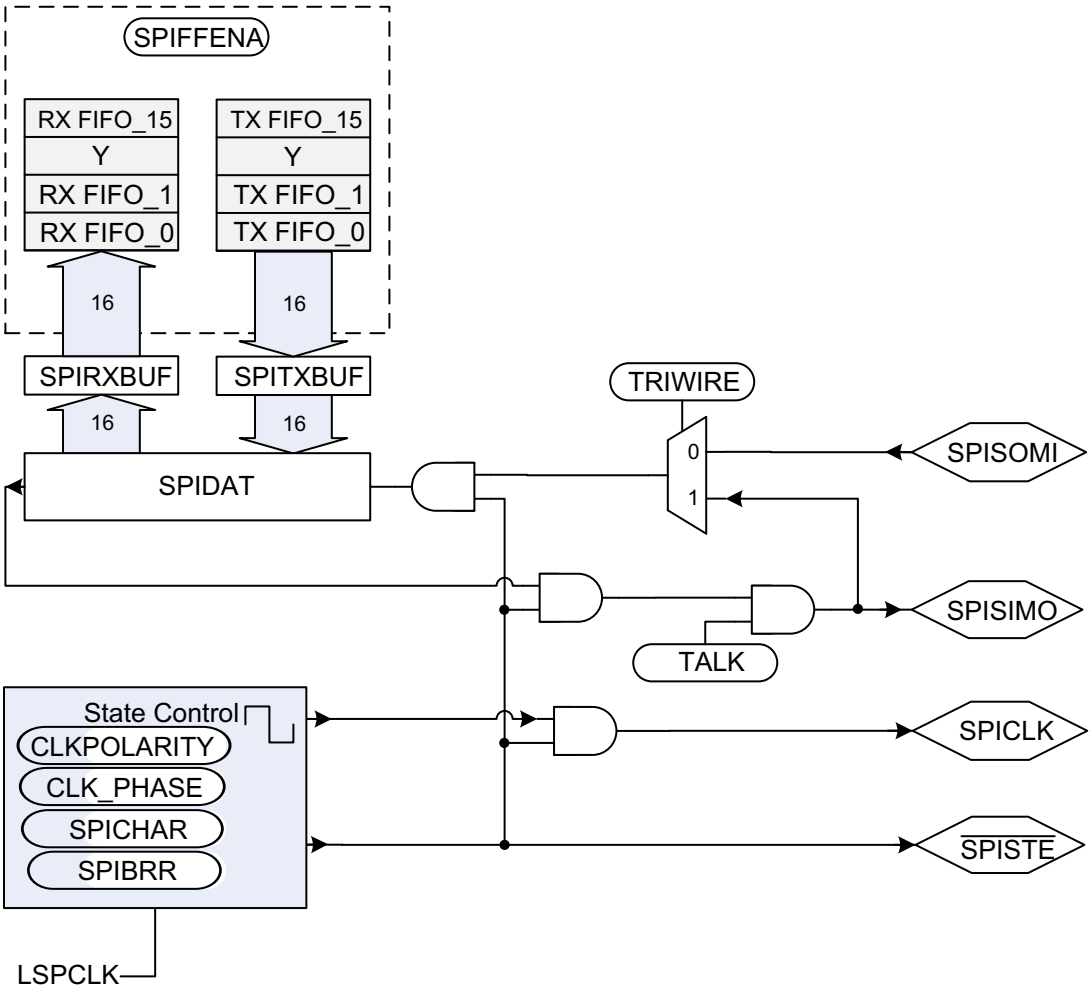
When the specified number of data bits has been shifted through SPIDAT, the following events occur:

- SPIDAT contents are transferred to SPIRXBUF.
- INT_FLAG bit s set to 1.
- If there is valid data in the transmit buffer SPITXBUF, as indicated by the Transmit Buffer Full Flag (BUFFULL_FLAG), this data is transferred to SPIDAT and is transmitted; otherwise, SPICLK stops after all bits have been shifted out of SPIDAT.
- If the SPIINTENA bit is set to 1, an interrupt is asserted.

In a typical application, the $\overline{\text{SPISTE}}$ pin serves as a chip-enable pin for a slave SPI device. This pin is driven low by the master before transmitting data to the slave and is taken high after the transmission is complete.

Figure 23-5 is a block diagram of the SPI in master mode. It shows the basic control blocks available in SPI master mode.

**Figure 23-5. SPI Module Master Configuration**




### 23.3.3 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 should be no greater than the LSPCLK frequency divided by 4.

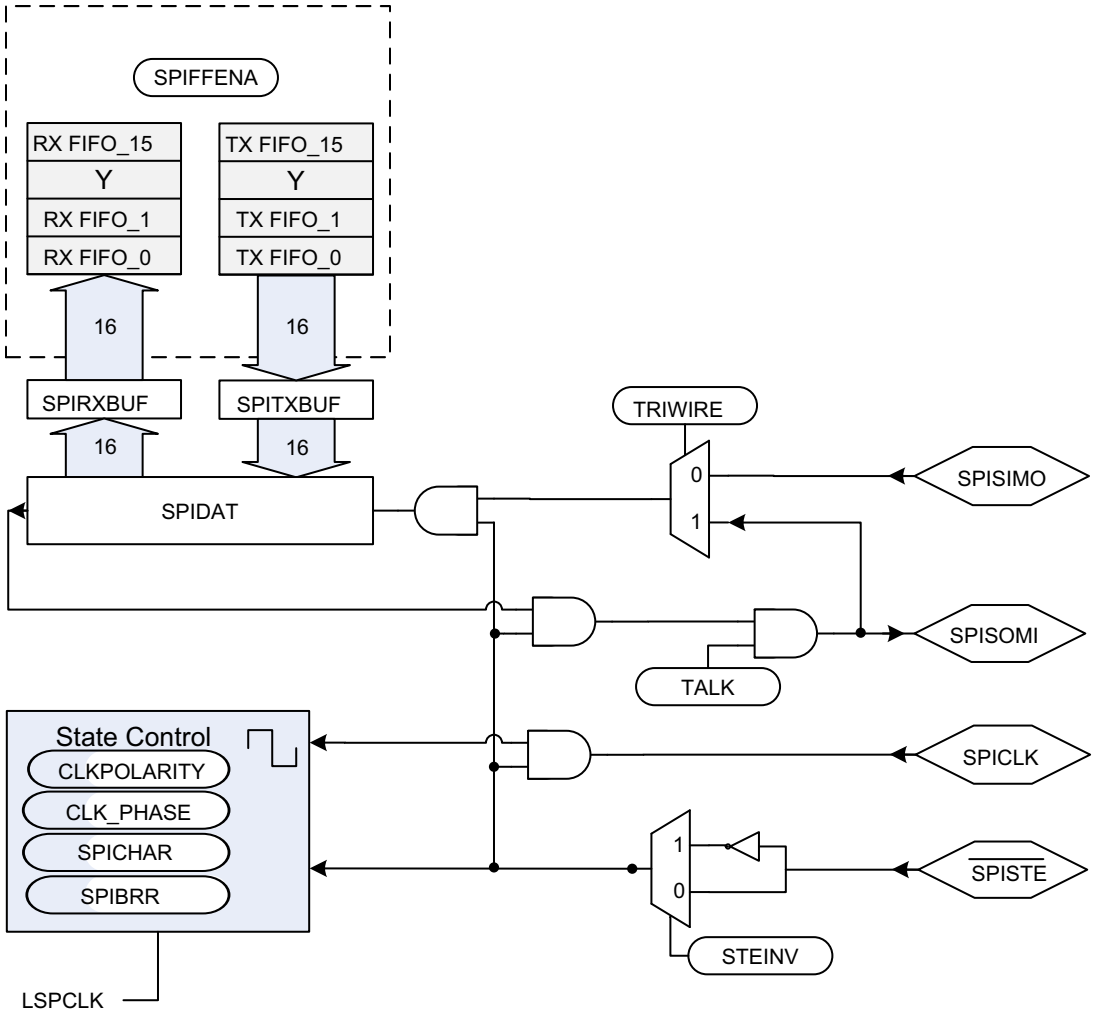
Data written to SPIDAT or SPITXBUF is transmitted to the network when appropriate edges of the SPICLK signal are received from the network master. Data written to the SPITXBUF register will be transferred to the SPIDAT register when all bits of the character to be transmitted have been shifted out of SPIDAT. If no character is currently being transmitted when SPITXBUF is written to, the data will be transferred immediately to SPIDAT. To receive data, the SPI waits for the network master to send the SPICLK signal and then shifts the data on the SPISIMO pin into SPIDAT. If data is to be transmitted by the slave simultaneously, and SPITXBUF has not been previously loaded, the data must be written to SPITXBUF or SPIDAT before the beginning of the SPICLK signal.

When the TALK bit is cleared, data transmission is disabled, and the output line (SPISOMI) is put into the high-impedance state. If this occurs while a transmission is active, the current character is completely transmitted even though SPISOMI is forced into the high-impedance state. This ensures that the SPI is still able to receive incoming data correctly. This TALK bit allows many slave devices to be tied together on the network, but only one slave at a time is allowed to drive the SPISOMI line.

The $\overline{\text{SPISTE}}$ pin operates as the slave-select pin. An active-low signal on the $\overline{\text{SPISTE}}$ pin allows the slave SPI to transfer data to the serial data line; an inactive- high signal causes the slave SPI serial shift register to stop and its serial output pin to be put into the high-impedance state. This allows many slave devices to be tied together on the network, although only one slave device is selected at a time.

Figure 23-6 is a block diagram of the SPI in slave mode. It shows the basic control blocks available in SPI slave mode.

**Figure 23-6. SPI Module Slave Configuration**




### 23.3.4 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 23-1).

***Example 23-1. Transmission of Bit From SPIRXBUF***

Conditions:

1. Transmission character length = 1 bit (specified in bits SPICHAR)
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.

### 23.3.5 Baud Rate Selection

The SPI module supports 125 different baud rates and four different clock schemes. Depending on whether the SPI clock is in slave or master mode, the SPICLK pin can receive an external SPI clock signal or provide the SPI clock signal, respectively.

- In the slave mode, the SPI clock is received on the SPICLK pin from the external source, and can be no greater than the LSPCLK frequency divided by 4.
- In the master mode, the SPI clock is generated by the SPI and is output on the SPICLK pin, and can be no greater than the LSPCLK frequency divided by 4.

**NOTE:** The baud rate should be configured to not exceed the maximum rated GPIO toggle frequency. Refer to the device Data Manual for the maximum GPIO toggle frequency

Example 23-2 shows how to determine the SPI baud rates.

***Example 23-2. Baud Rate Determination***

For SPIBRR = 3 to 127:

$$\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{(\text{SPIBRR} + 1)} \quad (8)$$

For SPIBRR = 0, 1, or 2:

$$\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{4} \quad (9)$$

where:

LSPCLK = Low-speed peripheral clock frequency of the device

SPIBRR = Contents of the SPIBRR in the master SPI device

To determine what value to load into SPIBRR, you must know the device system clock (LSPCLK) frequency (which is device-specific) and the baud rate at which you will be operating.

The following example shows how to calculate the baud rate of the SPI module in standard SPI mode (HS_MODE=0).

***Example 23-3. Baud Rate Calculation in Non-High Speed Mode (HS_MODE=0)***

$$SPI\ Baud\ Rate = \frac{LSPCLK}{SPIBRR + 1'} \quad LSPCLK = 50\ MHz$$
$$= \frac{50 \times 10^6}{3 + 1}$$
$$= 12.5\ Mbps \tag{10}$$

### 23.3.6 SPI Clocking Schemes

The clock polarity select bit (CLKPOLARITY) and the clock phase select bit (CLK_PHASE) control four different clocking schemes on the SPICLK pin. CLKPOLARITY selects the active edge, either rising or falling, of the clock. CLK_PHASE selects a half-cycle delay of the clock. The four different clocking schemes are as follows:

- Falling Edge Without Delay. The SPI transmits data on the falling edge of the SPICLK and receives data on the rising edge of the SPICLK.
- Falling Edge With Delay. The SPI transmits data one half-cycle ahead of the falling edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
- Rising Edge Without Delay. The SPI transmits data on the rising edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
- Rising Edge With Delay. The SPI transmits data one half-cycle ahead of the rising edge of the SPICLK signal and receives data on the rising edge of the SPICLK signal.

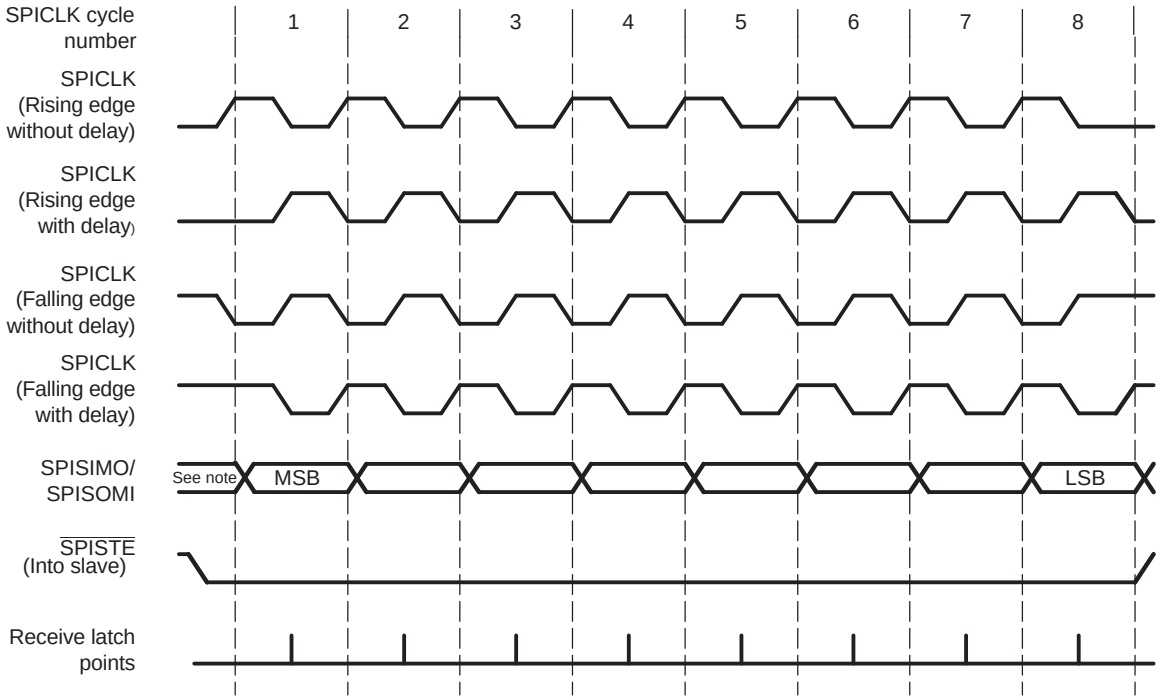
The selection procedure for the SPI clocking scheme is shown in Table 23-3. Examples of these four clocking schemes relative to transmitted and received data are shown in Figure 23-7.

**Table 23-3. SPI Clocking Scheme Selection Guide**

| SPICLK Scheme | CLKPOLARITY | 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.

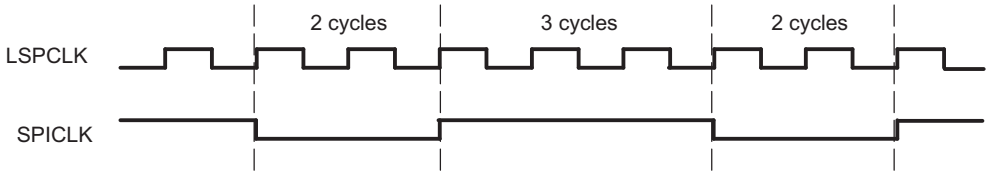
**Figure 23-7. SPICLK Signal Options**




**Note:** Previous data bit

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 23-8.

**Figure 23-8. SPI: SPICLK-LSPCLK Characteristic When (BRR + 1) is Odd, BRR > 3, and CLKPOLARITY = 1**




### *23.3.7 SPI FIFO Description*

The following steps explain the FIFO features and help with programming the SPI FIFOs:

1. **Reset.** At reset the SPI powers up in standard SPI mode and the FIFO function is disabled. The FIFO registers SPIFFTX, SPIFFRX and SPIFFCT remain inactive.
2. **Standard SPI.** The standard 28x SPI mode will work with SPIINT/SPIRXINT as the interrupt source.
3. **Mode change.** FIFO mode is enabled by setting the SPIFFENA bit to 1 in the SPIFFTX register. SPIRST can reset the FIFO mode at any stage of its operation.
4. **Active registers.** All the SPI registers and SPI FIFO registers SPIFFTX, SPIFFRX, and SPIFFCT will be active.
5. **Interrupts.** FIFO mode has two interrupts one for the transmit FIFO, SPITXINT and one for the receive FIFO, SPIRXINT. SPIRXINT is the common interrupt for SPI FIFO receive, receive error and receive FIFO overflow conditions. The single SPIINT for both transmit and receive sections of the standard SPI will be disabled and this interrupt will service as SPI receive FIFO interrupt. For more information, refer to Section 23.2.3

6. **Buffers.** Transmit and receive buffers are each supplemented with a 16 word FIFO. The one-word transmit buffer (SPITXBUF) of the standard SPI functions as a transition buffer between the transmit FIFO and shift register. The one-word transmit buffer will be loaded from transmit FIFO only after the last bit of the shift register is shifted out.
7. **Delayed transfer.** The rate at which transmit words in the FIFO are transferred to transmit shift register is programmable. The SPIFFCT register bits (7−0) FFTXDLY7−FFTXDLY0 define the delay between the word transfer. The delay is defined in number SPI serial clock cycles. The 8-bit register could define a minimum delay of 0 SPICLK cycles and a maximum of 255 SPICLK cycles. With zero delay, the SPI module can transmit data in continuous mode with the FIFO words shifting out back to back. With the 255 clock delay, the SPI module can transmit data in a maximum delayed mode with the FIFO words shifting out with a delay of 255 SPICLK cycles between each words. The programmable delay facilitates glueless interface to various slow SPI peripherals, such as EEPROMs, ADC, DAC, and so on.
8. **FIFO status bits.** Both transmit and receive FIFOs have status bits TXFFST or RXFFST that define the number of words available in the FIFOs at any time. The transmit FIFO reset bit (TXFIFO) and receive reset bit (RXFIFO) will reset the FIFO pointers to zero when these bits are set to 1. The FIFOs will resume operation from start once these bits are cleared to zero.
9. **Programmable interrupt levels.** Both transmit and receive FIFOs can generate CPU interrupts and DMA triggers. The transmit interrupt (SPITXINT) is generated whenever the transmit FIFO status bits (TXFFST) match (less than or equal to) the interrupt trigger level bits (TXFFIL). The receive interrtupt (SPIRXINT) is generated whenever the receive FIFO status bits (RXFFST) match (greater than or equal to) the interrupt trigger level RXFFIL. This provides a programmable interrupt trigger for transmit and receive sections of the SPI. The default value for these trigger level bits will be 0x11111 for receive FIFO and 0x00000 for transmit FIFO, respectively.

### *23.3.8 SPI DMA Transfers*

#### 23.3.8.1 Transmitting Data Using SPI with DMA

When using the DMA with the TX FIFO, the DMA Burst Size (DMA_BURST_SIZE) should be no greater than 16 – TXFFIL in order to prevent the DMA from writing to an already full FIFO. This will lead to data loss and is not recommended.

For complete data transmission, please follow these steps:

1. Calculate the total number or words to be transmitted. [NUM_WORDS]
2. Decide the transmit FIFO level. [TXFFIL]
3. Calculate the number of DMA transfers. [DMA_TRANSFER_SIZE]
4. Calculate the size of the DMA Burst. [DMA_BURST_SIZE]
5. Configure DMA using calculated values.
6. Configure SPI with FIFO using the calculated values.

To transfer 128 words to SPI using the DMA:

NUM_WORDS: 128

TXFFIL: 8

DMA_TRANSFER_SIZE: (NUM_WORDS /TXFFIL) – 1 = (128/8) – 1 = 15 (16 transfers)

DMA_BURST_SIZE: (16 – TXFFIL) – 1 = (16 – 8) – 1 = 7 (8 words per burst)

#### 23.3.8.2 Receiving Data Using SPI with DMA

When using the DMA with the RX FIFO, the DMA Burst Size (BURST_SIZE) should be no greater than RXFFIL in order to prevent the DMA from reading from an empty FIFO. To ensure that the DMA correctly receives all data from the RX FIFO, the DMA Burst Size should equal RXFFIL and also be an integer divisor of the total number of SPI transmissions.

For complete data reception, please follow these steps:

1. Calculate the number of words to be received. [NUM_WORDS]
2. Calculate the necessary FIFO level [RXFFIL]

3. Calculate the total number of DMA transfers. [DMA_TRANSFER_SIZE]
4. Calculate the size of the DMA Burst. [DMA_BURST_SIZE]
5. Configure DMA using the calculated values.
6. Configure SPI with FIFO using the calculated values.

To receive 200 words from SPI using the DMA:
NUM_WORDS = 200
RXFFIL: 4
DMA_TRANSFER_SIZE: (NUM_WORDS /RXFFIL) – 1 = (200/4) – 1 = 49 (50 transfers)
DMA_BURST_SIZE = RXFFIL-1 = 3 (4 words per burst)

### 23.3.9 SPI High-Speed Mode

The SPI module is capable of reaching full-duplex communication speeds up to LSPCLK/4 (where LSPCLK equals SYSCLK). For the maximum rated speed, refer the device Data Manual.

In order to achieve the maximum full-duplex speeds, the following restrictions are placed on the design:

- Single master to single slave configuration is supported.
- Loading on the pins must not exceed the value stated in the device Data Manual.

When configuring the GPIOs to support High-Speed mode, refer to Section 23.2.2.1 for more information.

### 23.3.10 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 23-4 indicates the pin function differences between 3-wire and 4-wire SPI mode for a master and slave SPI.

**Table 23-4. 4-wire vs. 3-wire SPI Pin Functions**

| 4-wire SPI | 3-wire SPI (Master) | 3-wire SPI(Slave) |
|---|---|---|
| SPICLKx | SPICLKx | SPICLKx |
| SPISTEx | SPISTEx | SPISTEx |
| SPISIMOx | SPIMOMIx | Free |
| SPISOMIx | Free | SPISISOx |

Because in 3-wire mode, the receive and transmit paths within the SPI are connected, any data transmitted by the SPI module is also received by itself. The application software must take care to perform a dummy read to clear the SPI data register of the additional received data.

The TALK bit plays an important role in 3-wire SPI mode. The bit must be set to transmit data and cleared prior to reading data. In master mode, in order to initiate a read, the application software must write dummy data to the SPI data register (SPIDAT or SPIRXBUF) while the TALK bit is cleared (no data is transmitted out the SPIMOMI pin) before reading from the data register.

Figure 23-9 and Figure 23-10 illustrate 3-wire master and slave mode.

**Figure 23-9. SPI 3-wire Master Mode**

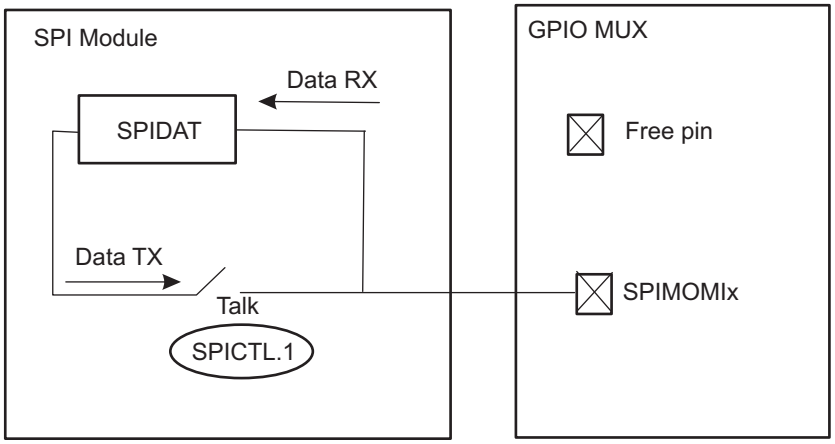



**Figure 23-10. SPI 3-wire Slave Mode**

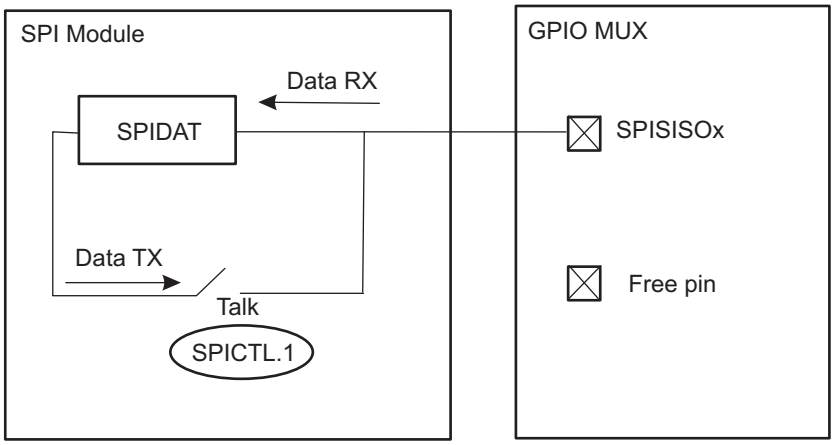



Table 23-5 indicates how data is received or transmitted in the various SPI modes while the TALK bit is set or cleared.

**Table 23-5. 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 |

## 23.4 Programming Procedure

This section describes the procedure for configuring the SPI for the various modes of operation.

### 23.4.1 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

### 23.4.2 Configuring the SPI

This section describes the procedure in which to configure the SPI module for operation. To prevent unwanted and unforeseen events from occurring during or as a result of initialization changes, clear the SPISWRESET bit before making initialization changes, and then set this bit after initialization is complete. While the SPI is held in reset (SPISWRESET = 0), configuration may be changed in any order. The following list shows the the SPI configuration procedure in a logical order. However, the SPI registers can be written with single 16-bit writes, so the order is not required with the exception of SPISWRESET.

To change the SPI configuration:

Step 1. Clear the SPI Software Reset bit (SPISWRESET) to 0 to force the SPI to the reset state.

Step 2. Configure the SPI as desired:

- Select either master or slave mode (MASTER_SLAVE).
- Choose SPICLK polarity and phase (CLKPOLARITY and CLK_PHASE).
- Set the desired baud rate (SPIBRR).
- Enable high speed mode if desired (HS_MODE).
- Set the SPI character length (SPICHAR).
- Clear the SPI Flags (OVERRUN_FLAG, INT_FLAG).
- Enable $\overline{\text{SPISTE}}$ inversion (STEINV).
- Enable 3-wire mode (TRIWIRE).
- If using FIFO enhancements:
  - Enable the FIFO enhancements (SPIFFENA).
  - Clear the FIFO Flags (TXFFINTCLR, RXFFOVFCLR, and RXFFINTCLR).
  - Release transmit and receive FIFO resets (TXFIFO and RXFIFORESET).
  - Release SPI FIFO channels from reset (SPIRST).

Step 3. If interrupts are used:

- In non-FIFO mode, enable the receiver overrun and/or SPI interrupts (OVERRUNINTENA and SPIINTENA).
- In FIFO mode, set the transmit and receive interrupt levels (TXFFIL and RXFFIL) then enable the interrupts (TXFFIENA and RXFFIENA).

Step 4. Set SPISWRESET to 1 to release the SPI from the reset state.

**NOTE:** Do not change the SPI configuration when communication is in progress.

### 23.4.3 Configuring the SPI for High-Speed Mode

In order to achieve the maximum rated speeds, the following settings must be made. This example assumes that the device is operating at 100 MHz.

Set LSPCLK equal to SYSCLK:

```
ClkCfgRegs.LOSPCP.bit.LSPCLKDIV = 0;
```

Select the appropriate Pin Mux options in GPIO_CTRL_REGS.

During the SPI configuration procedure:

Set HS_MODE to 1.

```
SpiaRegs.SPICCR.bit.HS_MODE = 0x1;
```

Set SPIBRR to 3. SPICLK = LSPCLK/(SPIBRR+1) = 25MHz

```
SpiaRegs.SPIBRR = 0x3;
```

There are no other differences in the configuration from normal SPI operation. Sending and receiving data, interrupts, and DMA operation will operate without change.

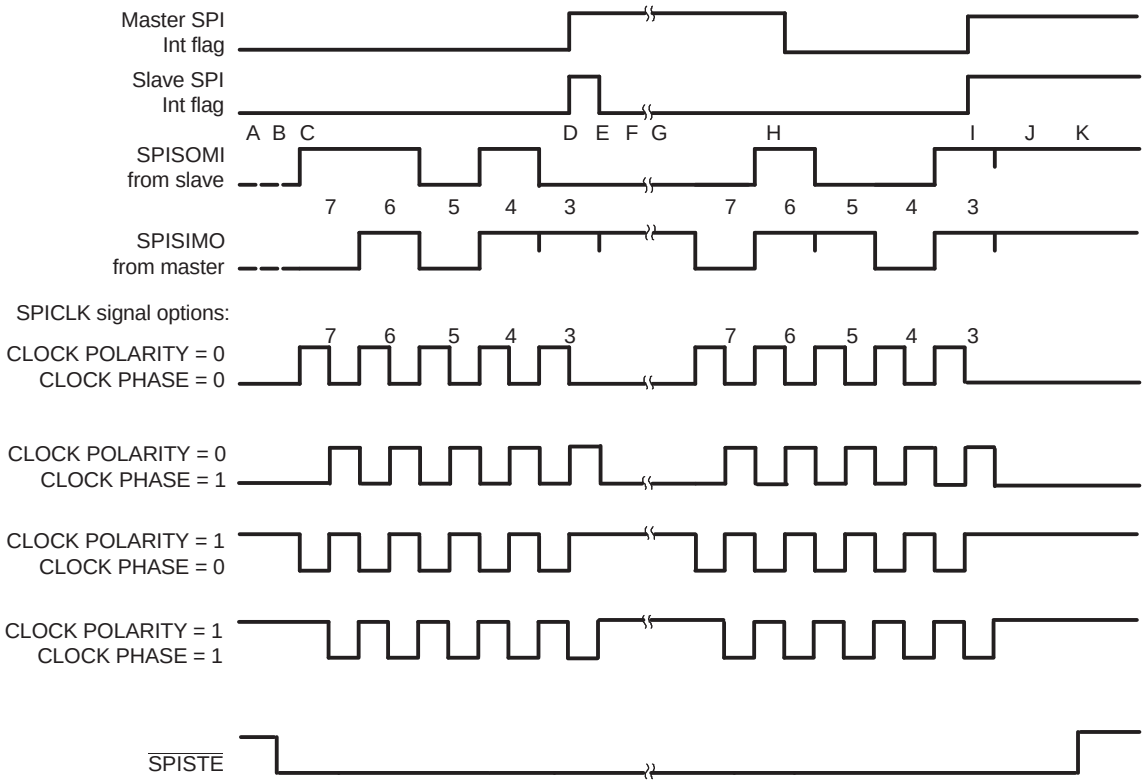
### 23.4.4 Data Transfer Example

The timing diagram shown in Figure 23-11 illustrates an SPI data transfer between two devices using a character length of five bits with the SPICLK being symmetrical.

The timing diagram with SPICLK asymmetrical (Figure 23-8) shares similar characterizations with Figure 23-11 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 23-11 is applicable for 8-bit SPI only and is not for 28x devices that are capable of working with 16-bit data. The figure is shown for illustrative purposes only.

**Figure 23-11. Five Bits per Character**




A Slave writes 0D0h to SPIDAT and waits for the master to shift out the data.

B Master sets the slave $\overline{\text{SPISTE}}$ signal low (active).

C Master writes 058h to SPIDAT, which starts the transmission procedure.

D First byte is finished and sets the interrupt flags.

E Slave reads 0Bh from its SPIRXBUF (right-justified).

F Slave writes 04Ch to SPIDAT and waits for the master to shift out the data.

G Master writes 06Ch to SPIDAT, which starts the transmission procedure.

H Master reads 01Ah from the SPIRXBUF (right−justified).

I Second byte is finished and sets the interrupt flags.

J Master reads 89h and the slave reads 8Dh from their respective SPIRXBUF. After the user's software masks off the unused bits, the master receives 09h and the slave receives 0Dh.

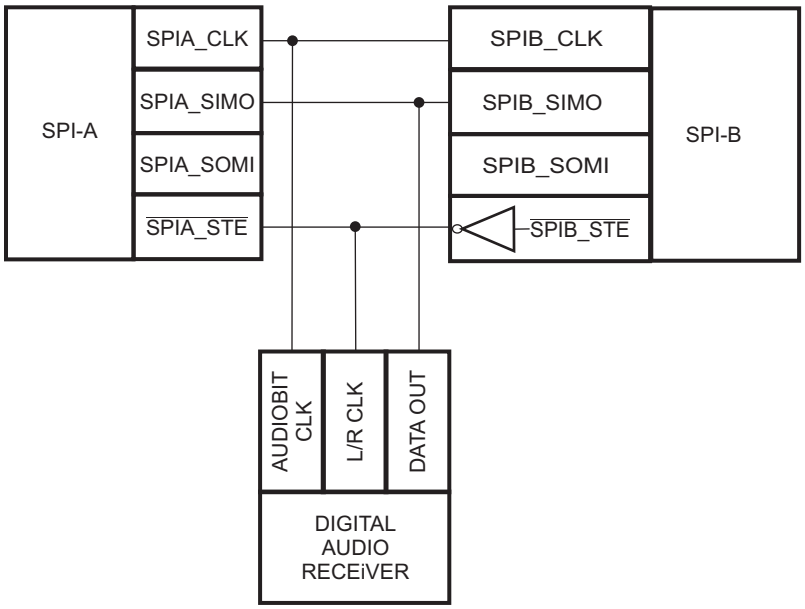
K Master clears the slave $\overline{\text{SPISTE}}$ signal high (inactive).

### 23.4.5 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 23-12.

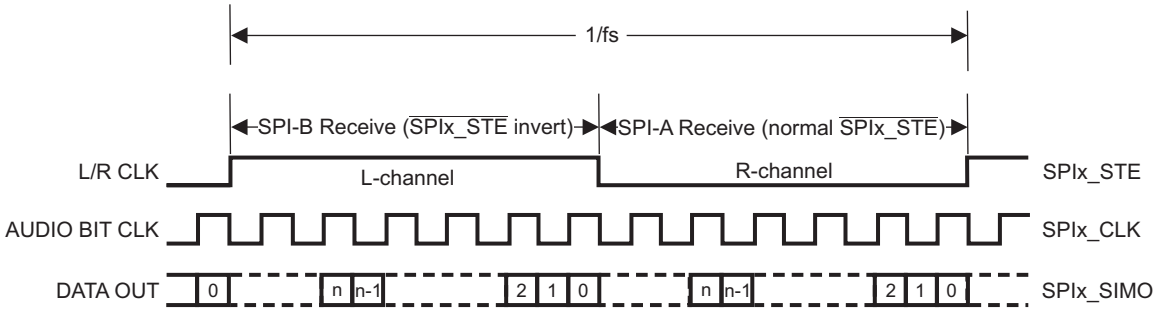
**NOTE:** This configuration is only applicable to slave mode (MASTER_SLAVE = 0). When the SPI is configured as master (MASTER_SLAVE = 1), the STEINV bit will have no effect on the $\overline{\text{SPISTE}}$ pin.

**Figure 23-12. SPI Digital Audio Receiver Configuration Using Two SPIs**




Standard 28x SPI timing requirements limit the number of digital audio interface formats supported using the 2-SPI configuration with the STEINV bit. See your device-specific data sheet electricals for SPI timing requirements. With the SPI clock phase configured such that the 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 23-13.

**Figure 23-13. Standard Right-Justified Digital Audio Data Format**



## 23.5 Registers

### 23.5.1 Serial Peripheral Interface Base Addresses

**Table 23-6. 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 |

### 23.5.1.1 SPI_REGS Registers

Table 23-7 lists the memory-mapped registers for the SPI_REGS. All register offset addresses not listed in Table 23-7 should be considered as reserved locations and the register contents should not be modified.

**Table 23-7. 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 23-8 shows the codes that are used for access types in this section.

**Table 23-8. SPI_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| RC | C<br>R | to Clear<br>Read |
| **Write Type** | | |
| W | W | Write |
| W1C | 1C<br>W | 1 to clear<br>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. |

#### 23.5.1.1.1 SPICCR Register (Offset = 0h) [reset = 0h]

SPICCR is shown in Figure 23-14 and described in Table 23-9.

Return to Summary Table.

SPI Configuration Control Register

**Figure 23-14. 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 23-9. 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. |

**Table 23-9. 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 |

#### 23.5.1.1.2 SPICTL Register (Offset = 1h) [reset = 0h]

SPICTL is shown in Figure 23-15 and described in Table 23-10.

Return to Summary Table.

SPI Operation Control Register

**Figure 23-15. 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 23-10. 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. |

**Table 23-10. 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. |

#### 23.5.1.1.3 SPISTS Register (Offset = 2h) [reset = 0h]

SPISTS is shown in Figure 23-16 and described in Table 23-11.

Return to Summary Table.

SPI Status Register

**Figure 23-16. 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 23-11. 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. |

**Table 23-11. 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 |

#### 23.5.1.1.4 SPIBRR Register (Offset = 4h) [reset = 0h]

SPIBRR is shown in Figure 23-17 and described in Table 23-12.

Return to Summary Table.

SPI Baud Rate Register

**Figure 23-17. 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 23-12. 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>where: LSPCLK = Function of CPU clock frequency X low-speed peripheral clock of the device SPIBRR = Contents of the SPIBRR in the master SPI device.<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 |

#### 23.5.1.1.5 SPIRXEMU Register (Offset = 6h) [reset = 0h]

SPIRXEMU is shown in Figure 23-18 and described in Table 23-13.

Return to Summary Table.

SPI Emulation Buffer Register

**Figure 23-18. SPIRXEMU Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ERXBn | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 23-13. 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 |

#### 23.5.1.1.6 SPIRXBUF Register (Offset = 7h) [reset = 0h]

SPIRXBUF is shown in Figure 23-19 and described in Table 23-14.

Return to Summary Table.

SPI Serial Input Buffer Register

**Figure 23-19. SPIRXBUF Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RXBn | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 23-14. 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 |

#### 23.5.1.1.7 SPITXBUF Register (Offset = 8h) [reset = 0h]

SPITXBUF is shown in Figure 23-20 and described in Table 23-15.

Return to Summary Table.

SPI Serial Output Buffer Register

**Figure 23-20. SPITXBUF Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TXBn | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 23-15. 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 |

#### 23.5.1.1.8 SPIDAT Register (Offset = 9h) [reset = 0h]

SPIDAT is shown in Figure 23-21 and described in Table 23-16.

Return to Summary Table.

SPI Serial Data Register

**Figure 23-21. SPIDAT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SDATn | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 23-16. 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, see the CLOCK POLARITY bit (SPICCR.6) described in Section 10.2.1.1 and the CLOCK PHASE bit (SPICTL.3) described in Section 10.2.1.2, for the requirements.<br>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 |

#### 23.5.1.1.9 SPIFFTX Register (Offset = Ah) [reset = A000h]

SPIFFTX is shown in Figure 23-22 and described in Table 23-17.

Return to Summary Table.

SPI FIFO Transmit Register

**Figure 23-22. 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 23-17. 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. |

**Table 23-17. 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. |

#### 23.5.1.1.10 SPIFFRX Register (Offset = Bh) [reset = 201Fh]

SPIFFRX is shown in Figure 23-23 and described in Table 23-18.

Return to Summary Table.

SPI FIFO Receive Register

**Figure 23-23. 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 23-18. 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. |

**Table 23-18. SPIFFRX Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4-0 | RXFFIL | R/W | 1Fh | Receive FIFO Interrupt Level Bits<br>11111 Receive FIFO generates an interrupt when the FIFO status bits (RXFFST4-0) are greater than or equal to the FIFO level bits (RXFFIL4-0). The default value of these bits after reset is 11111. This avoids frequent interrupts after reset, as the receive FIFO will be empty most of the time.<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. |

#### 23.5.1.1.11 SPIFFCT Register (Offset = Ch) [reset = 0h]

SPIFFCT is shown in Figure 23-24 and described in Table 23-19.

Return to Summary Table.

SPI FIFO Control Register

**Figure 23-24. 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 23-19. 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. |

#### 23.5.1.1.12 SPIPRI Register (Offset = Fh) [reset = 0h]

SPIPRI is shown in Figure 23-25 and described in Table 23-20.

Return to Summary Table.

SPI Priority Control Register

**Figure 23-25. 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-0h | R/W-0h | R/W-0h | R-0h | | R/W-0h | R/W-0h |

**Table 23-20. SPIPRI Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-7 | RESERVED | R | 0h | Reserved |
| 6 | RESERVED | R | 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 after midway in the bit stream while TSUSPEND is asserted. Once TSUSPEND is deasserted without a system reset, the remainder of the bits pending in the DATBUF are shifted. Example: If SPIDAT has shifted 3 out of 8 bits, the communication freezes right there. However, if TSUSPEND is later deasserted without resetting the SPI, SPI starts transmitting from where it had stopped (fourth bit in this case) and will transmit 8 bits from that point. The SCI module operates differently.<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>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) |

**Table 23-20. 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. |

# Serial Communications Interface (SCI)

This chapter describes the features and operation of the serial communication interface (SCI) module. SCI is a two−wire asynchronous serial port, commonly known as a UART. The SCI modules support digital communications between the CPU and other asynchronous peripherals that use the standard non-return-to-zero (NRZ) format. The SCI receiver and transmitter each have a 16-level deep FIFO for reducing servicing overhead, and each has its own separate enable and interrupt bits. Both can be operated independently for half-duplex communication, or simultaneously for full-duplex communication.

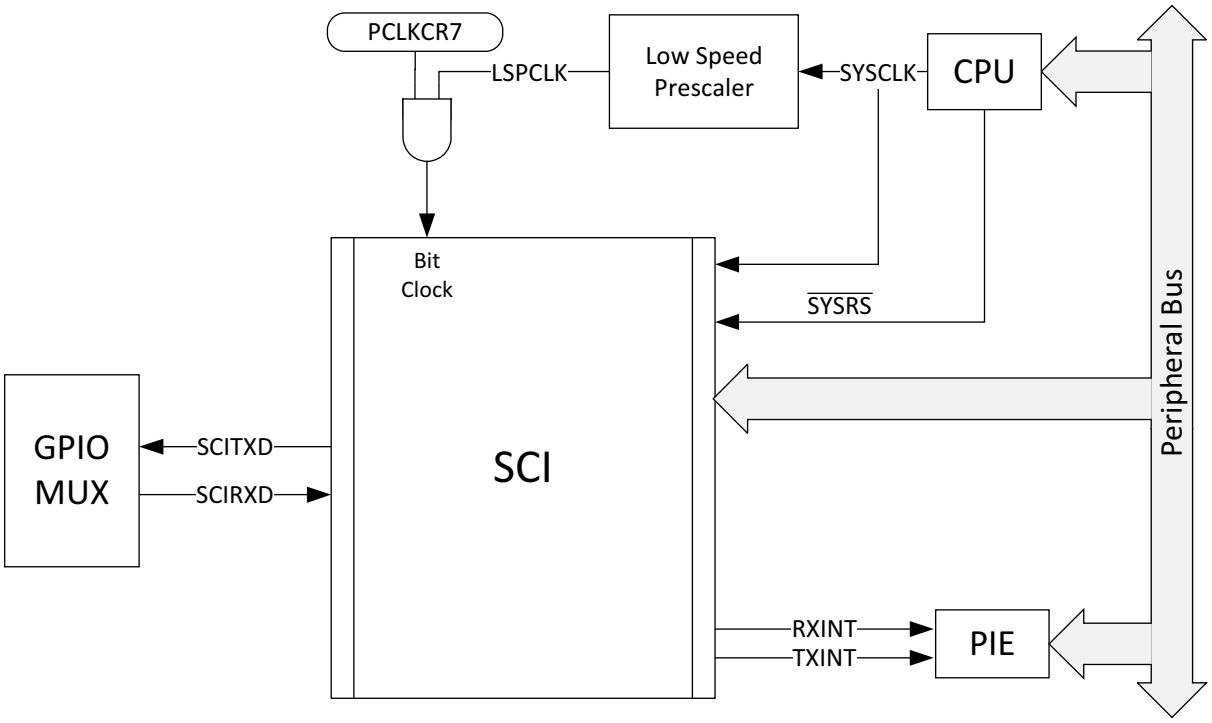
To specify data integrity, the SCI checks received data for break detection, parity, overrun, and framing errors. The bit rate is programmable to different speeds through a 16-bit baud-select register.

**Topic** **Page**

## 24.1 Enhanced SCI Module Overview

The SCI interfaces are shown in Figure 24-1.

**Figure 24-1. SCI CPU Interface**




Features of the SCI module include:

- Two external pins:
  - SCITXD: SCI transmit-output pin
  - SCIRXD: SCI receive-input pin

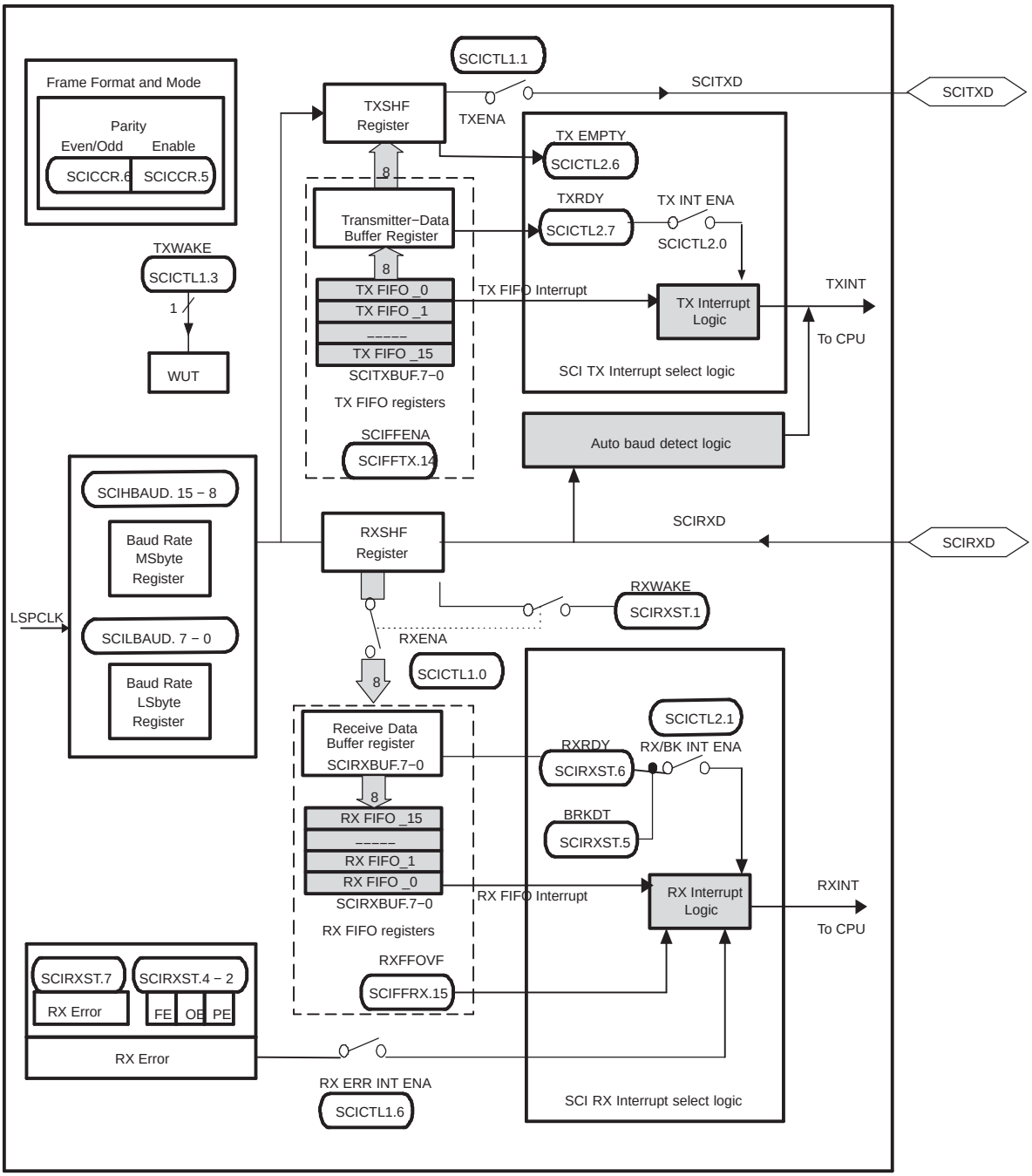
  Both pins can be used as GPIO if not used for SCI.
- Baud rate programmable to 64K different rates
- Data-word format
  - One start bit
  - Data-word length programmable from one to eight bits
  - Optional even/odd/no parity bit
  - One or two stop bits
  - An extra bit to distinguish addresses from data (address bit mode only)
- Four error-detection flags: parity, overrun, framing, and break detection
- Two wake-up multiprocessor modes: idle-line and address bit
- Half- or full-duplex operation
- Double-buffered receive and transmit functions
- Transmitter and receiver operations can be accomplished through interrupt-driven or polled algorithms with status flags.
- Separate enable bits for transmitter and receiver interrupts (except BRKDT)
- NRZ (non-return-to-zero) format
- 13 SCI module control registers located in the control register frame beginning at address 7050h

Enhanced features include:

- Auto-baud-detect hardware logic
- 16-level transmit/receive FIFO

Figure 24-2 shows the SCI module block diagram. The SCI port operation is configured and controlled by the registers listed in Section 24.14 of this chapter.

**Figure 24-2. Serial Communications Interface (SCI) Module Block Diagram**



## 24.2 Architecture

The major elements used in full-duplex operation are shown in Figure 24-2 and include:

- A transmitter (TX) and its major registers (upper half of Figure 24-2)
  - SCITXBUF — transmitter data buffer register. Contains data (loaded by the CPU) to be transmitted
  - TXSHF register — transmitter shift register. Accepts data from register SCITXBUF and shifts data onto the SCITXD pin, one bit at a time
- A receiver (RX) and its major registers (lower half of Figure 24-2)
  - 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.

## 24.3 SCI Module Signal Summary

A summarized description of each SCI signal name is shown in Table 24-1.

**Table 24-1. 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 |

## 24.4 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

Some IO functionality is defined by GPIO register settings independent of this peripheral. For input signals, the GPIO input qualification should be set to asynchronous mode by setting the appropriate GPxQSELn register bits to 11b. The internal pullups can be configured in the GPyPUD register.

See the *GPIO* chapter for more details on GPIO mux and settings.

## 24.5 Multiprocessor and Asynchronous Communication Modes

The SCI has two multiprocessor protocols, the idle-line multiprocessor mode (see Section 24.8) and the address-bit multiprocessor mode (see Section 24.9). 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 24.10) 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

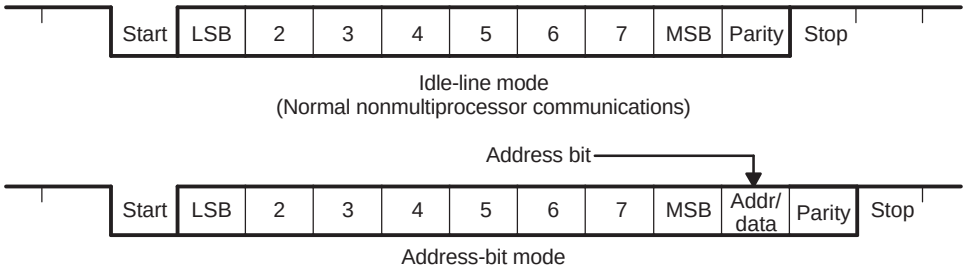
## 24.6 SCI Programmable Data Format

SCI data, both receive and transmit, is in NRZ (non-return-to-zero) format. The NRZ data format, shown in Figure 24-3, consists of:

- One start bit
- One to eight data bits
- An even/odd parity bit (optional)
- One or two stop bits
- An extra bit to distinguish addresses from data (address-bit mode only)

The basic unit of data is called a character and is one to eight bits in length. Each character of data is formatted with a start bit, one or two stop bits, and optional parity and address bits. A character of data with its formatting information is called a frame and is shown in Figure 24-3.

**Figure 24-3. 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 24-2.

**Table 24-2. Programming the Data Format Using SCICCR**

| Bit(s) | Bit Name | Designation | Functions |
|---|---|---|---|
| 2-0 | SCI CHAR2-0 | SCICCR.2:0 | Select the character (data) length (one to eight bits). |
| 5 | PARITY ENABLE | SCICCR.5 | Enables the parity function if set to 1, or disables the parity function if cleared to 0. |
| 6 | EVEN/ODD PARITY | SCICCR.6 | If parity is enabled, selects odd parity if cleared to 0 or even parity if set to 1. |
| 7 | STOP BITS | SCICCR.7 | Determines the number of stop bits transmitted—one stop bit if cleared to 0 or two stop bits if set to 1. |

## 24.7 SCI Multiprocessor Communication

The multiprocessor communication format allows one processor to efficiently send blocks of data to other processors on the same serial link. On one serial line, there should be only one transfer at a time. In other words, there can be only one talker on a serial line at a time.

### Address Byte

The first byte of a block of information that the talker sends contains an address byte that is read by all listeners. Only listeners with the correct address can be interrupted by the data bytes that follow the address byte. The listeners with an incorrect address remain uninterrupted until the next address byte.

**Sleep Bit**

All processors on the serial link set the SCI SLEEP bit (bit 2 of SCICTL1) to 1 so that they are interrupted only when the address byte is detected. When a processor reads a block address that corresponds to the CPU device address as set by your application software, your program must clear the SLEEP bit to enable the SCI to generate an interrupt on receipt of each data byte.

Although the receiver still operates when the SLEEP bit is 1, it does not set RXRDY, RXINT, or any of the receiver error status bits to 1 unless the address byte is detected and the address bit in the received frame is a 1 (applicable to address-bit mode). The SCI does not alter the SLEEP bit; your software must alter the SLEEP bit.

### *24.7.1 Recognizing the Address Byte*

A processor recognizes an address byte differently, depending on the multiprocessor mode used. For example:

- The idle-line mode (Section 24.8) leaves a quiet space before the address byte. This mode does not have an extra address/data bit and is more efficient than the address-bit mode for handling blocks that contain more than ten bytes of data. The idle-line mode should be used for typical non-multiprocessor SCI communication.
- The address-bit mode (Section 24.9) adds an extra bit (that is, an address bit) into every byte to distinguish addresses from data. This mode is more efficient in handling many small blocks of data because, unlike the idle mode, it does not have to wait between blocks of data. However, at a high transmit speed, the program is not fast enough to avoid a 10-bit idle in the transmission stream.

### *24.7.2 Controlling the SCI TX and RX Features*

The multiprocessor mode is software selectable via the ADDR/IDLE MODE bit (SCICCR, bit 3). Both modes use the TXWAKE flag bit (SCICTL1, bit 3), RXWAKE flag bit (SCIRXST, bit1), and the SLEEP flag bit (SCICTL1, bit 2) to control the SCI transmitter and receiver features of these modes.

### *24.7.3 Receipt Sequence*

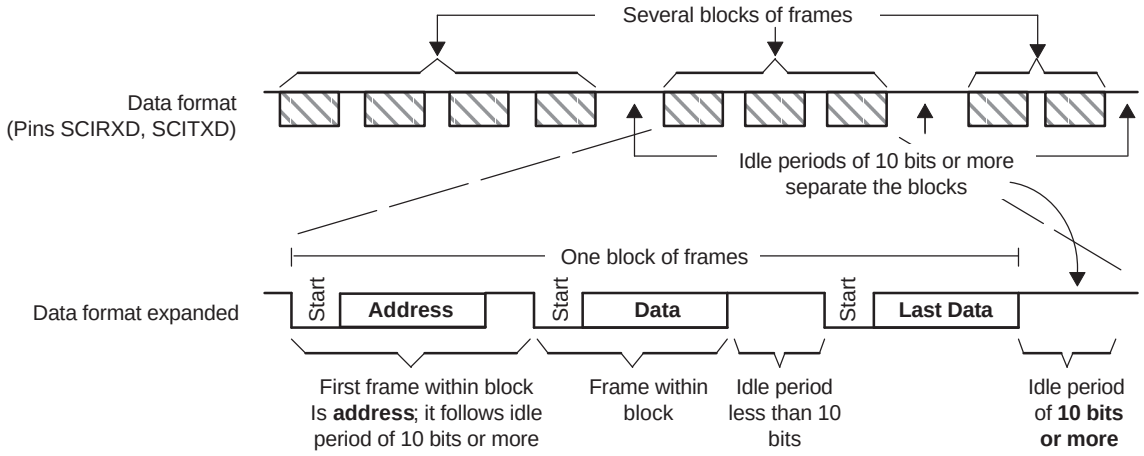
In both multiprocessor modes, the receive sequence is as follows:

1. At the receipt of an address block, the SCI port wakes up and requests an interrupt (bit number 1 RX/BK INT ENA-of SCICTL2 must be enabled to request an interrupt). It reads the first frame of the block, which contains the destination address.
2. A software routine is entered through the interrupt and checks the incoming address. This address byte is checked against its device address byte stored in memory.
3. If the check shows that the block is addressed to the device CPU, the CPU clears the SLEEP bit and reads the rest of the block. If not, the software routine exits with the SLEEP bit still set, and does not receive interrupts until the next block start.

## 24.8 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 24-4 (ADDR/IDLE MODE bit is bit 3 of SCICCR).

**Figure 24-4. Idle-Line Multiprocessor Communication Format**




### 24.8.1 Idle-Line Mode Steps

The steps followed by the idle-line mode:

Step 1. SCI wakes up after receipt of the block-start signal.

Step 2. The processor recognizes the next SCI interrupt.

Step 3. The interrupt service routine compares the received address (sent by a remote transmitter) to its own.

Step 4. If the CPU is being addressed, the service routine clears the SLEEP bit and receives the rest of the data block.

Step 5. If the CPU is not being addressed, the SLEEP bit remains set. This lets the CPU continue to execute its main program without being interrupted by the SCI port until the next detection of a block start.

### 24.8.2 Block Start Signal

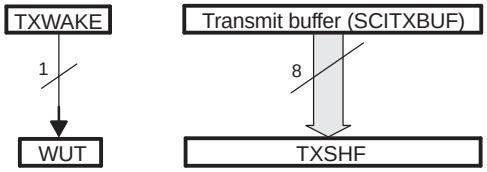
There are two ways to send a block-start signal:

1. **Method 1:** Deliberately leave an idle time of ten bits or more by delaying the time between the transmission of the last frame of data in the previous block and the transmission of the address frame of the new block.
2. **Method 2:** The SCI port first sets the TXWAKE bit (SCICTL1, bit 3) to 1 before writing to the SCITXBUF register. This sends an idle time of exactly 11 bits. In this method, the serial communications line is not idle any longer than necessary. (A don't care byte has to be written to SCITXBUF after setting TXWAKE, and before sending the address, so as to transmit the idle time.)

### 24.8.3 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 24-5.

**Figure 24-5. Double-Buffered WUT and TXSHF**




#### 24.8.3.1 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.

### 24.8.4 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.

## 24.9 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 24-6).

### 24.9.1 Sending an Address

The TXWAKE bit value is placed in the address bit. During transmission, when the SCITXBUF register and TXWAKE are loaded into the TXSHF register and WUT respectively, TXWAKE is reset to 0 and WUT becomes the value of the address bit of the current frame. Thus, to send an address:

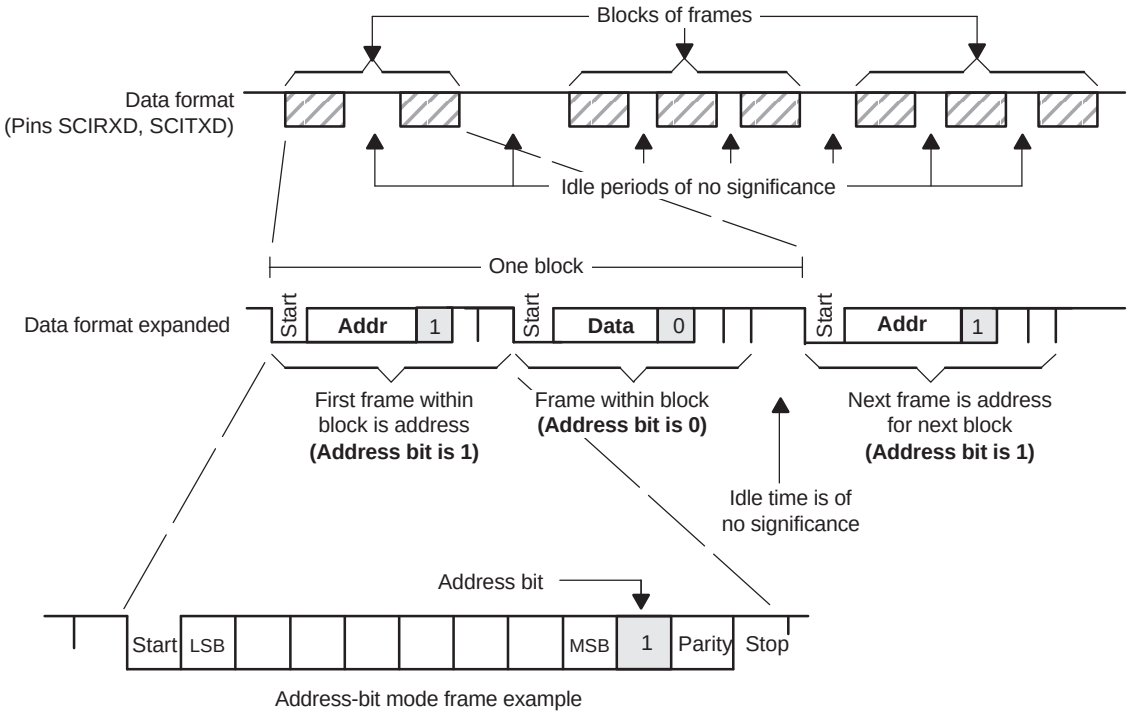
1. Set the TXWAKE bit to 1 and write the appropriate address value to the SCITXBUF register.

When this address value is transferred to the TXSHF register and shifted out, its address bit is sent as a 1. This flags the other processors on the serial link to read the address.

2. Write to SCITXBUF and TXWAKE after TXSHF and WUT are loaded. (Can be written to immediately since both TXSHF and WUT are both double-buffered.
3. Leave the TXWAKE bit set to 0 to transmit non-address frames in the block.

**NOTE:** As a general rule, the address-bit format is typically used for data frames of 11 bytes or less. This format adds one bit value (1 for an address frame, 0 for a data frame) to all data bytes transmitted. The idle-line format is typically used for data frames of 12 bytes or more.

**Figure 24-6. Address-Bit Multiprocessor Communication Format**




## 24.10 SCI Communication Format

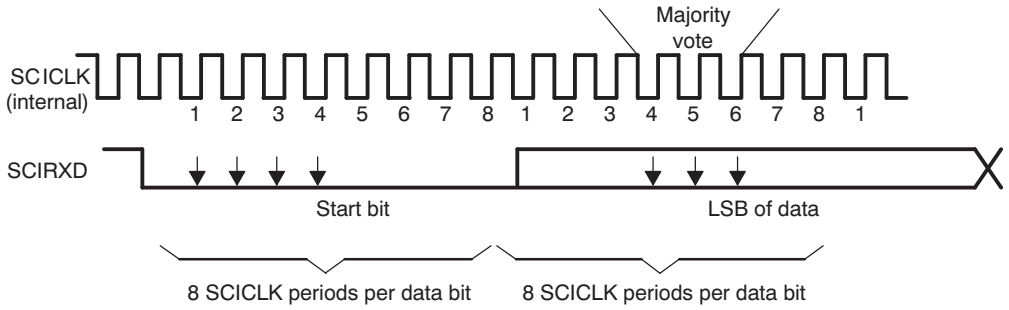
The SCI asynchronous communication format uses either single line (one way) or two line (two way) communications. In this mode, the frame consists of a start bit, one to eight data bits, an optional even/odd parity bit, and one or two stop bits (shown in Figure 24-7). There are eight SCICLK periods per 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 24-7. 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 24-7 illustrates the asynchronous communication format for this with a start bit showing where a majority vote is taken.

Since the receiver synchronizes itself to frames, the external transmitting and receiving devices do not have to use a synchronized serial clock. The clock can be generated locally.

**Figure 24-7. SCI Asynchronous Communications Format**

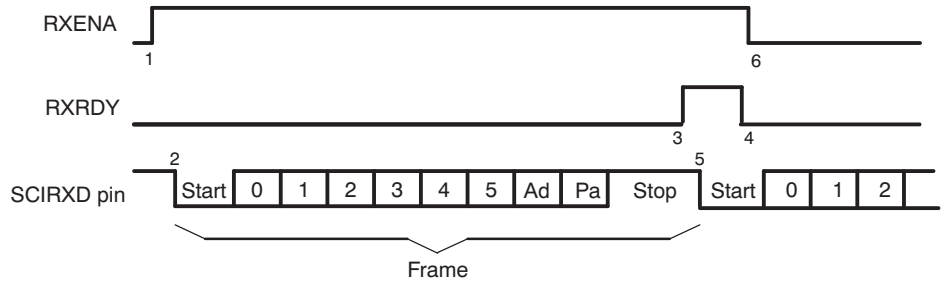



### 24.10.1 Receiver Signals in Communication Modes

Figure 24-8 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 24-8. SCI RX Signals in Communication Modes**




(1) Data arrives on the SCIRXD pin, start bit detected.

(2) Bit RXENA is brought low to disable the receiver. Data continues to be assembled in RXSHF but is not transferred to the receiver buffer register.

Notes:

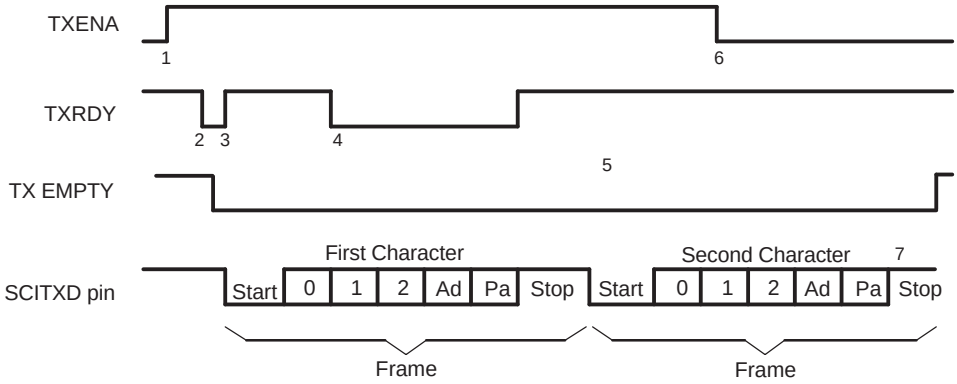
1. Flag bit RXENA (SCICTL1, bit 0) goes high to enable the receiver.
2. Data arrives on the SCIRXD pin, start bit detected.
3. Data is shifted from RXSHF to the receiver buffer register (SCIRXBUF); an interrupt is requested. Flag bit RXRDY (SCIRXST, bit 6) goes high to signal that a new character has been received.
4. The program reads SCIRXBUF; flag RXRDY is automatically cleared.
5. The next byte of data arrives on the SCIRXD pin; the start bit is detected, then cleared.
6. Bit RXENA is brought low to disable the receiver. Data continues to be assembled in RXSHF but is not transferred to the receiver buffer register.

### 24.10.2 Transmitter Signals in Communication Modes

Figure 24-9 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 24-9. 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.

## 24.11 SCI Port Interrupts

The SCI receiver and transmitter can be interrupt controlled. The SCICTL2 register has one flag bit (TXRDY) that indicates active interrupt conditions, and the SCIRXST register has two interrupt flag bits (RXRDY and BRKDT), plus the RX ERROR interrupt flag which is a logical OR of the FE, OE, BRKDT, and PE conditions. The transmitter and receiver have separate interrupt-enable bits. When not enabled, the interrupts are not asserted; however, the condition flags remain active, reflecting transmission and receipt status.

The SCI has independent peripheral interrupt vectors for the receiver and transmitter. Peripheral interrupt requests can be either high priority or low priority. This is indicated by the priority bits which are output from the peripheral to the PIE controller. When both RX and TX interrupt requests are made at the same priority level, the receiver always has higher priority than the transmitter, reducing the possibility of receiver overrun.

The operation of peripheral interrupts is described in the peripheral interrupt expansion controller section of the *External Peripheral Interface (ePIE)* chapter.

- If the RX/BK INT ENA bit (SCICTL2, bit 1) is set, the receiver peripheral interrupt request is asserted when one of the following events occurs:
  - The SCI receives a complete frame and transfers the data in the RXSHF register to the SCIRXBUF register. This action sets the RXRDY flag (SCIRXST, bit 6) and initiates an interrupt.
  - A break detect condition occurs (the SCIRXD is low for ten bit periods following a missing stop bit). This action sets the BRKDT flag bit (SCIRXST, bit 5) and initiates an interrupt.
- If the TX INT ENA bit (SCICTL2.0) is set, the transmitter peripheral interrupt request is asserted whenever the data in the SCITXBUF register is transferred to the TXSHF register, indicating that the CPU can write to SCITXBUF; this action sets the TXRDY flag bit (SCICTL2, bit 7) and initiates an interrupt.

**NOTE:** Interrupt generation due to the RXRDY and BRKDT bits is controlled by the RX/BK INT ENA bit (SCICTL2, bit 1). Interrupt generation due to the RX ERROR bit is controlled by the RX ERR INT ENA bit (SCICTL1, bit 6).

## 24.12 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 24-3 shows the baud-select values for common SCI bit rates.

**Table 24-3. Asynchronous Baud Register Values for Common SCI Bit Rates**

| | | LSPCLK Clock Frequency, 100 MHz | |
|---|---|---|---|
| **Ideal Baud** | **BRR** | **Actual Baud** | **% Error** |
| 2400 | 5207 (1457h) | 2400 | 0 |
| 4800 | 2603 (A2Bh) | 4800 | 0 |
| 9600 | 1301 (515h) | 9601 | 0.01 |
| 19200 | 650 (28Ah) | 19201 | 0.01 |
| 38400 | 324 (144h) | 38462 | 0.16 |

LSPCLK/16 is the maximum baud rate. For example, if LSPCLK is 100MHz, then the maximum baud rate is 6.25Mbps.

## 24.13 SCI Enhanced Features

The 28x SCI features autobaud detection and transmit/receive FIFO. The following section explains the FIFO operation.

### *24.13.1 SCI FIFO Description*

The following steps explain the FIFO features and help with programming the SCI with FIFOs.

1. *Reset.* At reset the SCI powers up in standard SCI mode and the FIFO function is disabled. The FIFO registers SCIFFTX, SCIFFRX, and SCIFFCT remain inactive.
2. *Standard SCI.* The standard F24x SCI modes will work normally with TXINT/RXINT interrupts as the interrupt source for the module.
3. *FIFO enable.* FIFO mode is enabled by setting the SCIFFEN bit in the SCIFFTX register. SCIRST can reset the FIFO mode at any stage of its operation.
4. Active registers. All the SCI registers and SCI FIFO registers (SCIFFTX, SCIFFRX, and SCIFFCT) are active.
5. *Interrupts.* FIFO mode has two interrupts; one for transmit FIFO, TXINT and one for receive FIFO, RXINT. RXINT is the common interrupt for SCI FIFO receive, receive error, and receive FIFO overflow conditions. The TXINT of the standard SCI will be disabled and this interrupt will service as SCI transmit FIFO interrupt.
6. *Buffers.* Transmit and receive buffers are supplemented with two 16-level FIFOs. The transmit FIFO registers are 8 bits wide and receive FIFO registers are 10 bits wide. The one-word transmit buffer of the standard SCI functions as a transition buffer between the transmit FIFO and shift register. The one-word transmit buffer is loaded from the transmit FIFO only after the last bit of the shift register is shifted out. With the FIFO enabled, TXSHF is directly loaded after an optional delay value (SCIFFCT), TXBUF is not used. When FIFO mode is enabled for SCI, characters written to SCITXBUF are queued in to SCI-TXFIFO and the characters received in SCI-RXFIFO can be read using SCIRXBUF.
7. *Delayed transfer.* The rate at which words in the FIFO are transferred to the transmit shift register is programmable. The SCIFFCT register bits (7−0) FFTXDLY7−FFTXDLY0 define the delay between the word transfer. The delay is defined in the number SCI baud clock cycles. The 8 bit register can define

a minimum delay of 0 baud clock cycles and a maximum of 256-baud clock cycles. With zero delay, the SCI module can transmit data in continuous mode with the FIFO words shifting out back to back. With the 256 clock delay the SCI module can transmit data in a maximum delayed mode with the FIFO words shifting out with a delay of 256 baud clocks between each words. The programmable delay facilitates communication with slow SCI/UARTs with little CPU intervention.

8. *FIFO status bits.* Both the transmit and receive FIFOs have status bits TXFFST or RXFFST (bits 12−8) that define the number of words available in the FIFOs at any time. The transmit FIFO reset bit TXFIFO and receive reset bit RXFIFO reset the FIFO pointers to zero when these bits are cleared to 0. The FIFOs resumes operation from start once these bits are set to one.
9. *Programmable interrupt levels.* Both transmit and receive FIFO can generate CPU interrupts. The interrupt trigger is generated whenever the transmit FIFO status bits TXFFST (bits 12−8) match (less than or equal to) the interrupt trigger level bits TXFFIL (bits 4−0 ). This provides a programmable interrupt trigger for transmit and receive sections of the SCI. Default value for these trigger level bits will be 0x11111 for receive FIFO and 0x00000 for transmit FIFO, respectively.

Figure 24-10 and Table 24-4 explain the operation/configuration of SCI interrupts in nonFIFO/FFO mode.

**Figure 24-10. SCI FIFO Interrupt Flags and Enable Logic**

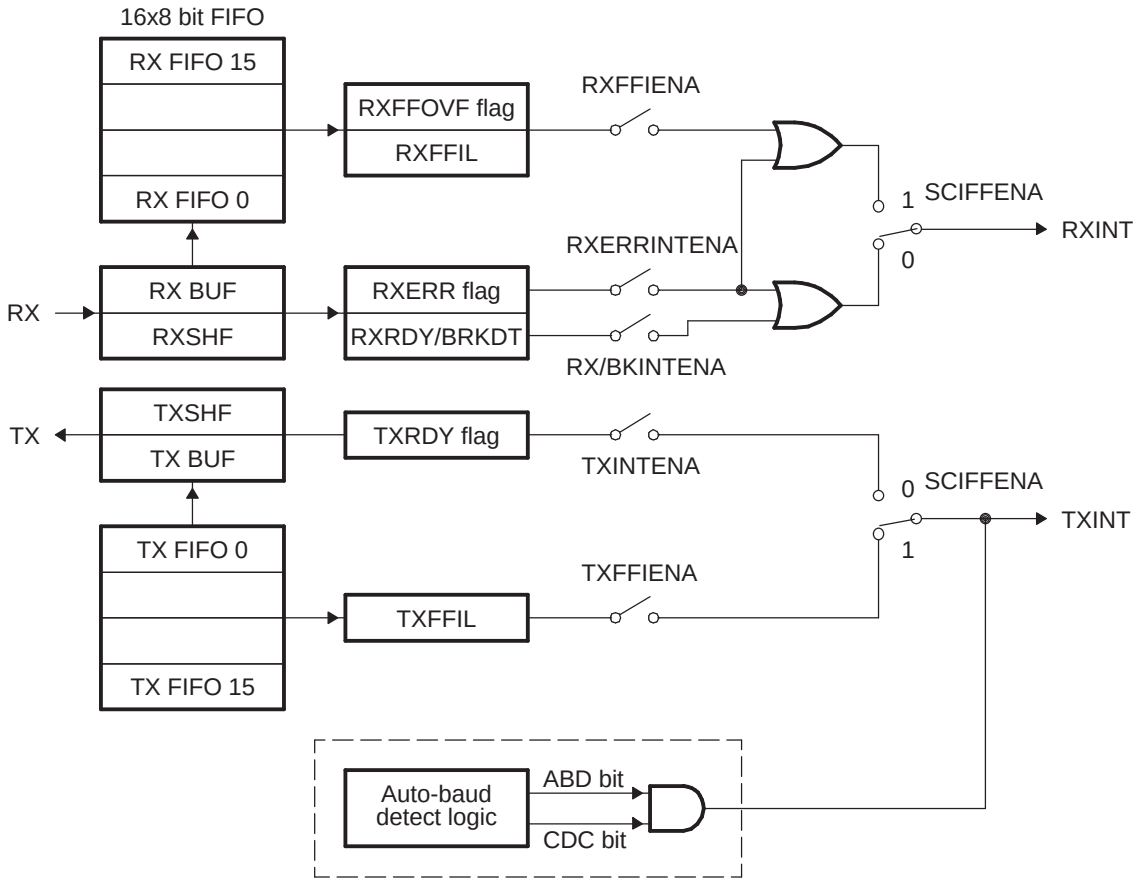


**Table 24-4. SCI Interrupt Flags**

| 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

### 24.13.2 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.

### 24.13.3 Autobaud-Detect Sequence

Bits ABD and CDC in SCIFFCT control the autobaud logic. The SCIRST bit should be enabled to make autobaud logic work.

If ABD is set while CDC is 1, which indicates auto-baud alignment, SCI transmit FIFO interrupt will occur (TXINT). After the interrupt service CDC bit has to be cleared by software. If CDC remains set even after interrupt service, there should be no repeat interrupts.

1. Enable autobaud-detect mode for the SCI by setting the CDC bit (bit 13) in SCIFFCT and clearing the ABD bit (Bit 15) by writing a 1 to ABDCLR bit (bit 14).
2. Initialize the baud register to be 1 or less than a baud rate limit of 500 Kbps.
3. Allow SCI to receive either character "A" or "a" from a host at the desired baud rate. If the first character is either "A" or "a", the autobaud- detect hardware will detect the incoming baud rate and set the ABD bit.
4. The auto-detect hardware will update the baud rate register with the equivalent baud value hex. The logic will also generate an interrupt to the CPU.
5. Respond to the interrupt clear ADB bit by writing a 1 to ABD CLR (bit 14) of SCIFFCT register and disable further autobaud locking by clearing CDC bit by writing a 0.
6. Read the receive buffer for character "A" or "a" to empty the buffer and buffer status.
7. If ABD is set while CDC is 1, which indicates autobaud alignment, the SCI transmit FIFO interrupt will occur (TXINT). After the interrupt service CDC bit must be cleared by software.

**NOTE:** At higher baud rates, the slew rate of the incoming data bits can be affected by transceiver and connector performance. While normal serial communications may work well, this slew rate may limit reliable autobaud detection at higher baud rates (typically beyond 100k baud) and cause the auto-baudlock feature to fail.

To avoid this, the following is recommended:

- Achieve a baud-lock between the host and 28x SCI boot loader using a lower baud rate.
- The host may then handshake with the loaded 28x application to set the SCI baud rate register to the desired higher baud rate.

## 24.14 Registers

### 24.14.1 Serial Communications Interface Base Addresses

**Table 24-5. 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 |

### 24.14.1.1 SCI_REGS Registers

Table 24-6 lists the memory-mapped registers for the SCI_REGS. 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. 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 24-7 shows the codes that are used for access types in this section.

**Table 24-7. SCI_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| W=1 | 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. |

#### 24.14.1.1.1 SCICCR Register (Offset = 0h) [reset = 0h]

SCICCR is shown in Figure 24-11 and described in Table 24-8.

Return to Summary Table.

Communications control register

**Figure 24-11. 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 24-8. 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 |

**Table 24-8. 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<br>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_LEGNTH_1<br>1h (R/W) = SCICHAR_LEGNTH_2<br>2h (R/W) = SCICHAR_LEGNTH_3<br>3h (R/W) = SCICHAR_LEGNTH_4<br>4h (R/W) = SCICHAR_LEGNTH_5<br>5h (R/W) = SCICHAR_LEGNTH_6<br>6h (R/W) = SCICHAR_LEGNTH_7<br>7h (R/W) = SCICHAR_LEGNTH_8 |

#### 24.14.1.1.2 SCICTL1 Register (Offset = 1h) [reset = 0h]

SCICTL1 is shown in Figure 24-12 and described in Table 24-9.

Return to Summary Table.

Control register 1

**Figure 24-12. 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 24-9. 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. The SW RESET bit does not affect any of the configuration bits.<br>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.<br>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. |
| 4 | RESERVED | R | 0h | Reserved |

**Table 24-9. 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. The receiver still operates when the SLEEP bit is set however, operation does not update the receiver buffer ready bit (SCIRXST, bit 6, RXRDY) or the error status bits (SCIRXST, bit 5-2: BRKDT, FE, OE, and PE) unless the address byte is detected. SLEEP is not cleared when the address byte is detected.<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, the receiver shift register can continue to assemble characters. Thus, if RXENA is set during the reception of a character, the complete character will be transferred into the receiver buffer registers, SCIRXEMU and SCIRXBUF.<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 |

#### 24.14.1.1.3 SCIHBAUD Register (Offset = 2h) [reset = 0h]

SCIHBAUD is shown in Figure 24-13 and described in Table 24-10.

Return to Summary Table.

Baud rate (high) register

**Figure 24-13. 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 24-10. 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>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 |

#### 24.14.1.1.4 SCILBAUD Register (Offset = 3h) [reset = 0h]

SCILBAUD is shown in Figure 24-14 and described in Table 24-11.

Return to Summary Table.

Baud rate (low) register

**Figure 24-14. 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 24-11. 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 |

#### 24.14.1.1.5 SCICTL2 Register (Offset = 4h) [reset = C0h]

SCICTL2 is shown in Figure 24-15 and described in Table 24-12.

Return to Summary Table.

Control register 2

**Figure 24-15. 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 24-12. 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 |

**Table 24-12. 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 |

#### 24.14.1.1.6 SCIRXST Register (Offset = 5h) [reset = 0h]

SCIRXST is shown in Figure 24-16 and described in Table 24-13.

Return to Summary Table.

Receive status register

**Figure 24-16. 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 24-13. 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 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, 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 |
| 5 | BRKDT | R | 0h | SCI break-detect flag.<br>The SCI sets this bit when a break condition occurs. A break conditionoccurs when the SCI receiver data line (SCIRXD) remains continuously low for at least ten bits,<br>beginning after a missing first stop bit. The occurrence of a break causes a receiver interrupt to be generated if the RX/BK INT ENA bit is a 1, but it does not cause the receiver buffer to be loaded. A BRKDT interrupt can occur even if the receiver SLEEP bit is set to 1. BRKDT is cleared by an active SW RESET or by a system reset. It is not cleared by receipt of a character after the break is detected. In order to receive more characters, the SCI must be reset by toggling the SW RESET bit or by a system reset.<br>Reset type: SYSRSn<br>0h (R/W) = No break condition<br>1h (R/W) = Break condition occurred |

**Table 24-13. SCIRXST Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 or by a system reset.<br>Reset type: SYSRSn<br>0h (R/W) = No framing error detected<br>1h (R/W) = Framing error detected |
| 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 or by 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 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>- A system reset |
| 0 | RESERVED | R | 0h | Reserved |

#### 24.14.1.1.7 SCIRXEMU Register (Offset = 6h) [reset = 0h]

SCIRXEMU is shown in Figure 24-17 and described in Table 24-14.

Return to Summary Table.

Receive emulation buffer register

**Figure 24-17. 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 24-14. 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 |

#### 24.14.1.1.8 SCIRXBUF Register (Offset = 7h) [reset = 0h]

SCIRXBUF is shown in Figure 24-18 and described in Table 24-15.

Return to Summary Table.

Receive data buffer

**Figure 24-18. 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 24-15. 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>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>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 |
| 7-0 | SAR | R | 0h | Receive Character bits<br>Reset type: SYSRSn |

#### 24.14.1.1.9 SCITXBUF Register (Offset = 9h) [reset = 0h]

SCITXBUF is shown in Figure 24-19 and described in Table 24-16.

Return to Summary Table.

Transmit data buffer

**Figure 24-19. 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 24-16. 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 |

#### 24.14.1.1.10 SCIFFTX Register (Offset = Ah) [reset = A000h]

SCIFFTX is shown in Figure 24-20 and described in Table 24-17.

Return to Summary Table.

FIFO transmit register

**Figure 24-20. 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/W=1-0h | R/W-0h | R/W-0h | | | | |

**Table 24-17. SCIFFTX Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | SCIRST | R/W | 1h | SCI Reset<br>0 Write 0 to reset the SCI transmit and receive channels. SCI FIFO register configuration bits will be left as is.<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 |

**Table 24-17. SCIFFTX Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | TXFFINTCLR | R=0/W=1 | 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 based on TXFFIL match (less than or equal to) is disabled<br>1h (R/W) = TX FIFO interrupt based on TXFFIL match (less than or equal to) is enabled. |
| 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 |

#### 24.14.1.1.11 SCIFFRX Register (Offset = Bh) [reset = 201Fh]

SCIFFRX is shown in Figure 24-21 and described in Table 24-18.

Return to Summary Table.

FIFO Receive register

**Figure 24-21. SCIFFRX Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RXFFOVF | RXFFOVRCLR | RXFIFORESET | RXFFST | | | | |
| R-0h | R=0/W=1-0h | 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 24-18. 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>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/W=1 | 0h | RXFFOVF clear<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 |
| 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 |

**Table 24-18. SCIFFRX Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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>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 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 |
| 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 |

#### 24.14.1.1.12 SCIFFCT Register (Offset = Ch) [reset = 0h]

SCIFFCT is shown in Figure 24-22 and described in Table 24-19.

Return to Summary Table.

FIFO control register

**Figure 24-22. 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 24-19. 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<br>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 |

#### 24.14.1.1.13 SCIPRI Register (Offset = Fh) [reset = 0h]

SCIPRI is shown in Figure 24-23 and described in Table 24-20.

Return to Summary Table.

SCI Priority control

**Figure 24-23. 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 24-20. 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 |

# Inter-Integrated Circuit Module (I2C)

This chapter describes the features and operation of the inter-integrated circuit (I2C) module. The I2C module provides an interface between one of these devices and devices compliant with the NXP Semiconductors Inter-IC bus (I2C bus) specification version 2.1 and connected by way of an I2C bus. External components attached to this 2-wire serial bus can transmit/receive 1 to 8-bit data to/from the device through the I2C module. This guide assumes the reader is familiar with the I2C bus specification.

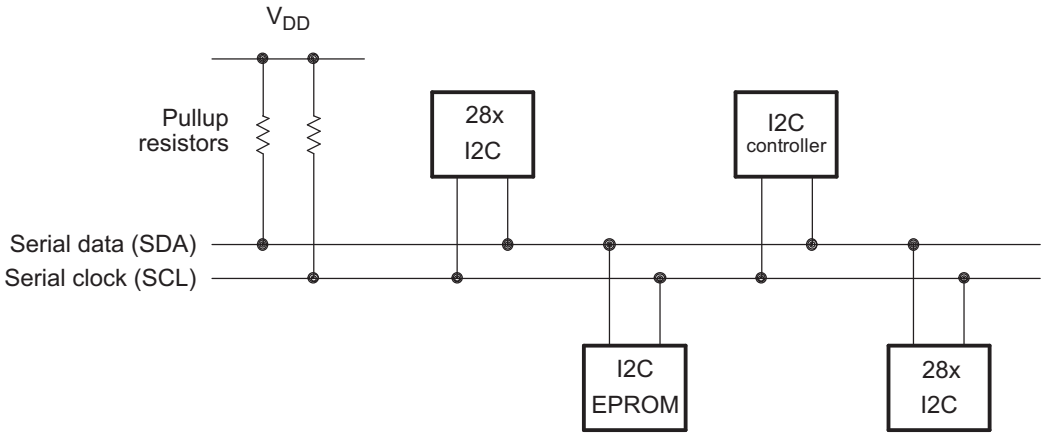
**NOTE:** A unit of data transmitted or received by the I2C module can have fewer than 8 bits; however, for convenience, a unit of data is called a data byte throughout this document. The number of bits in a data byte is selectable via the BC bits of the mode register, I2CMDR.

**Topic** **Page**

## 25.1 Introduction to the I2C Module

The I2C module supports any slave or master I2C-compatible device. Figure 25-1 shows an example of multiple I2C modules connected for a two-way transfer from one device to other devices.

**Figure 25-1. Multiple I2C Modules Connected**




### 25.1.1 Features

The I2C module has the following features:

- Compliance with the NXP Semiconductors I2C bus specification (version 2.1):
  - Support for 8-bit format transfers
  - 7-bit and 10-bit addressing modes
  - General call
  - START byte mode
  - Support for multiple master-transmitters and slave-receivers
  - Support for multiple slave-transmitters and master-receivers
  - Combined master transmit/receive and receive/transmit mode
  - Data transfer rate from 10 kbps up to 400 kbps (Fast-mode)
- One 16-byte receive FIFO and one 16-byte transmit FIFO
- Module enable/disable capability
- Free data format mode

### 25.1.2 Features Not Supported

The I2C module does not support:

- High-speed mode (Hs-mode)
- CBUS-compatibility mode

### 25.1.3 Functional Overview

Each device connected to an I2C bus is recognized by a unique address. Each device can operate as either a transmitter or a receiver, depending on the function of the device. A device connected to the I2C bus can also be considered as the master or the slave when performing data transfers. A master device is the device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. During this transfer, any device addressed by this master is considered a slave. The I2C module supports the multi-master mode, in which one or more devices capable of controlling an I2C bus can be connected to the same I2C bus.

For data communication, the I2C module has a serial data pin (SDA) and a serial clock pin (SCL), as shown in Section 25.6. These two pins carry information between the 28x device and other devices connected to the I2C bus. The SDA and SCL pins both are bidirectional. They each must be connected to a positive supply voltage using a pull-up resistor. When the bus is free, both pins are high. The driver of these two pins has an open-drain configuration to perform the required wired-AND function.

There are two major transfer techniques: .

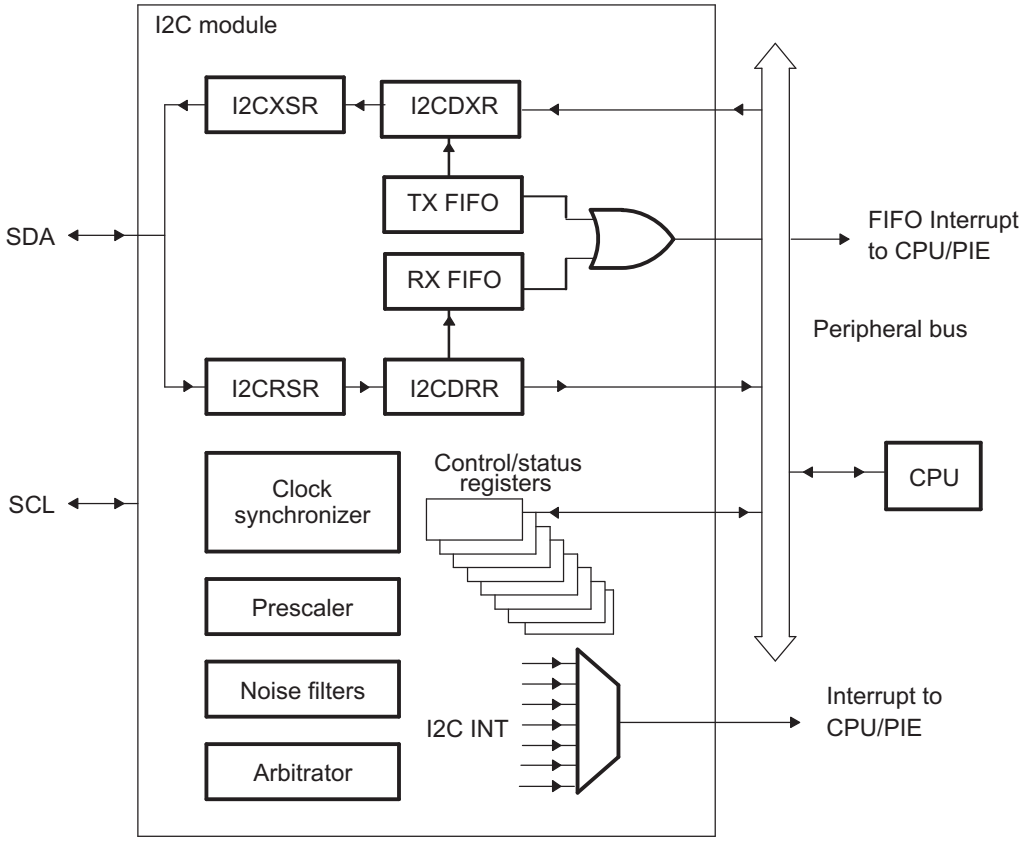
- Standard Mode: Send exactly n data values, where n is a value you program in an I2C module register. See the *Register* section for more information.
- Repeat Mode: Keep sending data values until you use software to initiate a STOP condition or a new START condition. See *Registers* for RM bit information.

The I2C module consists of the following primary blocks:

- A serial interface: one data pin (SDA) and one clock pin (SCL)
- Data registers and FIFOs to temporarily hold receive data and transmit data traveling between the SDA pin and the CPU
- Control and status registers
- A peripheral bus interface to enable the CPU to access the I2C module registers and FIFOs.
- A clock synchronizer to synchronize the I2C input clock (from the device clock generator) and the clock on the SCL pin, and to synchronize data transfers with masters of different clock speeds
- A prescaler to divide down the input clock that is driven to the I2C module
- A noise filter on each of the two pins, SDA and SCL
- An arbitrator to handle arbitration between the I2C module (when it is a master) and another master
- Interrupt generation logic, so that an interrupt can be sent to the CPU
- FIFO interrupt generation logic, so that FIFO access can be synchronized to data reception and data transmission in the I2C module

Figure 25-2 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 a time. When the I2C module is configured as a receiver, received data is shifted into I2CRSR and then copied to I2CDRR.

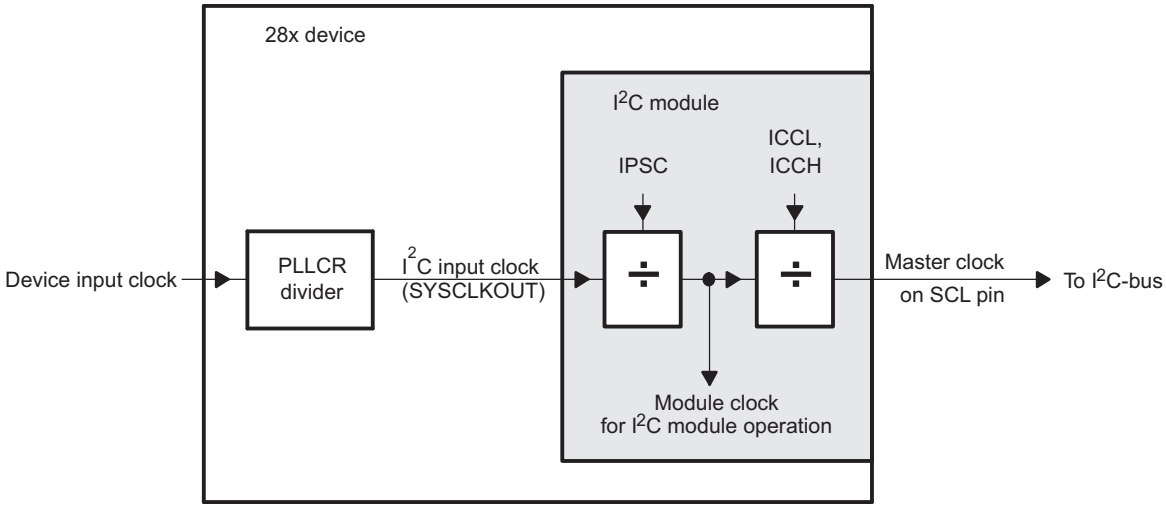
**Figure 25-2. I2C Module Conceptual Block Diagram**




### 25.1.4 Clock Generation

As shown in Figure 25-3, the device clock generator receives a signal from an external clock source and produces an I2C input clock with a programmed frequency. The I2C input clock is equivalent to the CPU clock and is then divided twice more inside the I2C module to produce the module clock and the master clock.

**Figure 25-3. Clocking Diagram for the I2C Module**



The module clock determines the frequency at which the I2C module operates. A programmable prescaler in the I2C module divides down the I2C input clock to produce the module clock. To specify the divide-down value, initialize the IPSC field of the prescaler register, I2CPSC. The resulting frequency is:

$$\text{module clock frequency} = \frac{\text{I2C input clock frequency}}{(\text{IPSC} + 1)}$$

**NOTE:** To meet all of the I2C protocol timing specifications, the module clock must be between 7 - 12 MHz.

The prescaler must be initialized only while the I2C module is in the reset state (IRS = 0 in I2CMDR). The prescaled frequency takes effect only when IRS is changed to 1. Changing the IPSC value while IRS = 1 has no effect.

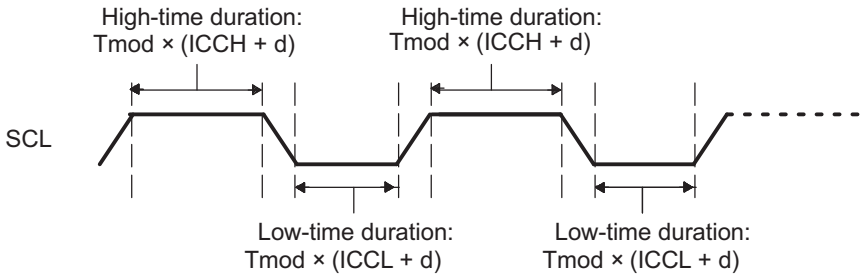
The master clock appears on the SCL pin when the I2C module is configured to be a master on the I2C bus. This clock controls the timing of communication between the I2C module and a slave. As shown in Figure 25-3, a second clock divider in the I2C module divides down the module clock to produce the master 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 25.1.5 for the master clock frequency equation.

### 25.1.5 I2C Clock Divider Registers (I2CCLKL and I2CCLKH)

As explained in Section 25.1.4, when the I2C module is a master, the module clock is divided down for use as the master clock on the SCL pin. As shown in Figure 25-4, 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 25-4. The Roles of the Clock Divide-Down Values (ICCL and ICCH)**




#### 25.1.5.1 Formula for the Master Clock Period

The period of the master clock (Tmst) is a multiple of the period of the module clock (Tmod):

$$T_{mst} = T_{mod} \times [(\text{ICCL} + d) + (\text{ICCH} + d)]$$

$$T_{mst} = \frac{(\text{IPSC} + 1)\ [(\text{ICCL} + d) + (\text{ICCH} + d)]}{\text{I2C input clock frequency}}$$

where d depends on the divide-down value IPSC, as shown in Table 25-1. IPSC is described in the I2CPSC register.

**Table 25-1. Dependency of Delay d on the Divide-Down Value IPSC**

| IPSC | d |
|---|---|
| 0 | 7 |
| 1 | 6 |

**Table 25-1. Dependency of Delay d on the Divide-Down Value IPSC (continued)**

| IPSC | d |
|---|---|
| Greater than 1 | 5 |

## 25.2 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

Some IO functionality is defined by GPIO register settings independent of this peripheral. For input signals, the GPIO input qualification should be set to asynchronous mode by setting the appropriate GPxQSELn register bits to 11b. The internal pullups can be configured in the GPyPUD register.

See the *GPIO* chapter for more details on GPIO mux and settings.

## 25.3 I2C Module Operational Details

This section provides an overview of the I2C bus protocol and how it is implemented.

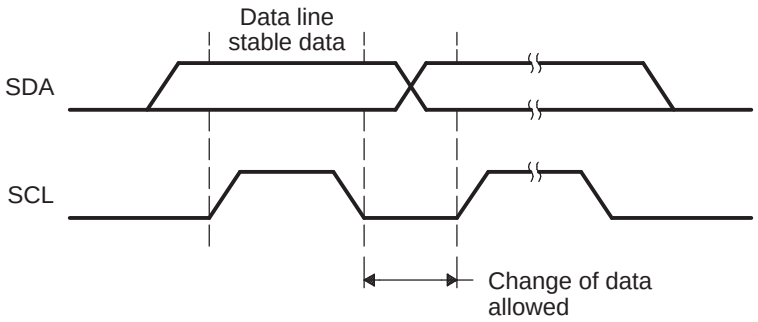
### *25.3.1 Input and Output Voltage Levels*

One clock pulse is generated by the master device for each data bit transferred. Due to a variety of different technology devices that can be connected to the I2C bus, the levels of logic 0 (low) and logic 1 (high) are not fixed and depend on the associated level of $V_{DD}$. For details, see the data manual for your particular device.

### *25.3.2 Data Validity*

The data on SDA must be stable during the high period of the clock (see Figure 25-5). The high or low state of the data line, SDA, should change only when the clock signal on SCL is low.

**Figure 25-5. Bit Transfer on the I2C bus**




### *25.3.3 Operating Modes*

The I2C module has four basic operating modes to support data transfers as a master and as a slave. See Table 25-2 for the names and descriptions of the modes.

If the I2C module is a master, it begins as a master-transmitter and typically transmits an address for a particular slave. When giving data to the slave, the I2C module must remain a master-transmitter. To receive data from a slave, the I2C module must be changed to the master-receiver mode.

If the I2C module is a slave, it begins as a slave-receiver and typically sends acknowledgment when it recognizes its slave address from a master. If the master will be sending data to the I2C module, the module must remain a slave-receiver. If the master has requested data from the I2C module, the module must be changed to the slave-transmitter mode.

**Table 25-2. Operating Modes of the I2C Module**

| Operating Mode | Description |
|---|---|
| Slave-receiver modes | The I2C module is a slave and receives data from a master. |
| | All slaves begin in this mode. In this mode, serial data bits received on SDA are shifted in with the clock pulses that are generated by the master. As a slave, the I2C module does not generate the clock signal, but it can hold SCL low while the intervention of the device is required (RSFULL = 1 in I2CSTR) after a byte has been received. See section Section 26.4 for more details. |
| Slave-transmitter mode | The I2C module is a slave and transmits data to a master. |
| | This mode can be entered only from the slave-receiver mode; the I2C module must first receive a command from the master. When you are using any of the 7-bit/10-bit addressing formats, the I2C module enters its slave-transmitter mode if the slave address byte is the same as its own address (in I2COAR) and the master has transmitted R/$\overline{W}$ = 1. As a slave-transmitter, the I2C module then shifts the serial data out on SDA with the clock pulses that are generated by the master. While a slave, the I2C module does not generate the clock signal, but it can hold SCL low while the intervention of the device is required (XSMT = 0 in I2CSTR) after a byte has been transmitted. See section Section 26.4 for more details. |
| Master-receiver mode | The I2C module is a master and receives data from a slave. |
| | This mode can be entered only from the master-transmitter mode; the I2C module must first transmit a command to the slave. When you are using any of the 7-bit/10-bit addressing formats, the I2C module enters its master-receiver mode after transmitting the slave address byte and R/$\overline{W}$ = 1. Serial data bits on SDA are shifted into the I2C module with the clock pulses generated by the I2C module on SCL. The clock pulses are inhibited and SCL is held low when the intervention of the device is required (RSFULL = 1 in I2CSTR) after a byte has been received. |
| Master-transmitter modes | The IC module is a master and transmits control information and data to a slave. |
| | All masters begin in this mode. In this mode, data assembled in any of the 7-bit/10-bit addressing formats is shifted out on SDA. The bit shifting is synchronized with the clock pulses generated by the I2C module on SCL. The clock pulses are inhibited and SCL is held low when the intervention of the device is required (XSMT = 0 in I2CSTR) after a byte has been transmitted. |

To summarize, SCL will be held low in the following conditions:

- When RSFULL = 1, in Slave-receiver mode.
- When XSMT = 0, in Slave-transmitter mode.

I2C slave nodes have to accept and provide data when the I2C master node requests it.

- To release SCL in slave-receiver mode, read data from I2CDRR.
- To release SCL in slave-transmitter mode, write data to I2CDXR.
- To force a release without handling the data, reset the module using the I2CMDR.IRS bit.

**Table 25-3. 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) |

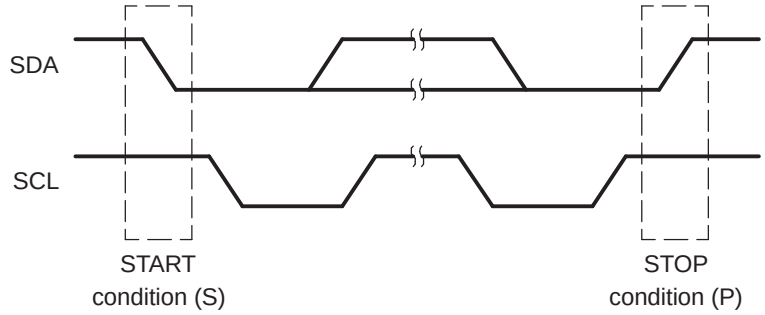
(1) S = START condition; A = Address; D = Data byte; P = STOP condition;

### 25.3.4 I2C Module START and STOP Conditions

START and STOP conditions can be generated by the I2C module when the module is configured to be a master on the I2C bus. As shown in Figure 25-6:

- The START condition is defined as a high-to-low transition on the SDA line while SCL is high. A master drives this condition to indicate the start of a data transfer.
- The STOP condition is defined as a low-to-high transition on the SDA line while SCL is high. A master drives this condition to indicate the end of a data transfer.

**Figure 25-6. I2C Module START and STOP Conditions**




After a START condition and before a subsequent STOP condition, the I2C bus is considered busy, and the bus busy (BB) bit of I2CSTR is 1. Between a STOP condition and the next START condition, the bus is considered free, and BB is 0.

For the I2C module to start a data transfer with a START condition, the master mode bit (MST) and the START condition bit (STT) in I2CMDR must both be 1. For the I2C module to end a data transfer with a STOP condition, the STOP condition bit (STP) must be set to 1. When the BB bit is set to 1 and the STT bit is set to 1, a repeated START condition is generated. For a description of I2CMDR and its bits (including MST, STT, and STP), see *Registers*.

### 25.3.5 Serial Data Formats

Figure 25-7 shows an example of a data transfer on the I2C bus. The I2C module supports 1 to 8-bit data values. In Figure 25-7, 8-bit data is transferred. Each bit put on the SDA line equates to 1 pulse on the SCL line, and the values are always transferred with the most significant bit (MSB) first. The number of data values that can be transmitted or received is unrestricted. The serial data format used in Figure 25-7 is the 7-bit addressing format. The I2C module supports the formats shown in Figure 25-8 through Figure 25-10 and described in the paragraphs that follow the figures.

**NOTE:** In Figure 25-7 through Figure 25-10, n = the number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

**Figure 25-7. I2C Module Data Transfer (7-Bit Addressing with 8-bit Data Configuration Shown)**

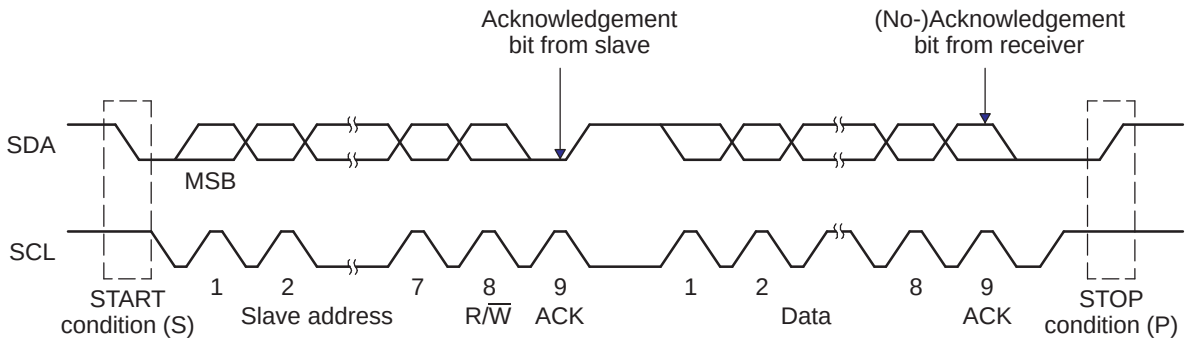


**Figure 25-8. I2C Module 7-Bit Addressing Format (FDF = 0, XA = 0 in I2CMDR)**

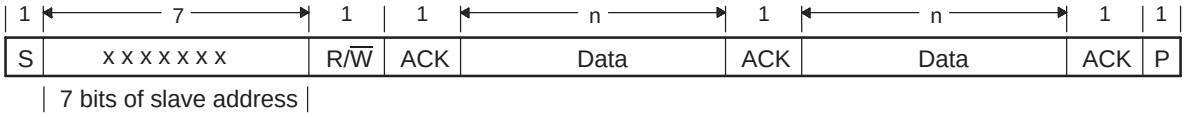



**Figure 25-9. I2C Module 10-Bit Addressing Format (FDF = 0, XA = 1 in I2CMDR)**

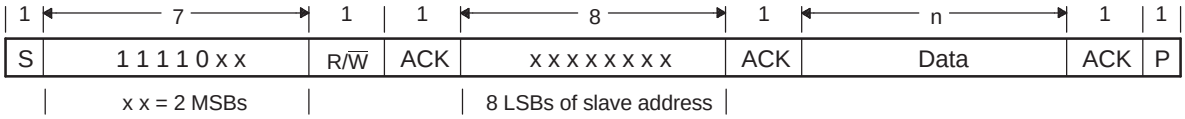
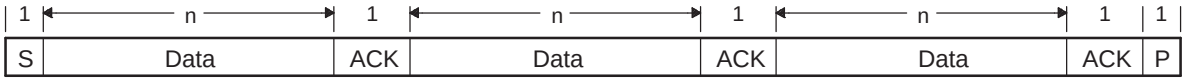



**Figure 25-10. I2C Module Free Data Format (FDF = 1 in I2CMDR)**




### 25.3.5.1 7-Bit Addressing Format

In the 7-bit addressing format (see Figure 25-8), 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 master writes (transmits) data to the addressed slave.
- R/$\overline{W}$ = 1: The master reads (receives) data from the slave.

An extra clock cycle dedicated for acknowledgment (ACK) is inserted after each byte. If the ACK bit is inserted by the slave after the first byte from the master, it is followed by n bits of data from the transmitter (master or slave, depending on the R/$\overline{W}$ 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.

To select the 7-bit addressing format, write 0 to the expanded address enable (XA) bit of I2CMDR, and make sure the free data format mode is off (FDF = 0 in I2CMDR).

### 25.3.5.2 10-Bit Addressing Format

The 10-bit addressing format (see Figure 25-9) is similar to the 7-bit addressing format, but the master sends the slave address in two separate byte transfers. The first byte consists of 11110b, the two MSBs of the 10-bit slave address, and R/$\overline{W}$ = 0 (write). The second byte is the remaining 8 bits of the 10-bit slave address. The slave must send acknowledgment after each of the two byte transfers. Once the master has written the second byte to the slave, the master can either write data or use a repeated START condition to change the data direction. For more details about using 10-bit addressing, see the NXP Semiconductors I2C bus specification.

To select the 10-bit addressing format, write 1 to the XA bit of I2CMDR and make sure the free data format mode is off (FDF = 0 in I2CMDR).

### 25.3.5.3 Free Data Format

In this format (see Figure 25-10), the first byte after a START condition (S) is a data byte. An ACK bit is inserted after each data byte, which can be from 1 to 8 bits, depending on the BC field of I2CMDR. No address or data-direction bit is sent. Therefore, the transmitter and the receiver must both support the free data format, and the direction of the data must be constant throughout the transfer.

To select the free data format, write 1 to the free data format (FDF) bit of I2CMDR. The free data format is not supported in the digital loopback mode (DLB = 1 in I2CMDR).

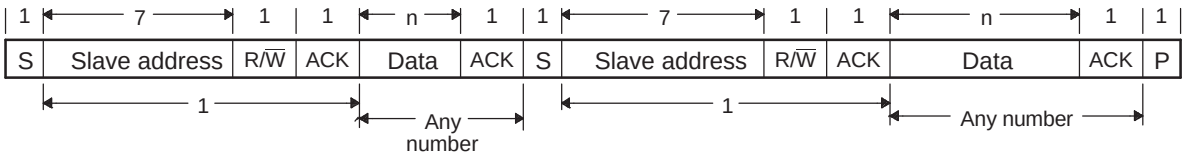
**Table 25-4. How the MST and FDF Bits of I2CMDR Affect the Role of the TRX Bit of I2CMDR**

| 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. |

#### 25.3.5.4 Using a Repeated START Condition

At the end of each data byte, the master can drive another START condition. Using this capability, a master can communicate with multiple slave addresses without having to give up control of the bus by driving a STOP condition. The length of a data byte can be from 1 to 8 bits and is selected with the BC field of I2CMDR. The repeated START condition can be used with the 7-bit addressing, 10-bit addressing, and free data formats. Figure 25-11 shows a repeated START condition in the 7-bit addressing format.

**Figure 25-11. Repeated START Condition (in This Case, 7-Bit Addressing Format)**




**NOTE:** In Figure 25-11, n = the number of data bits (from 1 to 8) specified by the bit count (BC) field of I2CMDR.

### *25.3.6 NACK Bit Generation*

When the I2C module is a receiver (master or slave), it can acknowledge or ignore bits sent by the transmitter. To ignore any new bits, the I2C module must send a no-acknowledge (NACK) bit during the acknowledge cycle on the bus. Table 25-5 summarizes the various ways you can tell the I2C module to send a NACK bit.

**Table 25-5. Ways to Generate a NACK Bit**

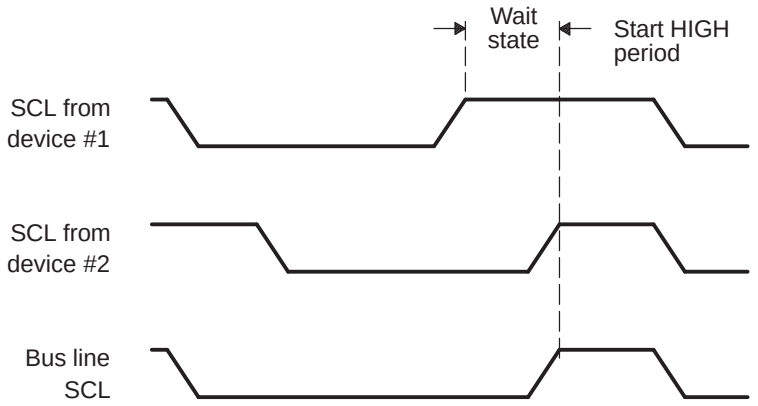
| I2C Module Condition | NACK Bit Generation Options |
|---|---|
| Slave-receiver modes | • Allow an overrun condition (RSFULL = 1 in I2CSTR)<br>• Reset the module (IRS = 0 in I2CMDR)<br>• Set the NACKMOD bit of I2CMDR before the rising edge of the last data bit you intend to receive |
| Master-receiver mode AND 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 Nonrepeat mode (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). = 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 |

### 25.3.7 Clock Synchronization

Under normal conditions, only one master device generates the clock signal, SCL. During the arbitration procedure, however, there are two or more masters and the clock must be synchronized so that the data output can be compared. Figure 25-12 illustrates the clock synchronization. The wired-AND property of SCL means that a device that first generates a low period on SCL overrules the other devices. At this high-to-low transition, the clock generators of the other devices are forced to start their own low period. The SCL is held low by the device with the longest low period. The other devices that finish their low periods must wait for SCL to be released, before starting their high periods. A synchronized signal on SCL is obtained, where the slowest device determines the length of the low period and the fastest device determines the length of the high period.

If a device pulls down the clock line for a longer time, the result is that all clock generators must enter the wait state. In this way, a slave slows down a fast master and the slow device creates enough time to store a received byte or to prepare a byte to be transmitted.

**Figure 25-12. Synchronization of Two I2C Clock Generators During Arbitration**




### 25.3.8 Arbitration

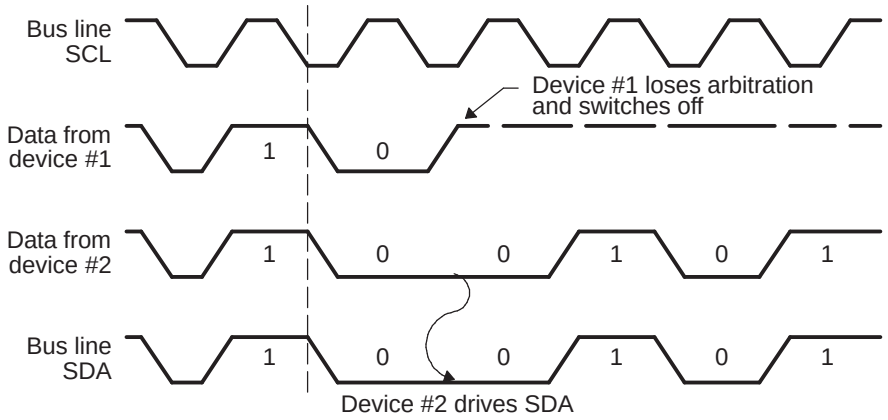
If two or more master-transmitters attempt to start a transmission on the same bus at approximately the same time, an arbitration procedure is invoked. The arbitration procedure uses the data presented on the serial data bus (SDA) by the competing transmitters. Figure 25-13 illustrates the arbitration procedure between two devices. The first master-transmitter that releases the SDA line high is overruled by another master-transmitter that drives the SDA low. The arbitration procedure gives priority to the device that transmits the serial data stream with the lowest binary value. Should two or more devices send identical first bytes, arbitration continues on the subsequent bytes.

If the I2C module is the losing master, it switches to the slave-receiver mode, sets the arbitration lost (ARBL) flag, and generates the arbitration-lost interrupt request.

If during a serial transfer the arbitration procedure is still in progress when a repeated START condition or a STOP condition is transmitted to SDA, the master-transmitters involved must send the repeated START condition or the STOP condition at the same position in the format frame. Arbitration is not allowed between:

- A repeated START condition and a data bit
- A STOP condition and a data bit
- A repeated START condition and a STOP condition

**Figure 25-13. Arbitration Procedure Between Two Master-Transmitters**



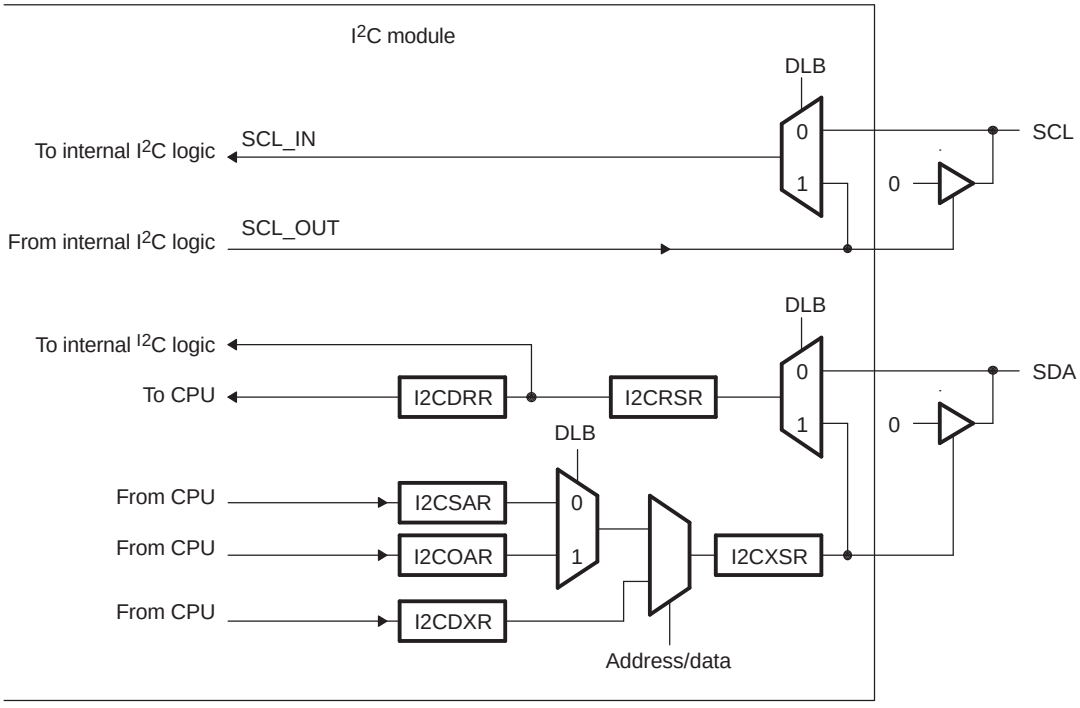

### 25.3.9 Digital Loopback Mode

The I2C module support a self-test mode called digital loopback, which is enabled by setting the DLB bit in the I2CMDR register. In this mode, data transmitted out of the I2CDXR register is received in the I2CDRR register. The data follows an internal path, and takes n cycles to reach I2CDRR, where:

n = 8 * (I2C input clock frequency) / (Module clock frequency)

The transmit clock and the receive clock are the same. The address seen on the external SDA pin is the address in the I2COAR register. Figure 25-14 shows the signal routing in digital loopback mode.

**Figure 25-14. Pin Diagram Showing the Effects of the Digital Loopback Mode (DLB) Bit**




**NOTE:** The free data format (I2CMDR.FDF = 1) is not supported in digital loopback mode.

## 25.4 Interrupt Requests Generated by the I2C Module

The I2C module can generate seven types of basic interrupt requests, which are described in Section 25.4.1. Two of these can tell the CPU when to write transmit data and when to read receive data. If you want the FIFOs to handle transmit and receive data, you can also use the FIFO interrupts described in Section 25.4.2. The basic I2C interrupts are combined to form PIE Group 8, Interrupt 1 (I2CINT1A_ISR), and the FIFO interrupts are combined to form PIE Group 8, Interrupt 2 (I2CINT2A_ISR).

### *25.4.1 Basic I2C Interrupt Requests*

The I2C module generates the interrupt requests described in Table 25-6. As shown in Figure 25-15, all requests are multiplexed through an arbiter to a single I2C interrupt request to the CPU. Each interrupt request has a flag bit in the status register (I2CSTR) and an enable bit in the interrupt enable register (I2CIER). When one of the specified events occurs, its flag bit is set. If the corresponding enable bit is 0, the interrupt request is blocked. If the enable bit is 1, the request is forwarded to the CPU as an I2C interrupt.

The I2C interrupt is one of the maskable interrupts of the CPU. As with any maskable interrupt request, if it is properly enabled in the CPU, the CPU executes the corresponding interrupt service routine (I2CINT1A_ISR). The I2CINT1A_ISR for the I2C interrupt can determine the interrupt source by reading the interrupt source register, I2CISRC. Then the I2CINT1A_ISR can branch to the appropriate subroutine.

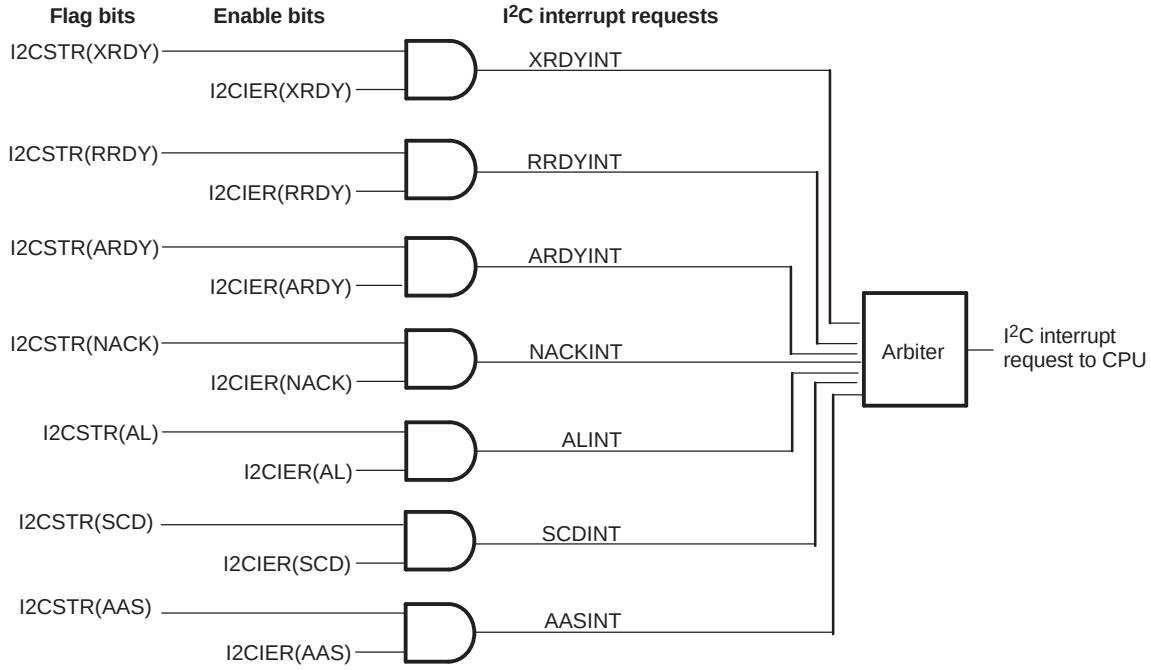
After the CPU reads I2CISRC, the following events occur:

1. The flag for the source interrupt is cleared in I2CSTR. Exception: The ARDY, RRDY, and XRDY bits in I2CSTR are not cleared when I2CISRC is read. To clear one of these bits, write a 1 to it.
2. The arbiter determines which of the remaining interrupt requests has the highest priority, writes the code for that interrupt to I2CISRC, and forwards the interrupt request to the CPU.

**Table 25-6. 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 should not be used when in FIFO mode. Use the FIFO interrupts instead. |
| RRDYINT | Receive ready condition: The data receive register (I2CDRR) is ready to be read because data has been copied from the receive shift register (I2CRSR) to I2CDRR.<br>As an alternative to using RRDYINT, the CPU can poll the RRDY bit of I2CSTR. RRDYINT should not be used when in FIFO mode. Use the FIFO interrupts instead. |
| ARDYINT | Register-access ready condition: The I2C module registers are ready to be accessed because the previously programmed address, data, and command values have been used.<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. |

**Figure 25-15. Enable Paths of the I2C Interrupt Requests**




The normal transmit interrupt timing makes it possible for stale data to remain in the transmit buffer if a transaction is aborted in the middle of a byte. To avoid this, set the FCM bit in the I2CEMDR register. When this bit is set, the transmit data ready interrupt is generated only when data is required for a bus transaction. In master mode, the interrupt is first generated when the ACK of the address byte is received. In slave mode, the interrupt is first generated when the address is matched. Further interrupts are generated when the data is ACKed. In this mode XRDY is asserted at the same time as the transmit ready interrupt.

### *25.4.2 I2C FIFO Interrupts*

In addition to the seven basic I2C interrupts, the transmit and receive FIFOs each contain the ability to generate an interrupt (I2CINT2A). The transmit FIFO can be configured to generate an interrupt after transmitting a defined number of bytes, up to 16 . The receive FIFO can be configured to generate an interrupt after receiving a defined number of bytes, up to 16. These two interrupt sources are ORed together into a single maskable CPU interrupt. The interrupt service routine can then read the FIFO interrupt status flags to determine from which source the interrupt came. See the I2C transmit FIFO register (I2CFFTX) and the I2C receive FIFO register (I2CFFRX) descriptions.

## 25.5 Resetting or Disabling the I2C Module

You can reset or disable the I2C module in two ways:

- Write 0 to the I2C reset bit (IRS) in the I2C mode register (I2CMDR). All status bits (in I2CSTR) are forced to their default values, and the I2C module remains disabled until IRS is changed to 1. The SDA and SCL pins are in the high-impedance state.
- Initiate a device reset by driving the $\overline{\text{XRS}}$ pin low. The entire device is reset and is held in the reset state until you drive the pin high. When the $\overline{\text{XRS}}$ pin is released, all I2C module registers are reset to their default values. The IRS bit is forced to 0, which resets the I2C module. The I2C module stays in the reset state until you write 1 to IRS.

The IRS must be 0 while you configure or reconfigure the I2C module. Forcing IRS to 0 can be used to save power and to clear error conditions.

## 25.6 Registers

### 25.6.1 Inter-Integrated Circuit Module Base Addresses

**Table 25-7. I2C Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| I2caRegs | I2C_REGS | 0x0000_7300 | 0x0000_733F |

### 25.6.1.1 I2C_REGS Registers

Table 25-8 lists the memory-mapped registers for the I2C_REGS. All register offset addresses not listed in Table 25-8 should be considered as reserved locations and the register contents should not be modified.

**Table 25-8. 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 25-9 shows the codes that are used for access types in this section.

**Table 25-9. I2C_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| R=0 | R | Read |
| **Write Type** | | |
| W | W | Write |
| W1C | 1C<br>W | 1 to clear<br>Write |
| W=1 | 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. |

#### 25.6.1.1.1 I2COAR Register (Offset = 0h) [reset = 0h]

I2COAR is shown in Figure 25-16 and described in Table 25-10.

Return to Summary Table.

I2C Own address

**Figure 25-16. I2COAR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | OAR | |
| R-0h | | | | | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OAR | | | | | | | |
| R/W-0h | | | | | | | |

**Table 25-10. 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 |

#### 25.6.1.1.2 I2CIER Register (Offset = 1h) [reset = 0h]

I2CIER is shown in Figure 25-17 and described in Table 25-11.

Return to Summary Table.

I2C Interrupt Enable

**Figure 25-17. I2CIER Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | AAS | SCD | XRDY | RRDY | ARDY | NACK | ARBL |
| R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 25-11. 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 |

#### 25.6.1.1.3 I2CSTR Register (Offset = 2h) [reset = 0h]

I2CSTR is shown in Figure 25-18 and described in Table 25-12.

Return to Summary Table.

I2C Status

**Figure 25-18. I2CSTR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | SDIR | NACKSNT | BB | RSFULL | XSMT | AAS | AD0 |
| R-0h | R/W1C-0h | R/W1C-0h | R-0h | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | BYTESENT | SCD | XRDY | RRDY | ARDY | NACK | ARBL |
| R-0h | R/W1C-0h | R/W1C-0h | R-0h | R/W1C-0h | R/W1C-0h | R/W1C-0h | R/W1C-0h |

**Table 25-12. 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. |

**Table 25-12. 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 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 | 0h | Transmit shift register empty bit.<br>XSMT = 0 indicates that the transmitter has experienced<br>underflow. Underflow occurs when the transmit shift register (I2CXSR) is empty but the data<br>transmit register (I2CDXR) has not been loaded since the last I2CDXR-to-I2CXSR transfer. The<br>next I2CDXR-to-I2CXSR transfer will not occur until new data is in I2CDXR. If new data is not<br>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 | RESERVED | R | 0h | Reserved |

**Table 25-12. I2CSTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | BYTESENT | R/W1C | 0h | Byte Transmit over indication.<br>BYTESENT is set when the master/slave has successfully sent the byte on SCL/SDA lines. This is diagnostic register which needs to be explicitly cleared by Software. In case not cleared the stale status would keep reflecting as no automated clear incorporated to avoid corner conditions.<br>Reset type: SYSRSn<br>0h (R/W) = The I2C module has not finished transmitting the next data byte. BYTESENT 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.<br>1h (R/W) = The I2C module has completed the transmission of a byte. |
| 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. |
| 4 | XRDY | R | 0h | Transmit-data-ready interrupt flag bit. When not in FIFO mode, XRDY indicates that the data transmit register (I2CDXR) is ready to accept new data.<br>FCM=0 : When the previous data has been copied from I2CDXR to the transmit shift register (I2CXSR). The CPU can poll XRDY or use the XRDY interrupt request When in FIFO mode, use TXFFINT instead.<br>FCM=1: XRDY is asserted only when next data is required it gets de asserted with write to I2CDXR. Both Polling and interrupt based data transfers are allowed in the FCM mode.<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<br>(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. |

**Table 25-12. I2CSTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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. |
| 1 | NACK | R/W1C | 0h | No-acknowledgment interrupt flag bit.<br>NACK applies when the I2C module is a transmitter (master or slave). NACK indicates whether the I2C module has detected an acknowledge bit (ACK) or a noacknowledge bit (NACK) from the receiver. The CPU can poll NACK or use the NACK interrupt request<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 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. |

**Table 25-12. I2CSTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 mastertransmitter. 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. |

#### 25.6.1.1.4 I2CCLKL Register (Offset = 3h) [reset = 0h]

I2CCLKL is shown in Figure 25-19 and described in Table 25-13.

Return to Summary Table.

I2C Clock low-time divider

**Figure 25-19. I2CCLKL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I2CCLKL | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 25-13. 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 |

#### 25.6.1.1.5 I2CCLKH Register (Offset = 4h) [reset = 0h]

I2CCLKH is shown in Figure 25-20 and described in Table 25-14.

Return to Summary Table.

I2C Clock high-time divider

**Figure 25-20. I2CCLKH Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I2CCLKH | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 25-14. 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 |

#### 25.6.1.1.6 I2CCNT Register (Offset = 5h) [reset = 0h]

I2CCNT is shown in Figure 25-21 and described in Table 25-15.

Return to Summary Table.

I2C Data count

**Figure 25-21. I2CCNT Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I2CCNT | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 25-15. I2CCNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | I2CCNT | R/W | 0h | Data count value. ICDC indicates the number of data bytes to transfer or receive.<br>The value in I2CCNT is a don't care when the RM bit in I2CMDR is set to 1.<br>The start value loaded to the internal data counter is 65536.<br>The start value loaded to internal data counter is 1-65535.<br>Reset type: SYSRSn |

#### 25.6.1.1.7 I2CDRR Register (Offset = 6h) [reset = 0h]

I2CDRR is shown in Figure 25-22 and described in Table 25-16.

Return to Summary Table.

I2C Data receive

**Figure 25-22. I2CDRR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATA | | | | | | | |
| R-0h | | | | | | | |

**Table 25-16. 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 |

#### 25.6.1.1.8 I2CSAR Register (Offset = 7h) [reset = 3FFh]

I2CSAR is shown in Figure 25-23 and described in Table 25-17.

Return to Summary Table.

I2C Slave address

**Figure 25-23. I2CSAR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | SAR | |
| R-0h | | | | | | R/W-3FFh | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SAR | | | | | | | |
| R/W-3FFh | | | | | | | |

**Table 25-17. 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 |

#### 25.6.1.1.9 I2CDXR Register (Offset = 8h) [reset = 0h]

I2CDXR is shown in Figure 25-24 and described in Table 25-18.

Return to Summary Table.

I2C Data Transmit

**Figure 25-24. I2CDXR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATA | | | | | | | |
| R/W-0h | | | | | | | |

**Table 25-18. 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 |

#### 25.6.1.1.10 I2CMDR Register (Offset = 9h) [reset = 0h]

I2CMDR is shown in Figure 25-25 and described in Table 25-19.

Return to Summary Table.

I2C Mode

**Figure 25-25. I2CMDR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| NACKMOD | FREE | STT | RESERVED | STP | MST | TRX | XA |
| R/W-0h | R/W-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RM | DLB | IRS | STB | FDF | BC | | |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | | |

**Table 25-19. I2CMDR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | NACKMOD | R/W | 0h | NACK mode bit.<br>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 |

**Table 25-19. I2CMDR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 12 | RESERVED | R | 0h | Reserved |
| 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).<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<br>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. |

**Table 25-19. 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:<br>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). |

**Table 25-19. 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 |

#### 25.6.1.1.11 I2CISRC Register (Offset = Ah) [reset = 0h]

I2CISRC is shown in Figure 25-26 and described in Table 25-20.

Return to Summary Table.

I2C Interrupt Source

**Figure 25-26. 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 25-20. I2CISRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-8 | WRITE_ZEROS | R/W | 0h | 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>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 |

#### 25.6.1.1.12 I2CEMDR Register (Offset = Bh) [reset = 1h]

I2CEMDR is shown in Figure 25-27 and described in Table 25-21.

Return to Summary Table.

I2C Extended Mode

**Figure 25-27. I2CEMDR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | FCM | BC |
| R-0h | | | | | | R/W-0h | R/W-1h |

**Table 25-21. I2CEMDR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-2 | RESERVED | R | 0h | Reserved |
| 1 | FCM | R/W | 0h | Forward Compatibility mode.<br>This bit when programmed brings the functionality of Tx request only when Tx data required regardless of data status in Tx buffer for non-FIFO mode.<br>This register affects the XRDY behavior hence needs to be set after releasing the IRS (I2CMDR[5]).<br>Reset type: SYSRSn<br>0h (R/W) = Legacy functionality of requesting Tx data upon buffer copy to shift register or upon start condition is active. Stale data is reused after illegal start, ARB Lost, NACK conditions.<br>1h (R/W) = New functionality of requesting data only upon ACK (address/data) is active. |
| 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.<br>Reset type: SYSRSn<br>0h (R/W) = See Figure 9-17 for details.<br>1h (R/W) = See Figure 9-17 for details. |

#### 25.6.1.1.13 I2CPSC Register (Offset = Ch) [reset = 0h]

I2CPSC is shown in Figure 25-28 and described in Table 25-22.

Return to Summary Table.

I2C Prescaler

**Figure 25-28. I2CPSC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IPSC | | | | | | | |
| R/W-0h | | | | | | | |

**Table 25-22. 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 |

#### 25.6.1.1.14 I2CFFTX Register (Offset = 20h) [reset = 0h]

I2CFFTX is shown in Figure 25-29 and described in Table 25-23.

Return to Summary Table.

I2C FIFO Transmit

**Figure 25-29. I2CFFTX Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | I2CFFEN | TXFFRST | TXFFST | | | | |
| R-0h | R/W-0h | R/W-0h | R-0h | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXFFINT | TXFFINTCLR | TXFFIENA | TXFFIL | | | | |
| R-0h | R=0/W=1-0h | R/W-0h | R/W-0h | | | | |

**Table 25-23. 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/W=1 | 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. |

**Table 25-23. I2CFFTX Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4-0 | TXFFIL | R/W | 0h | Transmit FIFO interrupt level.<br><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 these devices has a 16-level transmit FIFO, these bits cannot be configured for an interrupt of more than 16 FIFO levels.<br>Reset type: SYSRSn |

### 25.6.1.1.15 I2CFFRX Register (Offset = 21h) [reset = 0h]

I2CFFRX is shown in Figure 25-30 and described in Table 25-24.

Return to Summary Table.

I2C FIFO Receive

**Figure 25-30. I2CFFRX Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | RXFFRST | RXFFST | | | | |
| R-0h | | R/W-0h | R-0h | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RXFFINT | RXFFINTCLR | RXFFIENA | RXFFIL | | | | |
| R-0h | R=0/W=1-0h | R/W-0h | R/W-0h | | | | |

**Table 25-24. 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/W=1 | 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 these devices has a 16-level receive FIFO, these bits cannot be configured for an interrupt of more than 16 FIFO levels.<br>Reset type: SYSRSn |

# Power Management Bus Module (PMBus)

This chapter describes the features and operation of the Power Management Bus (PMBus) module.

**Topic** **Page**

## 26.1 Introduction

The PMBus module provides an interface between the microcontroller and devices compliant with the SMI Forum PMBus Specification Part I version 1.0 and Part II version 1.1. PMBus is based on SMBus, which uses a similar physical layer to I2C. This guide assumes the reader is familiar with the PMBus, SMBus, and I2C bus specifications.

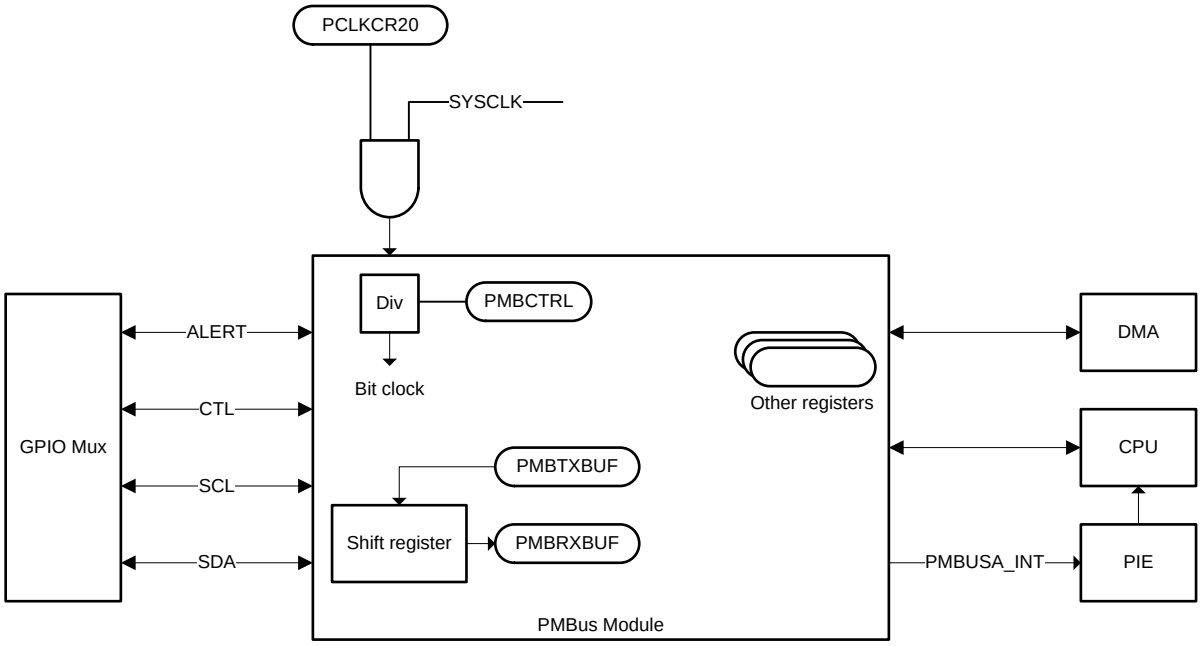
### *26.1.1 Features*

The PMBus module has the following features:

- Compliance with the SMI Forum PMBus Specification (Part I v1.0 and Part II v1.1)
- Support for master and slave modes
- Support for I2C modes
- Support for three speeds:
  - Standard Mode: Up to 100 kHz
  - Fast Mode: 400 kHz
  - Fast Mode+: 1000 kHz
    - Note: This mode is applicable to the PMBus module operating in I2C mode, with an input clock frequency of 20MHz.
- Packet error checking
- CONTROL and ALERT signals
- Clock high and low time-outs
- Four-byte transmit and receive buffers
- One maskable interrupt, which can be generated by several conditions:
  - Receive data ready
  - Transmit buffer empty
  - Slave address received
  - End of message
  - ALERT input asserted
  - Clock low time-out
  - Clock high time-out
  - Bus free

### 26.1.2 Functional Description

**Figure 26-1. PMBus Module Conceptual Block Diagram**




The PMBus module handles the lower levels of the PMBus protocol. In addition to controlling signal levels and timing, parsing addresses, and buffering data, it also directly supports complex transactions such as Read Word and Process Call.

There are four PMBus signals. SCL is the bus clock. It is normally controlled by the master, but may be held low by a slave to delay a transaction, allowing more time for processing. SDA is the bidirectional data line. CONTROL is a slave input that can trigger an interrupt. It can be used to tell a slave device to shut down. ALERT is a slave output/master input that allows a slave to request attention from the master.

The SDA and SCL timings produced by the module are derived from SYSCLK. To comply with the PMBus timing specs, the bit clock divider must be set via the PMBTIMCLK register to provide a bit clock of 10 MHz or less.

## 26.2 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

Some IO functionality is defined by GPIO register settings independent of this peripheral. For input signals, the GPIO input qualification should be set to asynchronous mode by setting the appropriate GPxQSELn register bits to 11b. The internal pullups can be configured in the GPyPUD register.

See the *GPIO* chapter for more details on GPIO mux and settings.

## 26.3 Slave Mode Operation

This section describes the configuration and operation of the PMBus module in slave mode.

### 26.3.1 Configuration

First, write a suitable clock divider to the PMBCTRL register's CLKDIV field to produce a bit clock frequency of less than 10 MHz. To activate slave mode, set the SLAVE_EN bit in the PMBCTRL register. Next, set up the PMBSC register. The following options are configurable:

- Slave address and mask (SLAVE_ADDR and SLAVE_MASK): Sets the slave address and mask for message acceptance.

- Manual slave address acknowledgement (MAN_SLAVE_ACK): When enabled, allows software to decide whether to ACK an address. When disabled, the decision to ACK is made automatically based on the slave address and mask.
- PEC enable (PEC_ENA): If Packet Error Checking (PEC) is used on the bus, set this bit.
- Manual command byte acknowledgement (MAN_CMD): Similar to manual slave acknowledgement, setting this bit allows software to decide whether to ACK a command byte.
- Number of bytes to acknowledge automatically (RX_BYTE_ACK_CNT): This is normally set to the max value, which allows the entire receive buffer to be used. However, smaller values may be used if the application requires that erroneous messages be detected and NACKed as soon as possible.

Manual acknowledgement is done by writing a one to the PMBACK register. Even with automatic acknowledgement, some writes to PMBACK are required. If the message (not including the address) is longer than 4 bytes, each packet of 4 bytes much be acknowledged. The PMBus module will stretch the clock (hold it low) until an ACK is issued. The module will then pull the data line low and release the clock, providing the ACK signal to the master.

If the complete message or the last part of the message is less than 4 bytes (or the RX_BYTE_ACK_CNT limit), do not write to PMBACK.

Writing a zero to PMBACK will send a NACK. This may only be done when the module is waiting for an acknowledgement. If a zero is written at any other time, the NACK will be issued during the next message.

### *26.3.2 Message Handling*

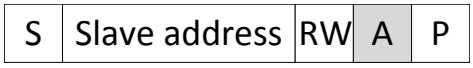
This section describes some of the message types for PMBus and how to determine which message type is being received in slave mode. It is oriented toward the most efficient mode of operation – with automatic address and command acknowledgment. It is also oriented toward having PEC enabled.

If automatic address acknowledgement is disabled, all messages will start with a SLAVE_ADDR_READY. Read commands will have two, one for the read, and one for the write. If automatic command acknowledgement is enabled, a DATA READY will occur for that as well. If the message has no PEC, the number of bytes available will be one less. For example, with PEC, a QUICK COMMAND will have one byte, the PEC. With no PEC, a QUICK COMMAND will have zero bytes.

Note that the byte count does not increment as bytes arrive. No bits are set in the PMBST register until a stop message is received, the receive buffer is full, or a fault occurs. Then all appropriate bit values are placed in the register together. All that is necessary to receive a quick command is to ACK the message by writing a 1 to the PMBACK register.

#### 26.3.2.1 Quick Command

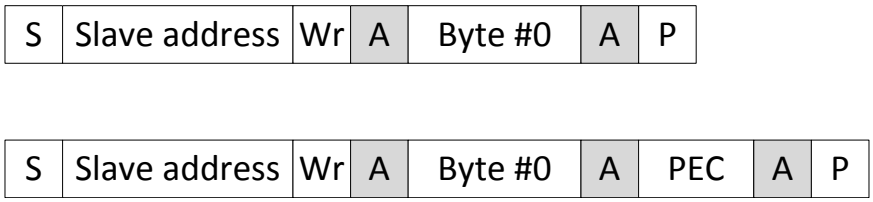
**Figure 26-2. Quick Command Message**




Quick Commands received by the PMBus module in slave mode require a simple acknowledgement of the received device address. In automatic address acknowledge mode, the module processes the quick command without firmware interaction. Upon receipt of the end of message, the firmware has the option to read the received address in the PMBHSA register. In manual address acknowledge mode, the address is acknowledged by writing to the PMBACK register.

#### 26.3.2.2 Send Byte

**Figure 26-3. Send Byte Message with and without PEC**



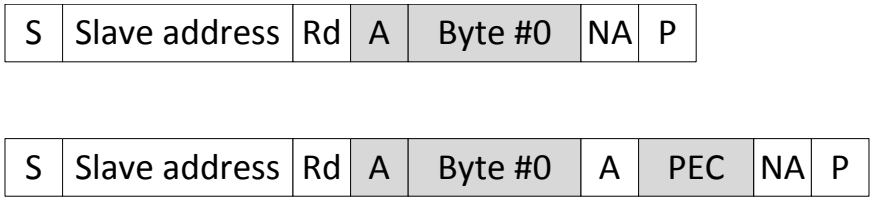

A Send Byte message consists of the device address, a single data byte and an optional PEC byte. To process the PEC byte correctly, PEC processing needs to be enabled in the PMBSC register. In automatic address acknowledge mode, the data and optional PEC byte are acknowledged without firmware interaction. The module generates an End of Message interrupt, reads the status register and finds the data ready indication bit set. In manual mode, the address is acknowledged by the firmware, while the remaining data and PEC bytes are acknowledged by the module

The PMBus module stores Data Byte #0 into the PMBRXBUF register. The data byte will be stored into bits 7-0. In non-PEC mode, the RX Byte Count in the PMBSTS register will indicate one byte received. If PEC processing is enabled, the PEC byte is also stored into the PMBRXBUF register, with the PEC byte residing in bits 15-8. The RX Byte Count in the PMBSTS register will indicate two bytes received. The PEC Valid bit in the PMBSTS register indicates the validity of the received PEC byte.

When a Send Byte message is received, the Data Ready bit is set along with the EOM and, assuming that the PEC is valid, the PEC valid bit. The read byte count (RD_BYTE_COUNT) register will contain a 2. All that is necessary to receive a send byte command is to ACK the message by writing a 1 to the PMBACK register. Before doing the ACK, read the byte from the lowest byte of the PMBRXBUF register.

#### 26.3.2.3 Receive Byte

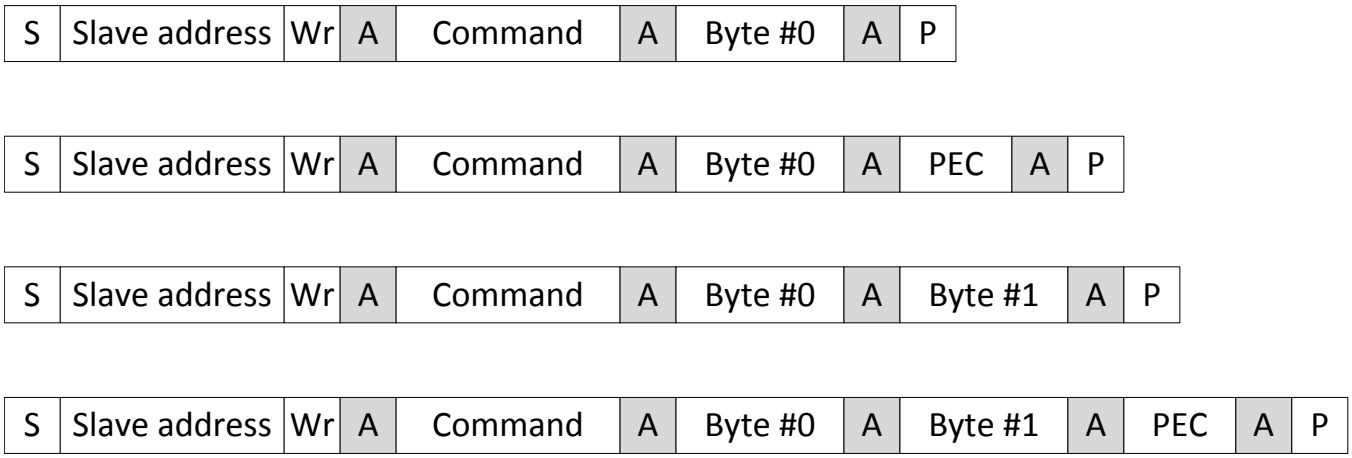
**Figure 26-4. Receive Byte Message with and without PEC**




A Receive Byte message consists of the device address, a single data byte and an optional PEC byte. In automatic address acknowledge mode, the firmware receives a data request interrupt following reception of the slave address. The data byte to be sent to the master is stored into bits 7-0 of the PMBTXBUF register and Transmit Byte Count bits within the PMBSC register are set to a value of one. If PEC processing is enabled, the Transmit PEC bit (Bit 19) within the PMBSC register is set to '1', along with the Enable PEC bit (Bit 15). The module will automatically append the calculated PEC byte at the completion of the message.

#### 26.3.2.4 Write Byte/Word

**Figure 26-5. Write Byte and Write Word Messages with and without PEC**




The Write Byte and Write Word messages consist of a slave address, a command word, transmitted data bytes and an optional PEC byte. In automatic address acknowledge mode, the data bytes and optional PEC byte are acknowledged without firmware interaction. The acknowledgement of the command word is configured through the PMBSC register. The firmware receives an End of Message interrupt in all cases except for Write Word with PEC message, reads the status register and finds the data ready indication bit set.

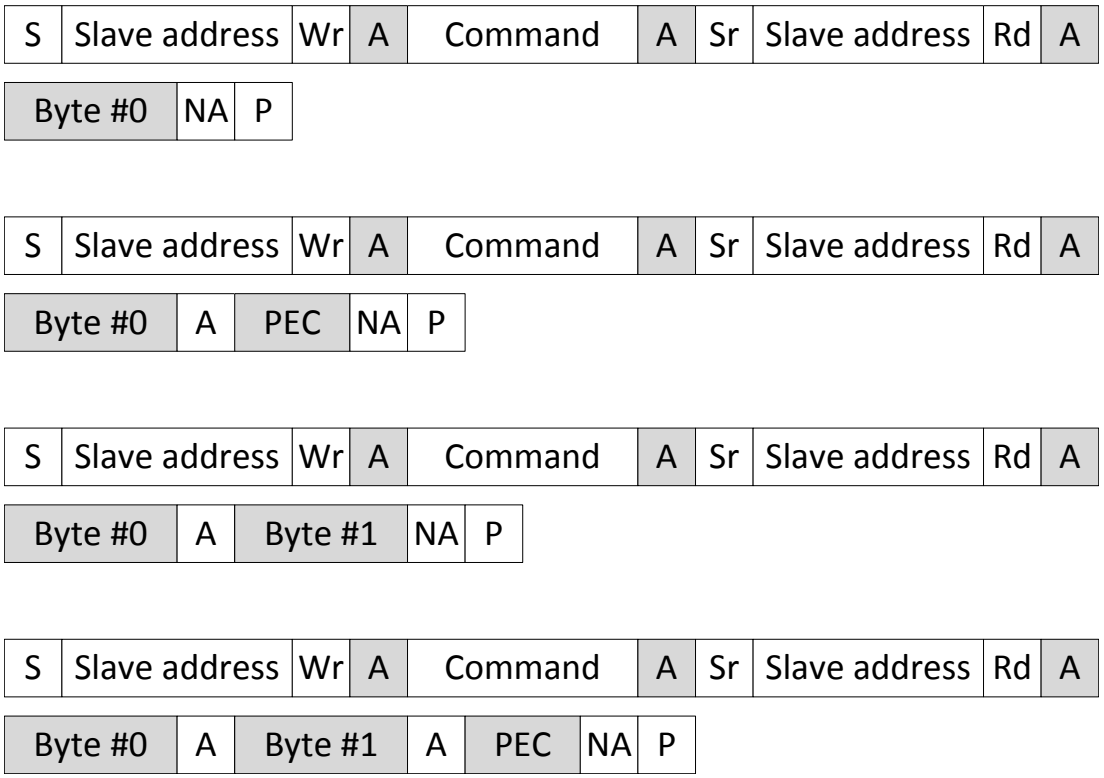
In the case of a Write Word with PEC byte message, the data ready interrupt is enabled after receiving 4 bytes (command byte, the 2 data bytes and the PEC byte). The firmware reads the data from the PMBRXBUF register and must write the PMBACK register to acknowledge back to the master. The PMBus module holds SCL low until the firmware responds to the received data.

In all other cases, the EOM interrupt is received and data can be read from the PMBRXBUF register. The firmware is not required to send an acknowledgement back to the master.

The Write Byte message will look exactly the same as the Send Byte, except the RD_BYTE_COUNT register will contain a 3. The Write Word message will have a RD_BYTE_COUNT of 4.

#### 26.3.2.5 Read Byte/Word

**Figure 26-6. Read Byte and Read Word Messages with and without PEC**



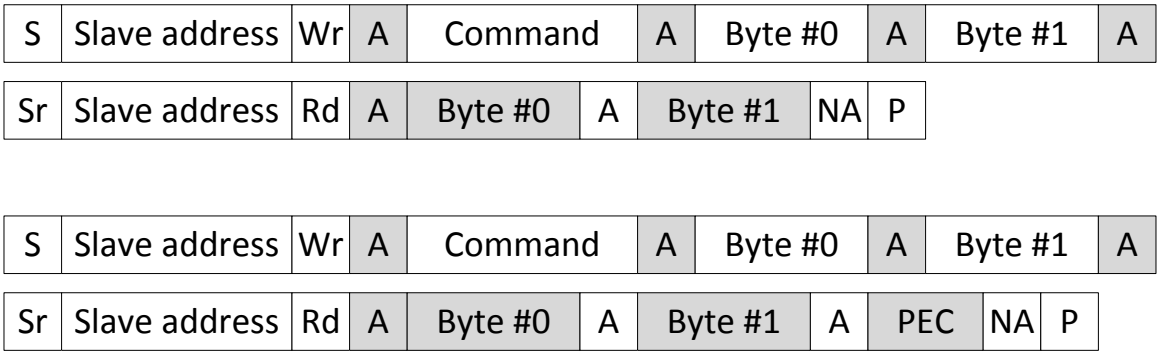

The Read Byte and Read Word messages consist of a slave address, a command word, received data bytes from a slave and an optional PEC byte. Address and command acknowledgment is configured through the PMBSC register. In automatic mode, following receipt of the repeated start and slave address, the PMBus module provides a data ready and data request interrupt. The received command byte is found in bits 7-0 of the PMBRXBUF register. The firmware responds to the data request by programming the data bytes into the PMBTXBUF register and the TX Byte Count bits in the PMBSC register. If PEC processing is enabled, the Transmit PEC bit should also be asserted. An EOM interrupt indicates completion of the message to the Master.

When the repeated start (Sr) signal is received, the Data Ready bit will be asserted with a RD_BYTE_COUNT of 1. At this point, the operation cannot be distinguished from a group command send byte message. When the same device address is sent out with a read, then the Data Request bit will be asserted. If data has already been written to the PMTXBUF register before the Device Addr is received, then the Data Request bit will not be asserted. So if group commands are also expected, it may be necessary to read the Data Ready with a RD_BYTE_COUNT of 1, and then wait and see whether the next event is an EOM or a Data Request. If it is an EOM, then the command should be processed as a group send byte. If it is a Data Request, then the command should be processed as a read. Depending on the command, it could be a read byte, word, or block. If the PMBus module is polled, it is possible that both the Data Ready and the Data Request bits will be set between polling intervals, so that possibility should be considered in the design of the firmware.

Once the read command is recognized, it is necessary to respond by writing data to the PMBTXBUF register. It is also necessary to make sure that the values in the PMBSC register are correct. The transmit byte count and PEC bit must be set appropriately. For a read byte, the transmit byte count can be loaded with a 1. If the transmission of a PEC byte is desired, the TX_PEC bit should be set. After this, the data can be written to PMBTXBUF, which starts the transmission. All bytes should be written to PMBTXBUF at the same time. After the master receives the message, it will NACK the last byte to indicate that the correct number of bytes have been received. This will cause the EOM bit to be set in the PMBSTS register, indicating to the firmware that the Read Byte message is complete.

#### 26.3.2.6 Process Call

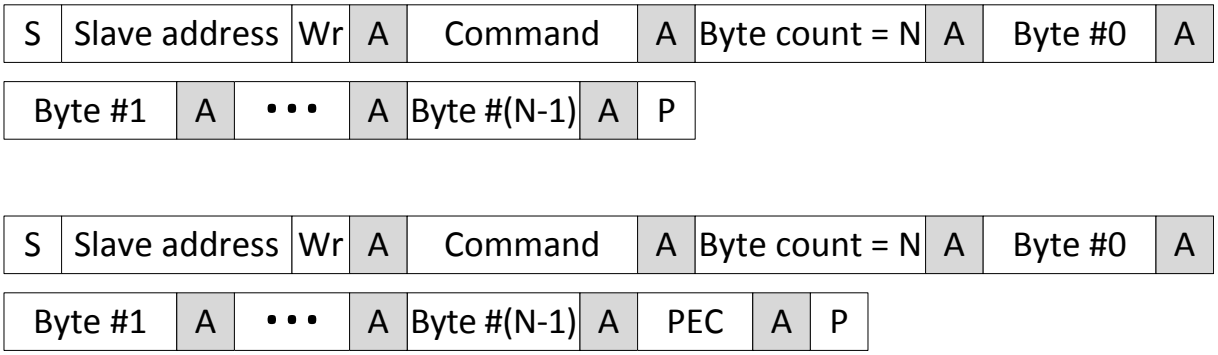
**Figure 26-7. Process Call Message with and without PEC**




The Process Call protocol consists of a Write Word message, followed by a Read Word message, without a stop condition between the two messages. Address and command acknowledgement is configured through the PMBSC register. In automatic mode, following receipt of the repeated start and slave address, the PMBus module provides a data ready and a data request interrupt. The repeated start bit is set in the PMBSTS register to indicate the receipt of the first part of the Process Call message. The received command byte is found in bits 7-0 of the PMBRXBUF register, while the two data bytes received from the master can be found in bits 23-8. Upon receipt of the repeated start and a data request from the module, the firmware programs the PMBTXBUF with the 2 data bytes to be sent to the master. If PEC processing is enabled, the Transmit PEC bit within the PMBSC register is asserted. The EOM interrupt will indicate the read word portion of the Process Call message has been completed by the module

#### 26.3.2.7 Block Write

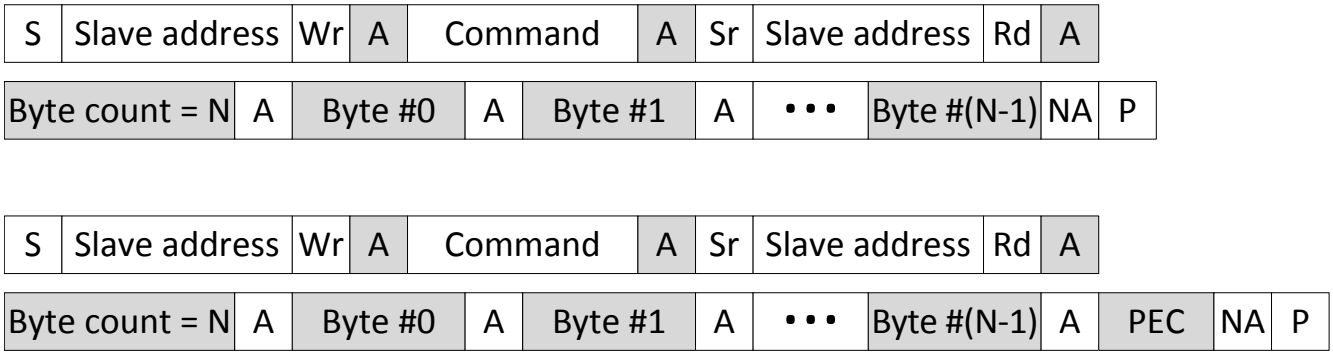
**Figure 26-8. Block Write Message with and without PEC**




The Block Write protocol is similar to Write Word in its structure, except that there are more than 2 data bytes in the message. Following the receipt of the command byte, the block length and 2 data bytes, the PMBus module provides a data ready interrupt. The module waits for the firmware to read the received data and program the acknowledge register. While waiting from acknowledgement from the firmware, the module will drive the clock line low, stalling the bus. The data ready interrupts will continue for the duration of the message at a frequency of every 4 data bytes. The number of bytes received can be found within the PMBSTS register. At the end of the message, less than 4 bytes may be stored in the PMBRXBUF register. The PEC Valid bit can be checked to determine if the received PEC value is accurate.

### 26.3.2.8 Block Read

**Figure 26-9. Block Read Message with and without PEC**

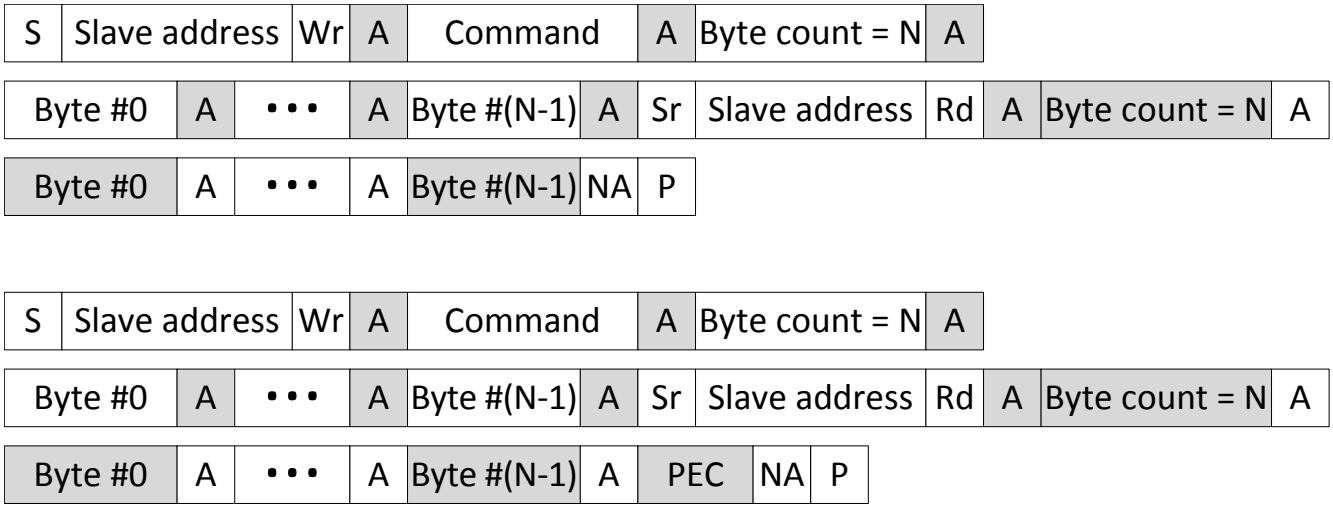



The Block Read protocol is similar to a Read Word in its structure, except that there are more than 2 data bytes in the message. Following the receipt of the repeated slave address, a data ready and data request interrupt is generated by the PMBus module. The command byte received from the master can be found in bits 7-0 of the PMBRXBUF register. The SCL line is held low until the firmware programs data bytes into the PMBTXBUF register. The firmware is required to load the block length into bits 7-0 of the PMBTXBUF register during the initial programming of the register. After 4 bytes have been transmitted, the module will issue a data request interrupt and hold SCL low again until the firmware has programmed additional data into the PMBTXBUF register.

Block read starts the same as Read Word or Read Byte, but TX_COUNT is loaded with a 4 the first time, and TX_PEC is not set. Instead of waiting for an EOM after the first transmission, the firmware instead waits for a Data Request, indicating that the master is ready for more data. Until the last 4 or less bytes, the firmware simply writes a 4 to TX_COUNT and then writes the 4 bytes to PMBTXBUF. TX_PEC is left cleared. Then when the last 4 or fewer bytes are to be transmitted, the firmware writes out the appropriate byte count, sets the TX_PEC bit, and writes the data to PMBTXBUF. The PMBus module will write out the data, followed by the PEC, and then the EOM bit will be set when the master NACKs the PEC.

### 26.3.2.9 Block Write-Block Read Process Call

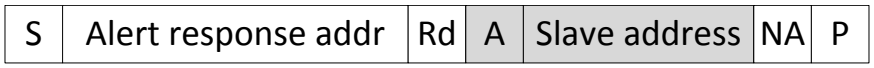
**Figure 26-10. Block Write-Block Read Process Call Message with and without PEC**



The Block Read-Block Write Process Call protocol combines the Block Write and Block Read protocols, removing the stop condition between the two messages. The processing of the Block Read-Block Write Process Call message is similar to the mode of operation for the Process Call message. After acknowledgement of the address and command bytes, the PMBus module generates a data ready interrupt upon detection of 4 data bytes or a repeated start condition. After receiving the repeated start, the firmware will be required to load transmit data to send to the master. Bits 7-0 of the initial programming of the PMBTXBUF register must represent the byte count of the block data sent to the master.

#### 26.3.2.10 Alert Response

**Figure 26-11. Alert Response Message**




The Alert Response Message is utilized when the master detects an alert condition from a slave on the PMBus. In automatic address acknowledge mode, upon detection of the Alert Response Address, the PMBus module provides an acknowledgement to the master and sends the programmed slave address within the PMBSC register. The module only responds to the message if the Alert En bit within PMBCTRL register has been previously set. After receiving the Alert Response message, the module will clear the alert condition and enable bit within the PMBCTRL register.

In manual address acknowledge mode, the firmware must read the received address from the PMBRXBUF register and transmit the desired slave address back to the master. The PMBCTRL register must be reprogrammed to disable the Alert En bit used to initiate the Alert Response message from the master.

#### 26.3.2.11 Extended Command

**Figure 26-12. Extended Command Write Byte and Write Word Messages with and without PEC**

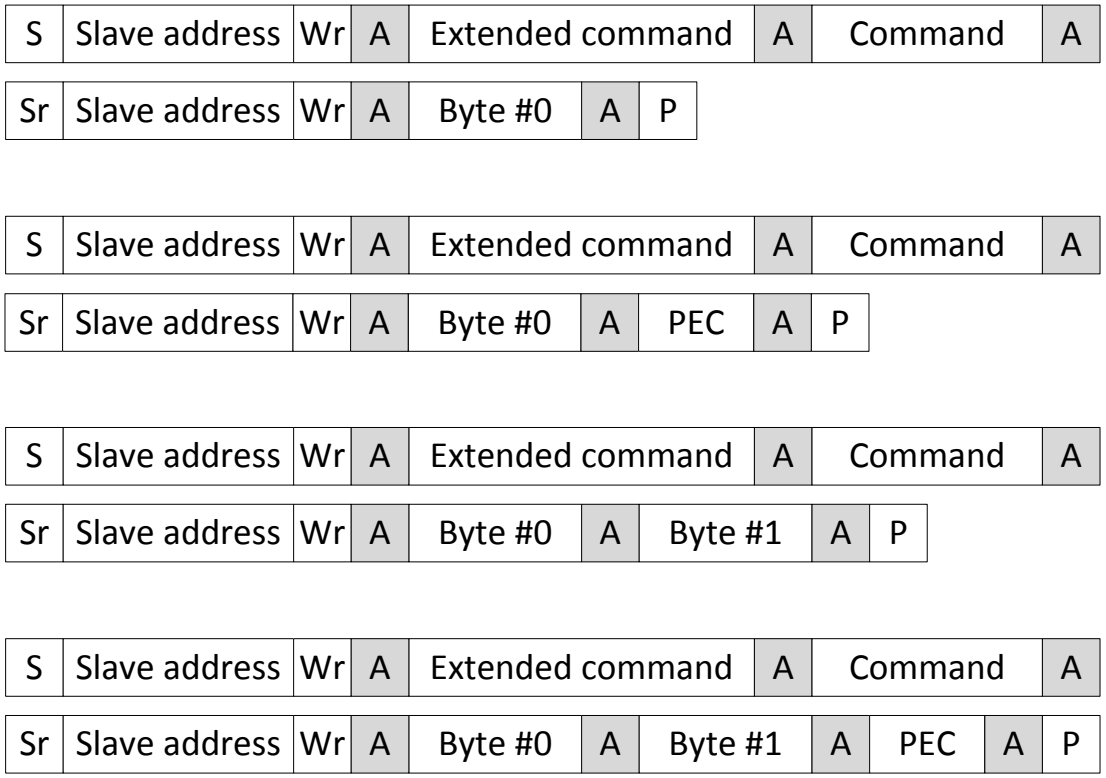
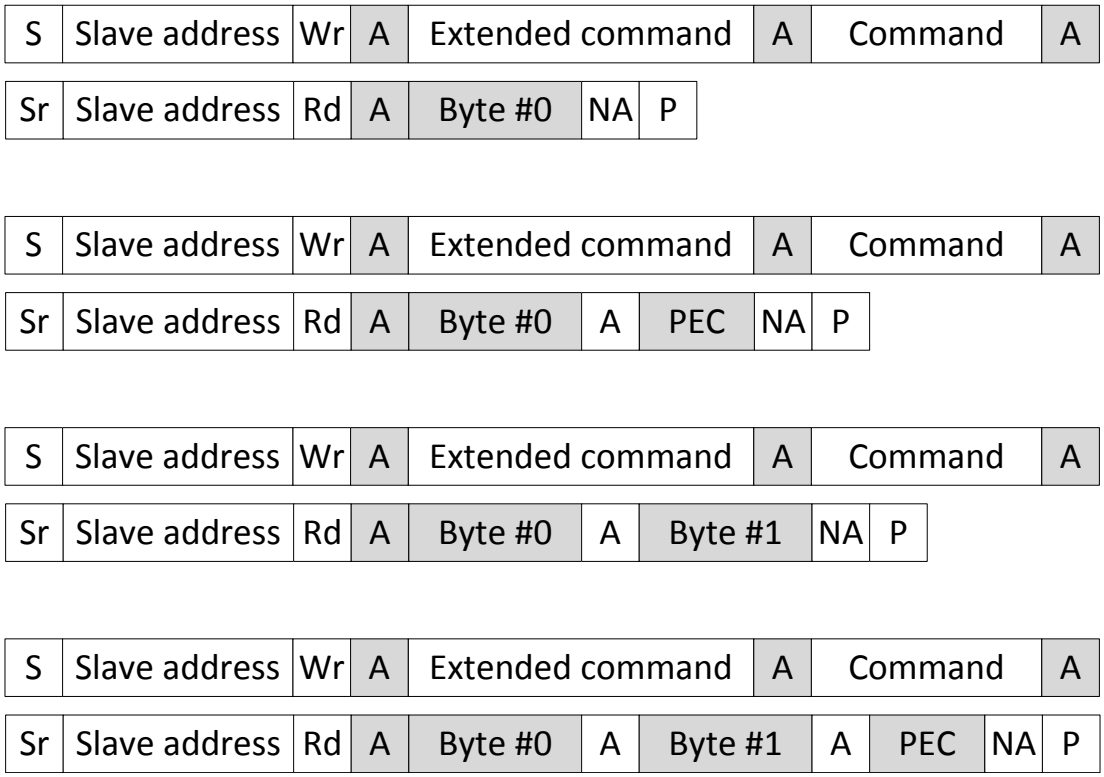


**Figure 26-13. Extended Command Write Byte and Write Word Messages with and without PEC**

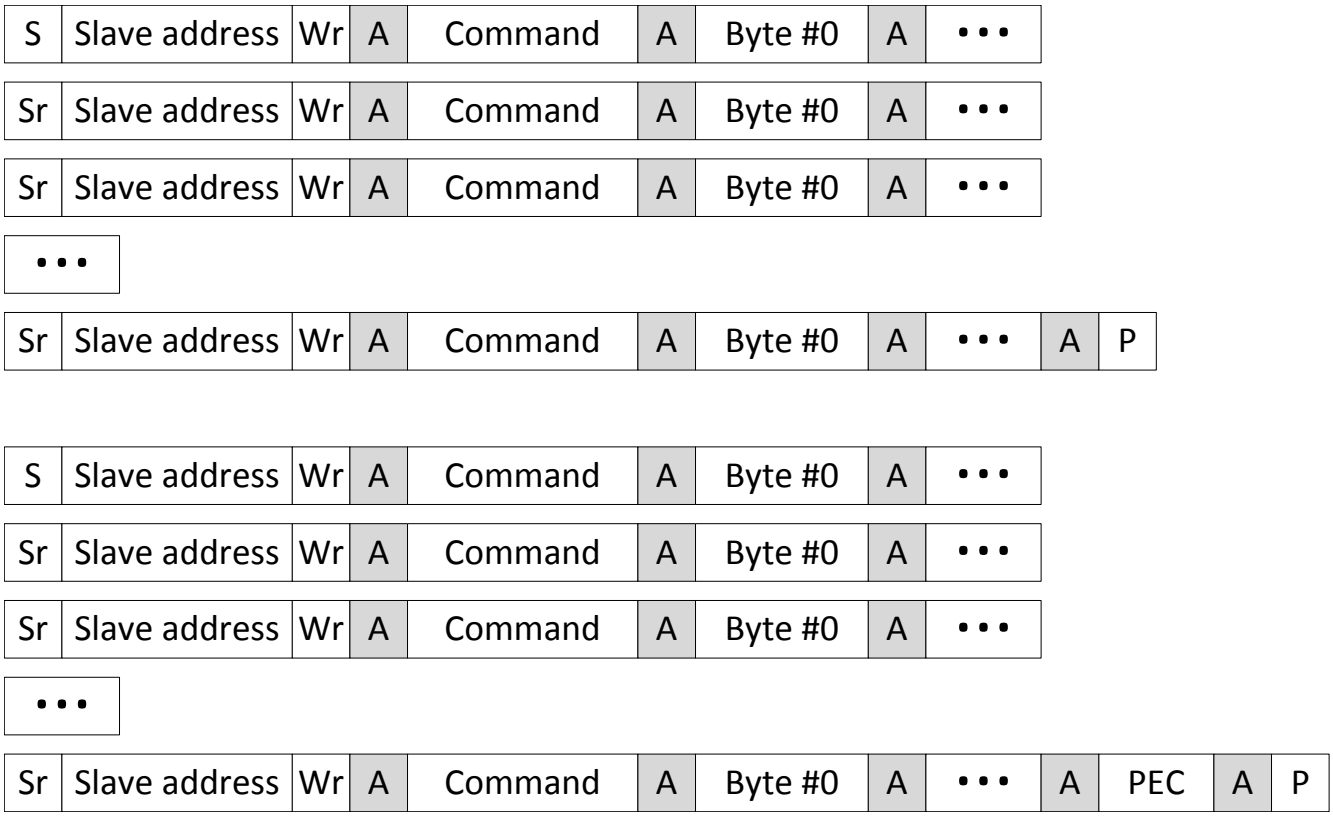



The PMBus module provides support for extended commands which allow for an extra 256 command codes. Both command bytes are stored in the PMBRXBUF register along with the data bytes. In recognizing the extended command messages, the Repeated Start bit and the Rd Byte Count Bits within the PMBSTS register are utilized. For Extended Command Write Byte/Write Word messages, the two command bytes are stored in bits 15-0 of the PMBRXBUF register. The initial command byte should hold the command extension code, representing utilization of the extended command protocol. The Repeated Start bit is also set, received after the retransmission of the device address. The Rd Byte Count equals 3 for an Ext Cmd Write Byte message and 4 for an Ext Cmd Write Word message.

For the Extended Command Read Byte/Read Word messages, the module will generate a data ready and data request interrupt following reception of the repeated device address. The two command bytes will be found in bits 15-0 of the PMBRXBUF register, with the initial command byte matching the command extension code. The firmware will be required to load transmit data to complete the message back to the master.

#### 26.3.2.12 Group Command

**Figure 26-14. Group Command Message with and without PEC**




The PMBus module supports the Group Command Protocol. The Group Command Protocol is used to send commands to more than one device within the same message. When devices on the bus detect the stop condition at the conclusion of the Group Command message, the received commands are executed concurrently. Following address and command acknowledgment, the module provides a data ready interrupt upon detection of 4 data bytes or the transmission of a repeated start on the bus. The firmware should wait for the EOM interrupt before processing the received command, as required by the use of the Group Command message.

For Group Commands, the data ready bit will be set as soon as the repeated start is received. The data can then be read into memory. But the data should not be acted upon until the EOM bit is set, which will occur when all of the messages have been received. Other than this delayed EOM, there is no difference for the slave firmware in receiving a Group Command than any other write message.

## 26.4 Master Mode Operation

This section describes the configuration and operation of the PMBus module in master mode.

### *26.4.1 Configuration*

First, write a suitable clock divider to the PMBCTRL register's CLKDIV field to produce a bit clock frequency of less than 10 MHz. To activate master mode, set the MASTER_EN bit and clear the SLAVE_EN bit in the PMBCTRL register. For each transaction, set up the PMBMC register. The following options are configurable:

- Slave address (SLAVE_ADDR): Sets the slave address for the next transaction.
- PEC enable (PEC_ENA): If Packet Error Checking (PEC) is used on the bus, set this bit.
- Extended command code enable (EXT_CMD): When set, uses two bytes for commands.
- Command code enable (CMD_ENA): When set, sends a command byte at the start of the transaction.

- Byte count (BYTE_COUNT): Determines the number of data bytes to transfer. This does not include the block length byte, which is generated automatically when needed.
- Special command enables (GRP_CMD and PRC_CALL): Enables special behavior for group commands and process calls.

Writing to the PMBMC register will start a transfer.

Manual acknowledgement of received data is not needed.

### 26.4.2 Message Handling

This section describes the behavior and required configuration for each command type.

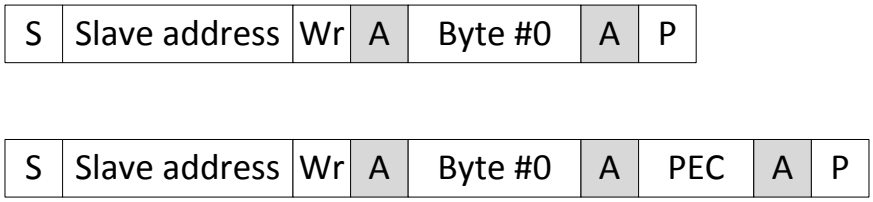
#### 26.4.2.1 Quick Command

**Figure 26-15. Quick Command Message**


Quick commands are initiated in master mode by simply programming the desired slave device address into the PMBMC. The byte count within the PMBMC register is configured to 0 bytes by writing all zeros to bits 15-8. Upon transmission of the device address, the PMBus module will monitor the slave acknowledgement of the address. If the address is not acknowledged, the NACK bit within the status register is enabled and the PMBus module automatically sends a stop condition on the bus to terminate the message. If the address is acknowledged, a data request is issued to the processor. The firmware writes a zero to the PMBACK to terminate the message, forcing the PMBus modules to write a stop condition onto the bus.

#### 26.4.2.2 Send Byte

**Figure 26-16. Send Byte Message with and without PEC**

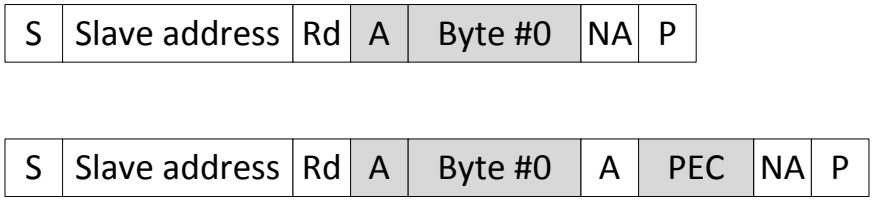



A Send Byte message consists of the device address, a single data byte and an optional PEC byte. To initiate a Send Byte message, the data byte to be transmitted to the slave is loaded into bits 7-0 of the PMBTXBUF register. The PMBMC register is configured with the device address. To transmit a PEC byte with the message, the PEC_EN bit within the PMBMC register is asserted high when the address is programmed.

After programming the PMBMC register, the PMBus module transmits the Send Byte message. The firmware can wait for an End of Message interrupt from the PMBus module. Upon receipt of the EOM interrupt, the PMBSTS register is read to verify the slave properly acknowledged the transmitted data.

#### 26.4.2.3 Receive Byte

**Figure 26-17. Receive Byte Message with and without PEC**

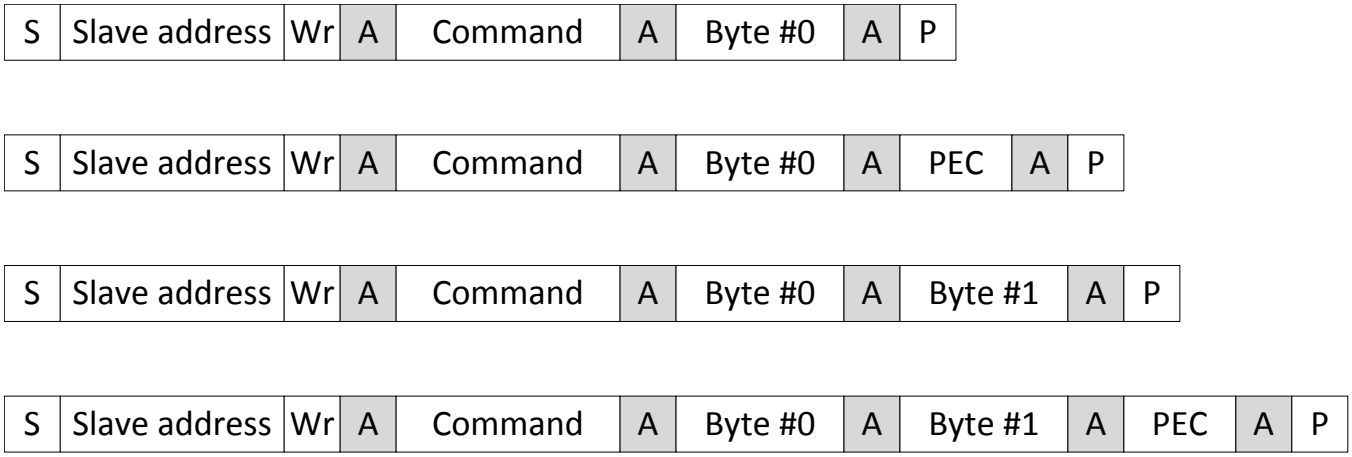


A Receive Byte message consists of the device address, a single data byte and an optional PEC byte. Data is being read from the slave in a Receive Byte message. To initiate a Receive Byte message, the firmware programs the device address, the R/W bit and the optional PEC_EN into the PMBMC register. The R/W bit is enabled high to indicate a read message type (data transmitted from slave to master).

After programming the PMBMC register, the PMBus module transmits the Receive Byte message. The firmware can wait for an End of Message interrupt from the PMBus module to verify the accuracy of the message transmission. Upon receipt of the EOM interrupt, the PMBSTS register is read to verify proper slave acknowledgement of the device address and to determine if any data is available for reading in the PMBRXBUF register. If PEC_EN was asserted in the PMBMC register, the PEC_VALID bit in the PMBSTS register is also checked to ensure a proper PEC byte was received from the slave with the received data.

#### 26.4.2.4 Write Byte/Word

**Figure 26-18. Write Byte and Write Word Messages with and without PEC**



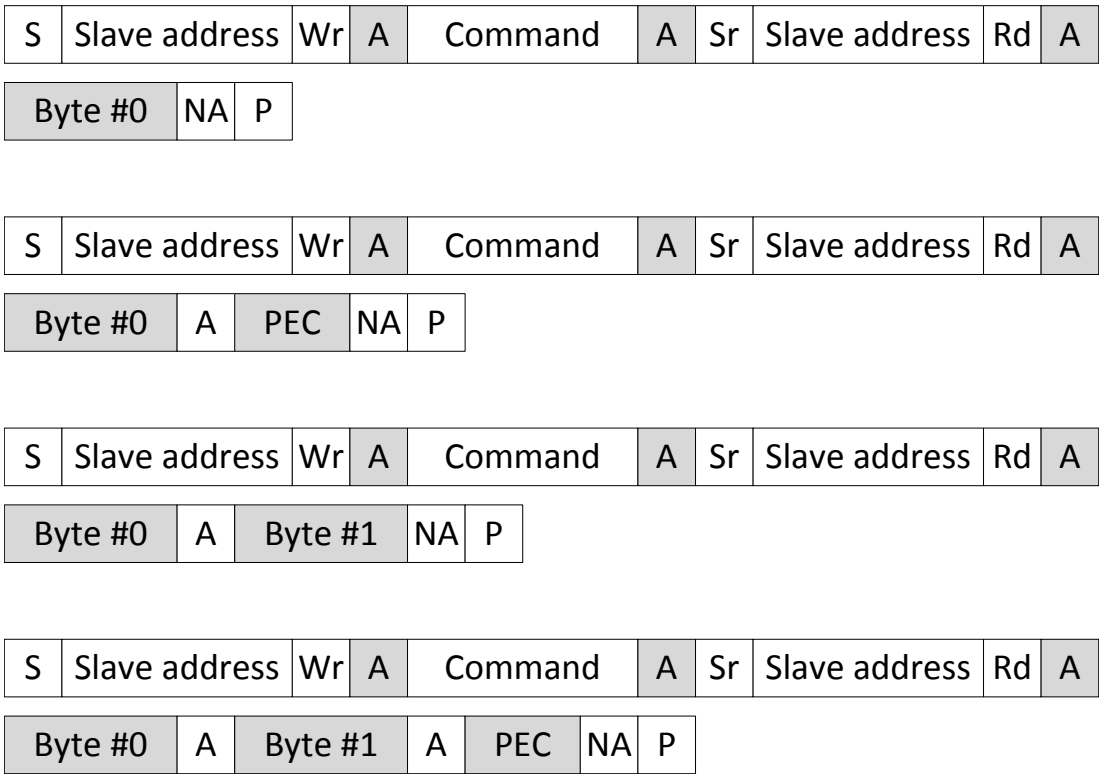

The Write Byte and Write Word messages consist of a device address, a command byte, transmitted data bytes and an optional PEC byte. Write Byte messages include a single byte, while the Write Word messages support transmission of 2 bytes to the corresponding slave module. Similar to the Send Byte protocol, the PMBMC register is configured to send 1 or 2 bytes, the CMD_EN bit is set to enable command byte transmission and the optional PEC_EN bit is set.

With the command byte transmission enabled, the format of the PMBTXBUF register differs from the Send Byte protocol. In bits 7-0, the firmware must program the command byte to be sent to the slave. The data byte(s) are programmed into bits 15-8 and bits 23-16.

After programming the PMBMC register, the PMBus module transmits the Write Byte/Word message. The firmware can wait for an End of Message interrupt from the module to verify the accuracy of the message transmission. The PMBSTS register indicates if the slave acknowledged the message properly.

### 26.4.2.5 Read Byte/Word

**Figure 26-19. Read Byte and Read Word Messages with and without PEC**

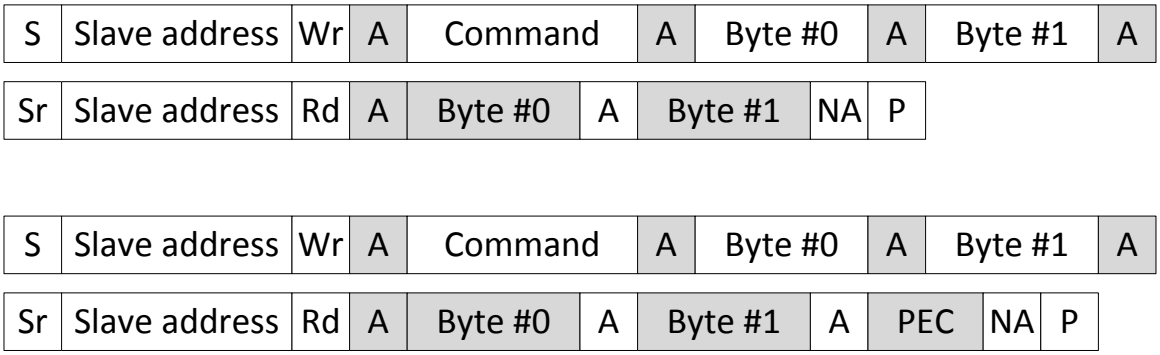



The Read Byte and Read Word messages consist of a device address, a command byte, received data bytes from a slave and an optional PEC byte. Read Byte messages include a single byte, while the Read Word message protocol supports receipt of 2 bytes from the slave. Similar to the Receive Byte Protocol, the PMBMC register is configured to receive 1 or 2 bytes, the CMD_EN bit is set and the PEC_EN is configured to expect or not expect a PEC byte appended to the message. The PMBus module will automatically terminate the message after the expected number of bytes is received from the slave or if the slave does not properly acknowledge any portion of the message.

In addition to programming the PMBMC register, the firmware is expected to load the command byte into bits 7-0 of the PMBTXBUF register. Any data received from the slave will be found in the PMBRXBUF register.

### 26.4.2.6 Process Call

**Figure 26-20. Process Call Message with and without PEC**



The Process Call protocol consists of a Write Word message, followed by a Read Word message, without a stop condition between the two messages. A PEC byte can be appended to the read data from the slave as an option to the message protocol. The PMBMC register includes a PRC_CALL bit, which enables the transmission of a Process Call message onto the PMBus. The PMBus module will automatically generate a repeated start condition and initiate the Read Word portion of the message when the process call bit is enabled.

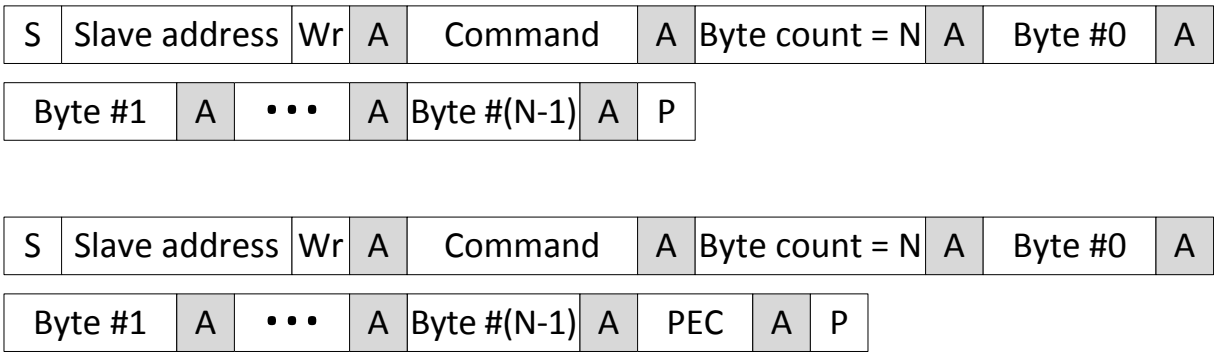
To complete the Write Word portion of the Process Call, the PMBTXBUF register is loaded with the command byte in bits 7-0 and the data bytes are loaded into bits 23-8 of the register.

After programming the PMBMC register, the PMBus module transmits the Process Call Message. The firmware can wait for an End of Message interrupt from the module to determine the validity of the message. Upon the receipt of the EOM, the PMBSTS register should indicate the receipt of 2 bytes from the Read Word portion of the Process Call message and the status of the Slave acknowledgment of the transmit data. If PEC processing is enabled, the PEC_VAL bit within the PMBSTS register indicates the accuracy of the PEC byte received from the slave during the Read Word part of the message.

The PRC_CALL bit within the PMBMC register must be disabled for the next non-Process Call message. Please note that any write to the PMBMC register initiates a message, so reconfiguration of the master is not recommended until the firmware requires a new message to be transmitted.

#### 26.4.2.7 Block Write

**Figure 26-21. Block Write Message with and without PEC**




The Block Write protocol is similar to a Write Word in its structure, with the exception of transmission of more than 2 data bytes in the message. Additionally, the first data byte following the command byte specifies the length of the block of data bytes. As with a majority of the message protocols, the PEC byte can be appended to the end of the write data to the slave.

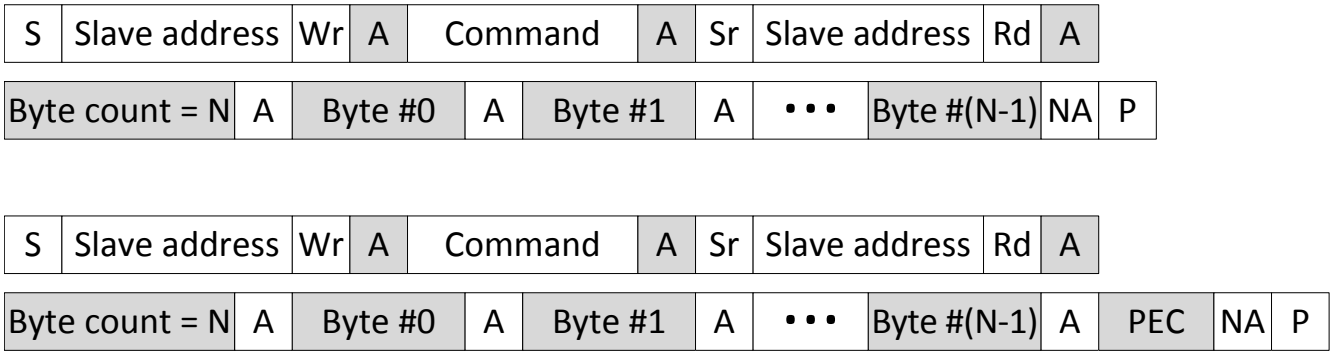
To initiate a Block Write message on the bus, the PMBMC register is programmed with the block length in the Byte Count bits. The block length is the number of data bytes, excluding the command byte and the first data byte that contains the block length. The PMBus module will automatically insert the block length into the message if the number of data bytes specified by the firmware exceeds 2. The initial write data is loaded into the PMBTXBUF register. With bits 7-0 representing the command byte, the remaining 3 bytes represent the first three data bytes following the block length.

Following programming of the PMBMC register, the Block Write message is transmitted. If the block length exceeds three bytes, the PMBus module will provide a data request interrupt, indicating the need for additional data bytes in the PMBTXBUF register. The PMBus module assumes that if more than 4 bytes are needed to complete the message, the firmware will utilize all 4 bytes when programming the PMBTXBUF register. If less than 4 bytes are needed to finish the Block Write message, the firmware only needs to program the appropriate bits of the PMBTXBUF register.

Upon completion of the message, the PMBus module issues an EOM interrupt. The PMBSTS register can be checked to verify the slave accepted the block of write data.

### 26.4.2.8 Block Read

**Figure 26-22. Block Read Message with and without PEC**



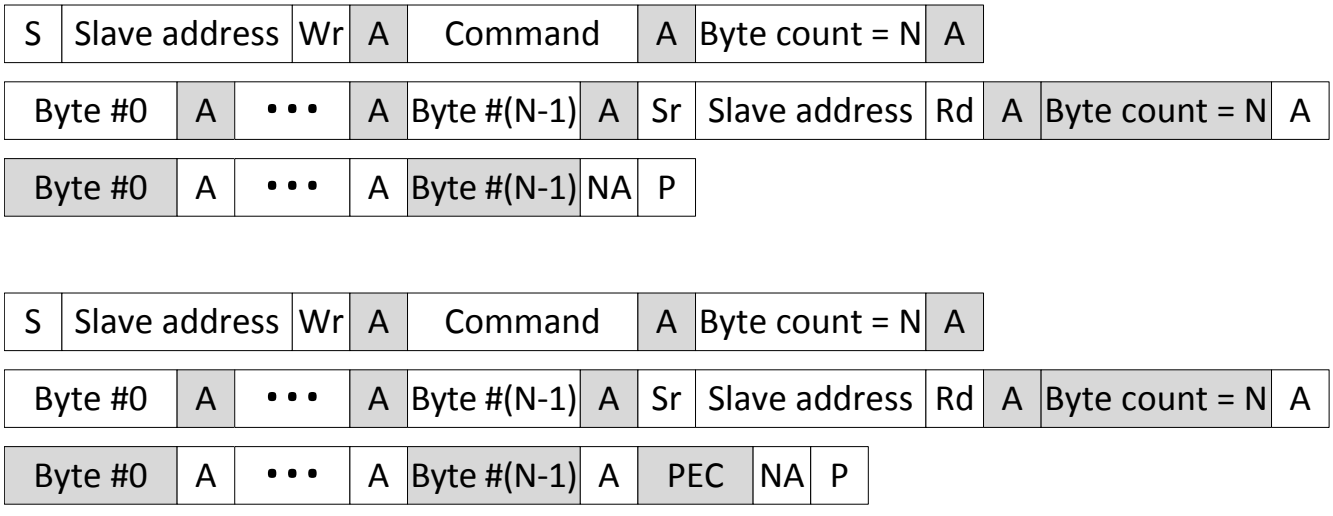

The Block Read protocol is similar to a Read Word in its structure, with the exception of reception of more than 2 data bytes from the slave. The first data byte transmitted by the slave represents the block length of the data being written by the slave. If PEC processing is enabled, the slave appends a PEC byte to the end of the message.

To initiate a Block Read message on the PMBus, the PMBMC register is programmed with the block length in the Byte Count bits. This count excludes the command byte, any slave address and the block length bytes in the message. The command byte to be transmitted to the slave is written into bits 7-0 of the PMBTXBUF register prior to the programming of the PMBMC register.

After configuring the PMBMC register, the Block Read message is transmitted. The module interrupts the firmware upon receipt of 4 data bytes from the slave. If the block length is 3, the EOM interrupt will be received concurrently with the data ready interrupt. Otherwise, only a data ready interrupt is asserted, indicating 4 bytes are ready for reading by the firmware. At the end of the message, less than 4 bytes may be stored in the PMBRXBUF register. The RX Byte Count bits in the PMBSTS register indicate the number of bytes available in the final data transfer. The firmware may verify the received PEC upon detection of the End of Message interrupt.

### 26.4.2.9 Block Write-Block Read Process Call

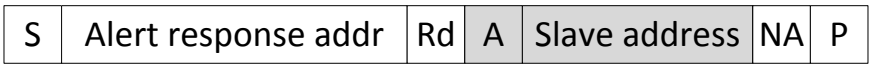
**Figure 26-23. Block Write-Block Read Process Call Message with and without PEC**



The Block Read-Block Write Process Call protocol combines the Block Write and Block Read protocols, removing the stop condition between the two messages. The operation of the master is similar to a Block Write operation. Loading the block length into the byte count bits of the PMBMC register provides the length of the Block Write portion of the message. In addition, the PRC_CALL bit within the PMBMC register must be enabled. Upon completion of the Block Write part of the message, the PMBus module will automatically issue a Repeated Start condition on the PMBus and start transmission of the Block Read portion of the message. Operation of the PMBus module after the Repeated Start condition is the same as it would be in a simple Block Read Message.

#### 26.4.2.10 Alert Response

**Figure 26-24. Alert Response Message**




The Alert Response Message is utilized when the master detects an alert condition from a slave. In master mode, the Alert Response Message is simply a Receive Byte message with PEC disable and the slave address set to 0xC (Alert Response Address). The PMBus module detects the alert condition on an input and interrupts the firmware indicating the assertion of an alert condition (slave desires to communicate with master). Programming the PMBMC register with the Alert Response Address initiates the Alert Response message and provides the device address of the slave requesting service. The device address will be found in the PMBRXBUF register following receipt of the EOM interrupt.

#### 26.4.2.11 Extended Command

**Figure 26-25. Extended Command Write Message with and without PEC**

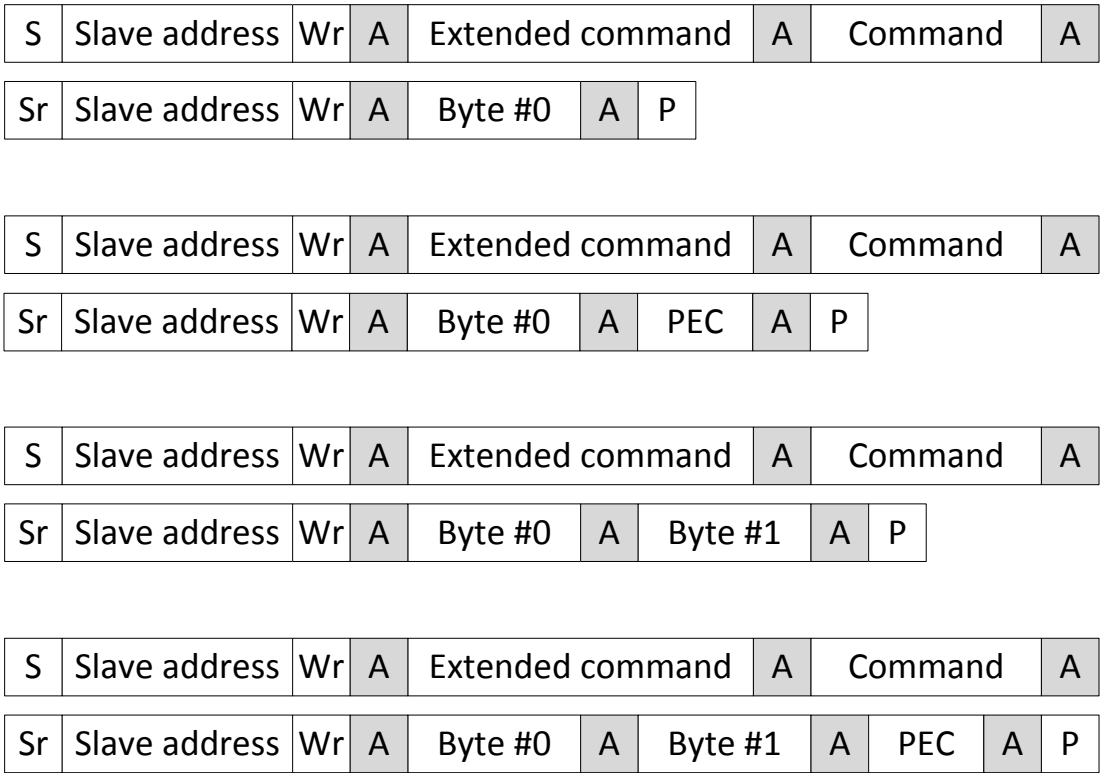
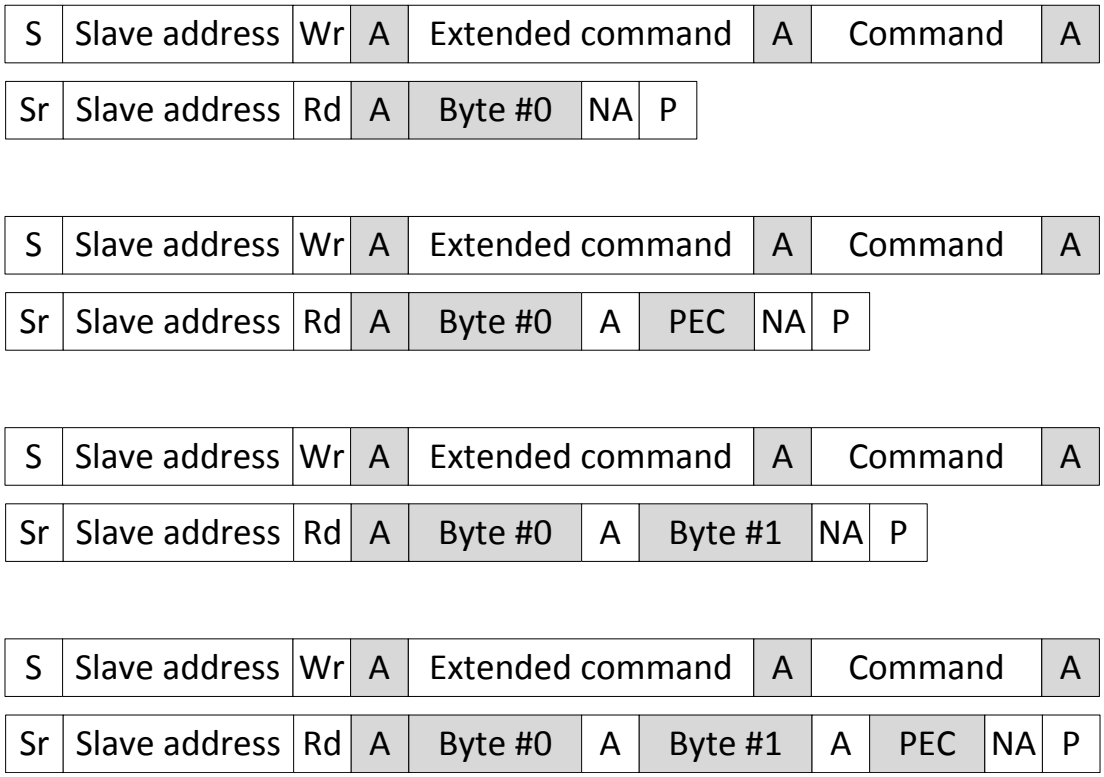


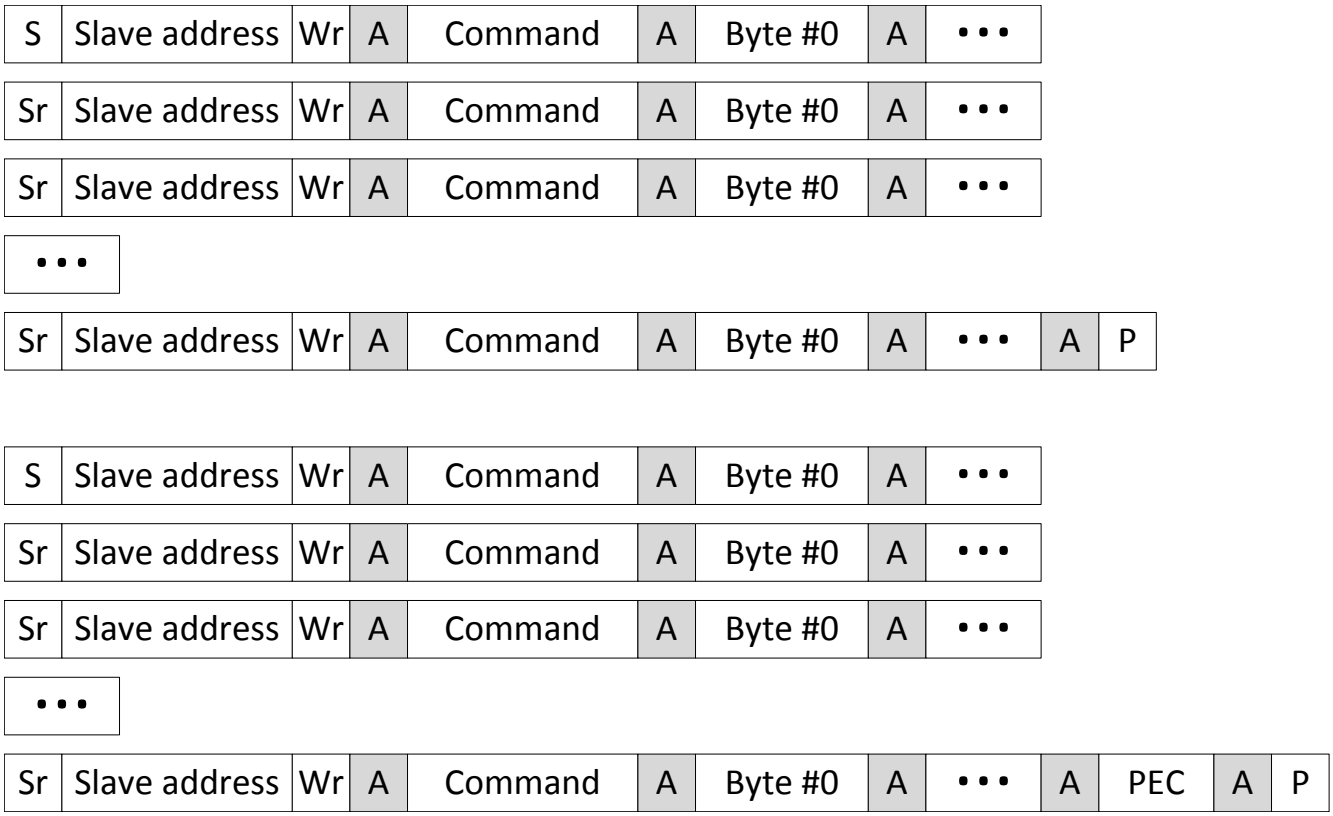
**Figure 26-26. Extended Command Read Message with and without PEC**




The PMBus module provides support for extended commands which allow for an extra 256 command codes. By asserting the EXT_CMD bit within the PMBMC register, two command bytes are transmitted on the message protocol. Extended commands can be added to the Read Byte, Read Word, Write Byte and Write Word protocols. Operation of the PMBus module in extended command mode is similar to these formats. In programming the write data or first part of the read message, the second command byte is loaded into Bits 15-8 of the PMBTXBUF register with the remaining data bytes. The remaining operation of the module is identical to the previous protocols, except for the inclusion of a Repeated Start condition and slave address in the write messages. No support is required by firmware for these additional bytes in the write messages. The module will interpret the EXT_CMD bit and make the appropriate format changes.

### 26.4.2.12 Group Command

**Figure 26-27. Group Command Message with and without PEC**




The Group Command Protocol is used to send commands to more than one device within the same message. When devices on the bus detect the stop condition at the conclusion of the Group Command message, the received commands are executed concurrently. To initiate a Group Command, the GRP_CMD bit within the PMBMC register must be set when programming the slave address for the first device in the message. The rest of the message is processed as a write byte/word message. At the conclusion of the first part of the Group Command message, the firmware programs the next device address in the PMBMC register. The PMBus module will send a repeated start on the bus and begin the next part of the message. When programming the last device address of the Group Command message, the firmware must disable the GRP_CMD bit when programming the PMBMC register.

## 26.5 Registers

### 26.5.1 Power Management Bus Base Addresses

**Table 26-1. PMBus Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| PmbusaRegs | PMBUS_REGS | 0x0000_6400 | 0x0000_641F |

### 26.5.1.1 PMBUS_REGS Registers

Table 26-2 lists the memory-mapped registers for the PMBUS_REGS. All register offset addresses not listed in Table 26-2 should be considered as reserved locations and the register contents should not be modified.

**Table 26-2. PMBUS_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | PMBMC | PMBUS Master Mode Control Register | EALLOW | Go |
| 2h | PMBTXBUF | PMBUS Transmit Buffer | | Go |
| 4h | PMBRXBUF | PMBUS Receive buffer | | Go |
| 6h | PMBACK | PMBUS Acknowledge Register | | Go |
| 8h | PMBSTS | PMBUS Status Register | | Go |
| Ah | PMBINTM | PMBUS Interrupt Mask Register | EALLOW | Go |
| Ch | PMBSC | PMBUS Slave Mode Configuration Register | EALLOW | Go |
| Eh | PMBHSA | PMBUS Hold Slave Address Register | | Go |
| 10h | PMBCTRL | PMBUS Control Register | EALLOW | Go |
| 12h | PMBTIMCTL | PMBUS Timing Control Register | EALLOW | Go |
| 14h | PMBTIMCLK | PMBUS Clock Timing Register | EALLOW | Go |
| 16h | PMBTIMSTSETUP | PMBUS Start Setup Time Register | EALLOW | Go |
| 18h | PMBTIMBIDLE | PMBUS Bus Idle Time Register | EALLOW | Go |
| 1Ah | PMBTIMLOWTIMOUT | PMBUS Clock Low Timeout Value Register | EALLOW | Go |
| 1Ch | PMBTIMHIGHTIMOUT | PMBUS Clock High Timeout Value Register | EALLOW | Go |

Complex bit access types are encoded to fit into small table cells. Table 26-3 shows the codes that are used for access types in this section.

**Table 26-3. PMBUS_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| RC | C<br>R | to Clear<br>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. |

#### 26.5.1.1.1 PMBMC Register (Offset = 0h) [reset = 0h]

PMBMC is shown in Figure 26-28 and described in Table 26-4.

Return to Summary Table.

PMBUS Master Mode Control Register

**Figure 26-28. PMBMC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | PRC_CALL | GRP_CMD | PEC_ENA | EXT_CMD | CMD_ENA |
| R-0h | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| BYTE_COUNT | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SLAVE_ADDR | | | | | | | RW |
| R/W-0h | | | | | | | R/W-0h |

**Table 26-4. PMBMC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-21 | RESERVED | R | 0h | Reserved |
| 20 | PRC_CALL | R/W | 0h | 0 = Default state for all messages besides Process Call message<br>1 = Enables transmission of Process Call message<br>Reset type: SYSRSn |
| 19 | GRP_CMD | R/W | 0h | 0 = Default state for all messages besides Group Command message<br>1 = Enables transmission of Group Command message<br>Reset type: SYSRSn |
| 18 | PEC_ENA | R/W | 0h | 0 = Disables PEC processing<br>1 = Enables PEC byte transmission/reception<br>Reset type: SYSRSn |
| 17 | EXT_CMD | R/W | 0h | 0 = Use 1 byte for Command Code<br>1 = Use 2 bytes for Command Code<br>Reset type: SYSRSn |
| 16 | CMD_ENA | R/W | 0h | 0 = Disables use of command code on Master initiated messages (<br>1 = Enables use of command code on Master initiated messages<br>Reset type: SYSRSn |
| 15-8 | BYTE_COUNT | R/W | 0h | Indicates number of data bytes transmitted in current message. Byte count does not include any device addresses, command words or block lengths in block messages. In block messages, the PMBus Interface automatically inserts the block length into the message based on the byte count setting. The firmware only needs to load the address, command words and data to be transmitted. PMBus Interface supports byte writes up to 255<br>bytes.<br>Reset type: SYSRSn |
| 7-1 | SLAVE_ADDR | R/W | 0h | Specifies the address of the slave to which the current message is directed towards.<br>Reset type: SYSRSn |

**Table 26-4. PMBMC Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | RW | R/W | 0h | 0 = Message is a write transaction (data from Master to Slave)<br>1 = Message is a read transaction (data from Slave to Master)<br>Reset type: SYSRSn |

#### 26.5.1.1.2 PMBTXBUF Register (Offset = 2h) [reset = 0h]

PMBTXBUF is shown in Figure 26-29 and described in Table 26-5.

Return to Summary Table.

PMBUS Transmit Buffer

**Figure 26-29. PMBTXBUF Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TXDATA | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 26-5. PMBTXBUF Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | TXDATA | R/W | 0h | Bits 31-24: BYTE3 - Last data byte transmitted from Transmit Data Buffer<br>Bits 23-16: BYTE2 - Third data byte transmitted from Transmit Data Buffer<br>Bits 15-8: BYTE1 - Second data byte transmitted from Transmit Data Buffer<br>Bits 7-0: BYTE0 - First data byte transmitted from Transmit Data Buffer<br>Reset type: SYSRSn |

#### 26.5.1.1.3 PMBRXBUF Register (Offset = 4h) [reset = 0h]

PMBRXBUF is shown in Figure 26-30 and described in Table 26-6.

Return to Summary Table.

PMBUS Receive buffer

**Figure 26-30. PMBRXBUF Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RXDATA | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 26-6. PMBRXBUF Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | RXDATA | R | 0h | Bits 31-24: BYTE3 - Last data byte received in Receive Data Buffer<br>Bits 23-16: BYTE2 - Third data byte received in Receive Data Buffer<br>Bits 15-8: BYTE1 - Second data byte received in Receive Data Buffer<br>Bits 7-0: BYTE0 - First data byte received in Receive Data Buffer<br>Reset type: SYSRSn |

#### 26.5.1.1.4 PMBACK Register (Offset = 6h) [reset = 0h]

PMBACK is shown in Figure 26-31 and described in Table 26-7.

Return to Summary Table.

PMBUS Acknowledge Register

**Figure 26-31. PMBACK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | | | | | | | | | ACK |
| R-0h | | | | | | | | | | | | | | | R/W-0h |

**Table 26-7. PMBACK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-1 | RESERVED | R | 0h | Reserved |
| 0 | ACK | R/W | 0h | 0 = NACK received data<br>1 = Acknowledge received data, bit clears upon issue of ACK on PMBus<br>Reset type: SYSRSn |

#### 26.5.1.1.5 PMBSTS Register (Offset = 8h) [reset = 00340000h]

PMBSTS is shown in Figure 26-32 and described in Table 26-8.

Return to Summary Table.

PMBUS Status Register

**Figure 26-32. PMBSTS Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | SCL_RAW | SDA_RAW | CONTROL_RAW | ALERT_RAW | CONTROL_EDGE | ALERT_EDGE |
| R-0h | | R-1h | R-1h | R-0h | R-1h | RC-0h | RC-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MASTER | LOST_ARB | BUS_FREE | UNIT_BUSY | RPT_START | SLAVE_ADDR_READY | CLK_HIGH_DETECTED | CLK_LOW_TIMEOUT |
| RC-0h | RC-0h | RC-0h | RC-0h | RC-0h | RC-0h | RC-0h | RC-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PEC_VALID | NACK | EOM | DATA_REQUEST | DATA_READY | RD_BYTE_COUNT | | |
| RC-0h | RC-0h | RC-0h | RC-0h | RC-0h | RC-0h | | |

**Table 26-8. PMBSTS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-22 | RESERVED | R | 0h | Reserved |
| 21 | SCL_RAW | R | 1h | 0 = PMBus clock pin observed at logic level low<br>1 = PMBus clock pin observed at logic level high<br>Reset type: SYSRSn |
| 20 | SDA_RAW | R | 1h | 0 = PMBus data pin observed at logic level low<br>1 = PMBus data pin observed at logic level high<br>Reset type: SYSRSn |
| 19 | CONTROL_RAW | R | 0h | 0 = Control pin observed at logic level low<br>1 = Control pin observed at logic level high<br>Reset type: SYSRSn |
| 18 | ALERT_RAW | R | 1h | 0 = Alert pin observed at logic level low<br>1 = Alert pin observed at logic level high<br>Reset type: SYSRSn |
| 17 | CONTROL_EDGE | RC | 0h | 0 = Control pin has not transitioned<br>1 = Control pin has been asserted by another device on PMBus<br>Reset type: SYSRSn |
| 16 | ALERT_EDGE | RC | 0h | 0 = Alert pin has not transitioned<br>1 = Alert pin has been asserted by another device on PMBus<br>Reset type: SYSRSn |
| 15 | MASTER | RC | 0h | 0 = PMBus Interface in Slave Mode or Idle Mode<br>1 = PMBus Interface in Master Mode<br>Reset type: SYSRSn |
| 14 | LOST_ARB | RC | 0h | 0 = Master has attained control of PMBus<br>1 = Master has lost arbitration and control of PMBus<br>Reset type: SYSRSn |

**Table 26-8. PMBSTS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 13 | BUS_FREE | RC | 0h | 0 = PMBus processing current message<br>1 = PMBus available for new message<br>Reset type: SYSRSn |
| 12 | UNIT_BUSY | RC | 0h | 0 = PMBus Interface is idle, ready to transmit/receive message<br>1 = PMBus Interface is busy, processing current message<br>Reset type: SYSRSn |
| 11 | RPT_START | RC | 0h | 0 = No Repeated Start received by interface<br>1 = Repeated Start condition received by interface<br>Reset type: SYSRSn |
| 10 | SLAVE_ADDR_READY | RC | 0h | 0 = Indicates no slave address is available for reading<br>1 = Slave address ready to be read from Receive Data Register (Bits 6:0)<br>Reset type: SYSRSn |
| 9 | CLK_HIGH_DETECTED | RC | 0h | 0 = No Clock High condition detected<br>1 = Clock High exceeded 50us during message<br>Reset type: SYSRSn |
| 8 | CLK_LOW_TIMEOUT | RC | 0h | 0 = No clock low timeout detected<br>1 = Clock low timeout detected, clock held low for greater than 35ms<br>Reset type: SYSRSn |
| 7 | PEC_VALID | RC | 0h | 0 = Received PEC not valid (if EOM is asserted)<br>1 = Received PEC is valid<br>Note: PEC_VALID status is don't care during the message. This will have a valid value only after EOM.<br>Reset type: SYSRSn |
| 6 | NACK | RC | 0h | 0 = Data transmitted has been accepted by receiver<br>1 = Receiver has not accepted transmitted data<br>Reset type: SYSRSn |
| 5 | EOM | RC | 0h | 0 = Message still in progress or PMBus in idle state.<br>1 = End of current message detected<br>Reset type: SYSRSn |
| 4 | DATA_REQUEST | RC | 0h | 0 = No data needed by PMBus Interface<br>1 = PMBus Interface request additional data. PMBus clock stretching enabled to stall bus<br>Reset type: SYSRSn |
| 3 | DATA_READY | RC | 0h | 0 = No data available for reading by processor<br>1 = PMBus Interface read buffer full, firmware required to read data prior to further bus<br>activity. PMBus clock stretching enabled to stall bus until data is read by firmware.<br>Reset type: SYSRSn |
| 2-0 | RD_BYTE_COUNT | RC | 0h | 0 = No received data<br>1 = 1 byte received. Data located in Receive Data Register, Bits 7-0<br>2 = 2 bytes received. Data located in Receive Data Register, Bits 15-0<br>3 = 3 bytes received. Data located in Receive Data Register, Bits 23-0<br>4 = 4 bytes received. Data located in Receive Data Register, Bits 31-0<br>Reset type: SYSRSn |

#### 26.5.1.1.6 PMBINTM Register (Offset = Ah) [reset = 3FFh]

PMBINTM is shown in Figure 26-33 and described in Table 26-9.

Return to Summary Table.

PMBUS Interrupt Mask Register

**Figure 26-33. PMBINTM Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | CLK_HIGH_DETECT | LOST_ARB |
| R-0h | | | | | | R/W-1h | R/W-1h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CONTROL | ALERT | EOM | SLAVE_ADDR_READY | DATA_REQUEST | DATA_READY | BUS_LOW_TIMEOUT | BUS_FREE |
| R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h | R/W-1h |

**Table 26-9. PMBINTM Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-10 | RESERVED | R | 0h | Reserved |
| 9 | CLK_HIGH_DETECT | R/W | 1h | 0 = Generates interrupt if clock high exceeds 50us during message<br>1 = Disables interrupt generation for Clock High detection<br>Reset type: SYSRSn |
| 8 | LOST_ARB | R/W | 1h | 0 = Generates interrupt upon assertion of Lost Arbitration flag<br>1 = Disables interrupt generation upon assertion of Lost Arbitration flag<br>Reset type: SYSRSn |
| 7 | CONTROL | R/W | 1h | 0 = Generates interrupt upon assertion of Control flag<br>1 = Disables interrupt generation upon assertion of Control flag<br>Reset type: SYSRSn |
| 6 | ALERT | R/W | 1h | 0 = Generates interrupt upon assertion of Alert flag<br>1 = Disables interrupt generation upon assertion of Alert flag<br>Reset type: SYSRSn |
| 5 | EOM | R/W | 1h | 0 = Generates interrupt upon assertion of End of Message flag<br>1 = Disables interrupt generation upon assertion of End of Message flag<br>Reset type: SYSRSn |
| 4 | SLAVE_ADDR_READY | R/W | 1h | 0 = Generates interrupt upon assertion of Slave Address Ready flag<br>1 = Disables interrupt generation upon assertion of Slave Address Ready flag<br>Reset type: SYSRSn |
| 3 | DATA_REQUEST | R/W | 1h | 0 = Generates interrupt upon assertion of Data Request flag<br>1 = Disables interrupt generation upon assertion of Data Request flag<br>Reset type: SYSRSn |

**Table 26-9. PMBINTM Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | DATA_READY | R/W | 1h | 0 = Generates interrupt upon assertion of Data Ready flag<br>1 = Disables interrupt generation upon assertion of Data Ready flag<br>Reset type: SYSRSn |
| 1 | BUS_LOW_TIMEOUT | R/W | 1h | 0 = Generates interrupt upon assertion of Clock Low Timeout flag<br>1 = Disables interrupt generation upon assertion of Clock Low Timeout flag<br>Reset type: SYSRSn |
| 0 | BUS_FREE | R/W | 1h | 0 = Generates interrupt upon assertion of Bus Free flag<br>1 = Disables interrupt generation upon assertion of Bus Free flag<br>Reset type: SYSRSn |

#### 26.5.1.1.7 PMBSC Register (Offset = Ch) [reset = 00607F7Ch]

PMBSC is shown in Figure 26-34 and described in Table 26-10.

Return to Summary Table.

PMBUS Slave Mode Configuration Register

**Figure 26-34. PMBSC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | RX_BYTE_ACK_CNT | | MAN_CMD | TX_PEC | TX_COUNT | | |
| R-0h | R/W-3h | | R/W-0h | R/W-0h | R/W-0h | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PEC_ENA | SLAVE_MASK | | | | | | |
| R/W-0h | R/W-7Fh | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MAN_SLAVE_ACK | SLAVE_ADDR | | | | | | |
| R/W-0h | R/W-7Ch | | | | | | |

**Table 26-10. PMBSC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-23 | RESERVED | R | 0h | Reserved |
| 22-21 | RX_BYTE_ACK_CNT | R/W | 3h | Configures number of data bytes to automatically<br>acknowledge when receiving data in slave mode.<br>00 = 1 byte received by slave. Firmware is required to manually acknowledge every received byte.<br>01 = 2 bytes received by slave. Hardware automatically acknowledges the first received<br>byte. Firmware is required to manually acknowledge after the second received byte.<br>10 = 3 bytes received by slave. Hardware automatically acknowledges the first 2 received<br>bytes. Firmware is required to manually acknowledge after the third received byte.<br>11 = 4 bytes received by slave. Hardware automatically acknowledges the first 3 received<br>bytes. Firmware is required to manually acknowledge after the fourth received byte<br>Reset type: SYSRSn |
| 20 | MAN_CMD | R/W | 0h | 0 = Slave automatically acknowledges received command code<br>1 = Data Request flag generated after receipt of command code, firmware required to<br>issue ACK to continue message<br>Reset type: SYSRSn |
| 19 | TX_PEC | R/W | 0h | Asserted when the slave needs to send a PEC byte at end of message.<br>PMBus Interface will transmit the calculated PEC byte after transmitting the number of data bytes<br>indicated by TX Byte Cnt(Bits 18:16).<br>Reset type: SYSRSn |

**Table 26-10. PMBSC Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18-16 | TX_COUNT | R/W | 0h | 0 = No bytes valid<br>1 = One byte valid, Byte #0 (Bits 7:0 of Transmit Data Register)<br>2 = Two bytes valid, Bytes #0 and #1 (Bits 15:0 of Transmit Data Register)<br>3 = Three bytes valid, Bytes #0-2 (Bits 23:0 of Transmit Data Register)<br>4 = Four bytes valid, Bytes #0-3 (Bits 31:0 of Transmit Data Register)<br>Reset type: SYSRSn |
| 15 | PEC_ENA | R/W | 0h | 0 = PEC processing disabled<br>1 = PEC processing enabled<br>Reset type: SYSRSn |
| 14-8 | SLAVE_MASK | R/W | 7Fh | Used in address detection, the slave mask enables acknowledgement<br>of multiple device addresses by the slave. Writing a '0' to a bit within the slave mask enables the<br>corresponding bit in the slave address to be either '1' or '0' and still allow for a match. Writing a '0'<br>to all bits in the mask enables the PMBus Interface to acknowledge any device address. Upon<br>power-up, the slave mask defaults to 7Fh, indicating the slave will only acknowledge the address<br>programmed into the Slave Address (Bits 6-0).<br>Reset type: SYSRSn |
| 7 | MAN_SLAVE_ACK | R/W | 0h | 0 = Slave automatically acknowledges device address specified in SLAVE_ADDR, Bits 6:0<br>1 = Enables the Manual Slave Address Acknowledgement Mode. Firmware is required to<br>read received address and acknowledge on every message<br>Note:<br>When bit 31 (I2C_mode) of PMBCTRL register is set it is recommended to use manual acknowledging of slave address only (MAN_SLAVE_ACK =1).<br>Reset type: SYSRSn |
| 6-0 | SLAVE_ADDR | R/W | 7Ch | Configures the current device address of the slave. Used in automatic slave address acknowledge mode (default mode).<br>The PMBus Interface will compare the received device address with the value stored in the Slave Address bits and the mask<br>configured in the Slave Mask bits. If matching, the slave will acknowledge the device address.<br>Reset type: SYSRSn |

#### 26.5.1.1.8 PMBHSA Register (Offset = Eh) [reset = 0h]

PMBHSA is shown in Figure 26-35 and described in Table 26-11.

Return to Summary Table.

PMBUS Hold Slave Address Register

**Figure 26-35. PMBHSA Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SLAVE_ADDR | | | | | | | SLAVE_RW |
| R-0h | | | | | | | R-0h |

**Table 26-11. PMBHSA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-8 | RESERVED | R | 0h | Reserved |
| 7-1 | SLAVE_ADDR | R | 0h | Stored device address acknowledged by the slave<br>Reset type: SYSRSn |
| 0 | SLAVE_RW | R | 0h | Stored R/W bit from address acknowledged by the slave<br>0 = Write Access<br>1 = Read Access<br>Reset type: SYSRSn |

#### 26.5.1.1.9 PMBCTRL Register (Offset = 10h) [reset = 00200000h]

PMBCTRL is shown in Figure 26-36 and described in Table 26-12.

Return to Summary Table.

PMBUS Control Register

**Figure 26-36. PMBCTRL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| I2CMODE | RESERVED | | | CLKDIV | | | |
| R/W-0h | R-0h | | | R/W-0h | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CLKDIV | MASTER_EN | SLAVE_EN | CLK_LO_DIS | IBIAS_B_EN | IBIAS_A_EN | SCL_DIR | SCL_VALUE |
| R/W-0h | R/W-0h | R/W-1h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SCL_MODE | SDA_DIR | SDA_VALUE | SDA_MODE | CNTL_DIR | CNTL_VALUE | CNTL_MODE | ALERT_DIR |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ALERT_VALUE | ALERT_MODE | CNTL_INT_EDGE | FAST_MODE_PLUS | FAST_MODE | BUS_LO_INT_EDGE | ALERT_EN | RESET |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 26-12. PMBCTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | I2CMODE | R/W | 0h | 0 = PMBUS mode<br>1 = I2C mode<br>Reset type: SYSRSn |
| 30-28 | RESERVED | R | 0h | Reserved |
| 27-23 | CLKDIV | R/W | 0h | The clock to the PMBUS transmit/receive FSMs (FSM_CLK) is divided version of the input IP clock. The IP clock can range anywhere from 10 Mhz to 300 Mhz. However the clock to the FSMs should not exceed 10Mhz (all the timing parameters are tuned for this frequency).<br>Frequency(FSM_CLK) = Frequency(IP_CLK)/(CLKDIV+1)<br>Reset type: SYSRSn |
| 22 | MASTER_EN | R/W | 0h | 0 = Disables PMBus Master capability<br>1 = Enables PMBus Master capability<br>Reset type: SYSRSn |
| 21 | SLAVE_EN | R/W | 1h | 0 = Disables PMBus Slave capability<br>1 = Enables PMBus Slave capability<br>Reset type: SYSRSn |
| 20 | CLK_LO_DIS | R/W | 0h | 0 = Clock Low Timeout Enabled<br>1 = Clock Low Timeout Disabled<br>Reset type: SYSRSn |
| 19 | IBIAS_B_EN | R/W | 0h | 0 = Disables Current Source for PMBUS address detection thru ADC<br>1 = Enables Current Source for PMBUS address detection thru ADC<br>Reset type: SYSRSn |
| 18 | IBIAS_A_EN | R/W | 0h | 0 = Disables Current Source for PMBUS address detection thru ADC<br>1 = Enables Current Source for PMBUS address detection thru ADC<br>Reset type: SYSRSn |

**Table 26-12. PMBCTRL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 17 | SCL_DIR | R/W | 0h | 0 = PMBus clock pin configured as output<br>1 = PMBus clock pin configured as input<br>Reset type: SYSRSn |
| 16 | SCL_VALUE | R/W | 0h | 0 = PMBus clock pin driven low in GPIO Mode<br>1 = PMBus clock pin driven high in GPIO Mode<br>Reset type: SYSRSn |
| 15 | SCL_MODE | R/W | 0h | 0 = PMBus clock pin configured in functional mode<br>1 = PMBus clock pin configured as GPIO<br>Reset type: SYSRSn |
| 14 | SDA_DIR | R/W | 0h | 0 = PMBus data pin configured as output<br>1 = PMBus data pin configured as input<br>Reset type: SYSRSn |
| 13 | SDA_VALUE | R/W | 0h | 0 = PMBus data pin driven low in GPIO Mode<br>1 = PMBus data pin driven high in GPIO Mode<br>Reset type: SYSRSn |
| 12 | SDA_MODE | R/W | 0h | 0 = PMBus data pin driven low in GPIO Mode<br>1 = PMBus data pin driven high in GPIO Mode<br>Reset type: SYSRSn |
| 11 | CNTL_DIR | R/W | 0h | 0 = Control pin configured as output<br>1 = Control pin configured as input<br>Reset type: SYSRSn |
| 10 | CNTL_VALUE | R/W | 0h | 0 = Control pin driven low in GPIO Mode<br>1 = Control pin driven high in GPIO Mode<br>Reset type: SYSRSn |
| 9 | CNTL_MODE | R/W | 0h | 0 = Control pin configured in functional mode (Default)<br>1 = Control pin configured as GPIO<br>Reset type: SYSRSn |
| 8 | ALERT_DIR | R/W | 0h | 0 = Alert pin configured as output<br>1 = Alert pin configured as input<br>Reset type: SYSRSn |
| 7 | ALERT_VALUE | R/W | 0h | 0 = Alert pin driven low in GPIO Mode<br>1 = Alert pin driven high in GPIO Mode<br>Reset type: SYSRSn |
| 6 | ALERT_MODE | R/W | 0h | 0 = Alert pin configured in functional mode<br>1 = Aler3 pin configured as GPIO<br>Reset type: SYSRSn |
| 5 | CNTL_INT_EDGE | R/W | 0h | 0 = Interrupt generated on falling edge of Control<br>1 = Interrupt generated on rising edge of Control<br>Reset type: SYSRSn |
| 4 | FAST_MODE_PLUS | R/W | 0h | 0 = Standard 100 KHz mode enabled<br>1 = Fast Mode Plus enabled (1MHz operation on PMBus)<br>Reset type: SYSRSn |
| 3 | FAST_MODE | R/W | 0h | 0 = Standard 100 KHz mode enabled<br>1 = Fast Mode enabled (400KHz operation on PMBus)<br>Reset type: SYSRSn |
| 2 | BUS_LO_INT_EDGE | R/W | 0h | 0 = Interrupt generated on rising edge of clock low timeout<br>1 = Interrupt generated on falling edge of clock low timeout<br>Reset type: SYSRSn |

**Table 26-12. PMBCTRL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | ALERT_EN | R/W | 0h | 0 = PMBus Alert is not driven by slave, pulled up high on PMBus<br>1 = PMBus Alert driven low by slave<br>Reset type: SYSRSn |
| 0 | RESET | R/W | 0h | 0 = No reset of internal state machines (Default)<br>1 = Control state machines are reset to initial states<br>Note: Status register PMBSTS should be explicitly cleared by reading the register after softrest as this will not be cleared by Software Reset.<br>Reset type: SYSRSn |

#### 26.5.1.1.10 PMBTIMCTL Register (Offset = 12h) [reset = 0h]

PMBTIMCTL is shown in Figure 26-37 and described in Table 26-13.

Return to Summary Table.

PMBUS Timing Control Register

**Figure 26-37. PMBTIMCTL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | TIM_OVERRIDE |
| R-0h | | | | | | | R/W-0h |

**Table 26-13. PMBTIMCTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-1 | RESERVED | R | 0h | Reserved |
| 0 | TIM_OVERRIDE | R/W | 0h | 0 PMBUS FSMs uses the default settings of the timing parameters.<br>1 PMBUS FSMs would use the settings in following registers:<br>* PMBTIMCLK<br>* PMBTIMSTSETUP<br>* PMBTIMBIDLE<br>* PMBTIMLOWTIMOUT<br>* PMBTIMHIGHTIMOUT<br>Reset type: SYSRSn |

#### 26.5.1.1.11 PMBTIMCLK Register (Offset = 14h) [reset = 0060002Fh]

PMBTIMCLK is shown in Figure 26-38 and described in Table 26-14.

Return to Summary Table.

PMBUS Clock Timing Register

**Figure 26-38. PMBTIMCLK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | CLK_FREQ | | | | | | | |
| R-0h | | | | | | | | R/W-60h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | | CLK_HIGH_LIMIT | | | | | | | |
| R-0h | | | | | | | | R/W-2Fh | | | | | | | |

**Table 26-14. PMBTIMCLK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | RESERVED | R | 0h | Reserved |
| 23-16 | CLK_FREQ | R/W | 60h | Defines the number of PMBUS FSM input clock in the PMBUS master clock period.<br>Number of FSM clocks in the one clock period = (CLK_HIGH_LIMIT+4)<br>Reset type: SYSRSn |
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-0 | CLK_HIGH_LIMIT | R/W | 2Fh | Defines the number of PMBUS FSM input clock in the PMBUS master clock high pulse.<br>Number of FSM clocks in the one clock high pulse = (CLK_HIGH_LIMIT+3)<br>Reset type: SYSRSn |

#### 26.5.1.1.12 PMBTIMSTSETUP Register (Offset = 16h) [reset = 2Fh]

PMBTIMSTSETUP is shown in Figure 26-39 and described in Table 26-15.

Return to Summary Table.

PMBUS Start Setup Time Register

**Figure 26-39. PMBTIMSTSETUP Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | | | | | | | | | TSU_STA | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | R/W-2Fh | | | | | | | |

**Table 26-15. PMBTIMSTSETUP Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-8 | RESERVED | R | 0h | Reserved |
| 7-0 | TSU_STA | R/W | 2Fh | Determines the Setup time between last rise edge of the PMBUS master clock and the next start edge, TSU_STA value defines the setup time in terms of PMBUS FSM clock cycles.<br>Reset type: SYSRSn |

#### 26.5.1.1.13 PMBTIMBIDLE Register (Offset = 18h) [reset = 1F3h]

PMBTIMBIDLE is shown in Figure 26-40 and described in Table 26-16.

Return to Summary Table.

PMBUS Bus Idle Time Register

**Figure 26-40. PMBTIMBIDLE Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | | | | | | | BUSIDLE | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | R/W-1F3h | | | | | | | | | |

**Table 26-16. PMBTIMBIDLE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-10 | RESERVED | R | 0h | Reserved |
| 9-0 | BUSIDLE | R/W | 1F3h | Determines the duration for which PMBUS clock and Data are 1 , to conclude that the bus is IDLE. BUSIDLE value is in terms of number of PMBUS FSM clock cycles.<br>Reset type: SYSRSn |

#### 26.5.1.1.14 PMBTIMLOWTIMOUT Register (Offset = 1Ah) [reset = 0005572Fh]

PMBTIMLOWTIMOUT is shown in Figure 26-41 and described in Table 26-17.

Return to Summary Table.

PMBUS Clock Low Timeout Value Register

**Figure 26-41. PMBTIMLOWTIMOUT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | CLKLOWTIMOUT | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | R/W-0005572Fh | | | | | | | | | | | | | | | | | | | |

**Table 26-17. PMBTIMLOWTIMOUT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-20 | RESERVED | R | 0h | Reserved |
| 19-0 | CLKLOWTIMOUT | R/W | 0005572Fh | Determines the duration for which PMBUS clock if low , will result in a clock low timeout condition. CLKLOWTIMOUT value is in terms of number of PMBUS FSM clock cycles.<br>Reset type: SYSRSn |

#### 26.5.1.1.15 PMBTIMHIGHTIMOUT Register (Offset = 1Ch) [reset = 1F3h]

PMBTIMHIGHTIMOUT is shown in Figure 26-42 and described in Table 26-18.

Return to Summary Table.

PMBUS Clock High Timeout Value Register

**Figure 26-42. PMBTIMHIGHTIMOUT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | CLKHIGHTIMOUT | | | | | | | | | |
| R-0h | | | | | | R/W-1F3h | | | | | | | | | |

**Table 26-18. PMBTIMHIGHTIMOUT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-10 | RESERVED | R | 0h | Reserved |
| 9-0 | CLKHIGHTIMOUT | R/W | 1F3h | Determines the duration for which PMBUS clock if high , will result in a clock high timeout condition. CLKHIGHTIMOUT value is in terms of number of PMBUS FSM clock cycles.<br>Reset type: SYSRSn |

# Controller Area Network (CAN)

This chapter contains a general description of the Controller Area Network (CAN) module. The CAN module is a serial communications protocol which efficiently supports distributed real-time control with a high level of reliability. The CAN module supports bit-rates up to 1 Mbit/s and is compliant with the ISO11898-1 (CAN 2.0B) protocol specification.

Further information about this device can be found in the following document(s):

- *Calculator for CAN Bit Timing Parameters*

**Topic** **Page**

## 27.1 Overview

This device uses the CAN IP known as D_CAN.

### 27.1.1 Features

The CAN module implements the following features:

- Complies with ISO11898-1 (Bosch® CAN protocol specification 2.0 A and B)
- Bit rates up to 1 MBit/s
- Multiple clock sources
- 32 message objects (mailboxes), 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
- Two interrupt lines
- DMA support

**NOTE:** For a CAN bit clock of 100 MHz, the smallest bit rate possible is 3.90625 kbps.

**NOTE:** Depending on the timing settings used, the accuracy of the on-chip zero-pin oscillator (specified in the datasheet) may not meet the requirements of the CAN protocol. In this situation, an external clock source must be used.

### 27.1.2 Functional Description

The CAN module performs CAN protocol communication according to ISO 11898-1 (identical to Bosch CAN protocol specification 2.0 A, B). The bit rate can be programmed to values up to 1 MBit/s. A CAN chip transceiver chip is required for the connection to the physical layer (CAN bus).

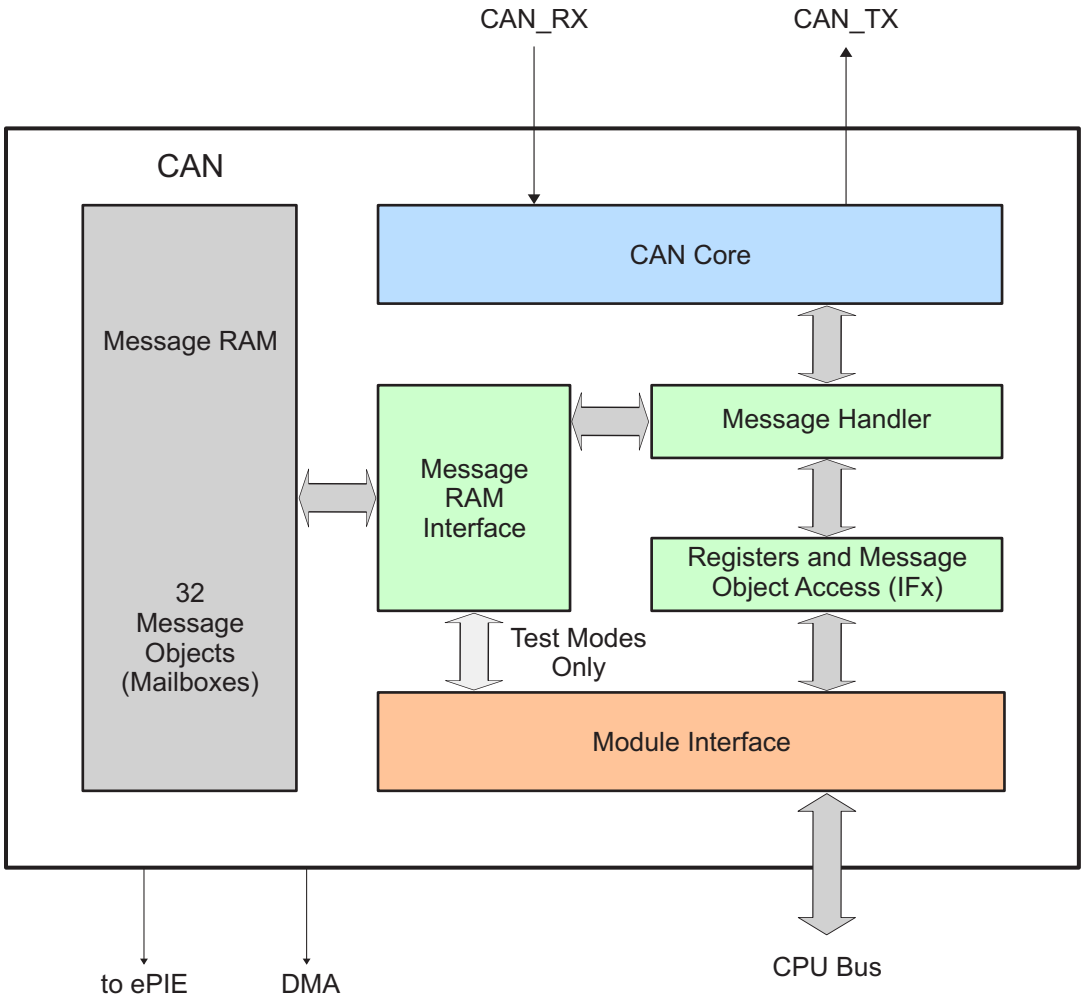
For communication on a CAN network, individual message objects can be configured. The message objects and identifier masks are stored in the Message RAM.

All functions concerning the handling of messages are implemented in the message handler. Those functions are acceptance filtering, the transfer of messages between the CAN Core and the Message RAM, and the handling of transmission requests as well as the generation of interrupts or DMA requests.

The register set of the CAN may be accessed directly by the CPU via the module interface. These registers are used to control/configure the CAN core and the message handler, and to access the message RAM.

### 27.1.3 Block Diagram

**Figure 27-1. CAN Block Diagram**




#### 27.1.3.1 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.

#### 27.1.3.2 Message Handler

The message handler is a state machine which controls the data transfer between the single ported Message RAM and the CAN Core's Rx/Tx Shift register. It also handles acceptance filtering and the Interrupt/DMA request generation as programmed in the control registers. It also handles acceptance filtering and the interrupt request generation as programmed in the control registers.

#### 27.1.3.3 Message RAM

The CAN message RAM enables the storage of 32 CAN messages.

#### 27.1.3.4 Registers and Message Object Access (IFx)

Data consistency is ensured by indirect accesses to the message objects. During normal operation, all CPU and DMA accesses to the message RAM are done through Interface registers.

Three Interface Register sets control the CPU read and write accesses to the Message RAM. There are two Interface register sets for read/write access (IF1 and IF2) and one Interface Register set for read access only (IF3). See also Section 27.14. The Interface registers have the same word-length as the message RAM.

In a dedicated test mode, the message RAM is memory mapped and can be directly accessed.

## 27.2 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

Some I/O functionality is defined by GPIO register settings independent of this peripheral. For input signals, the GPIO input qualification should be set to asynchronous mode by setting the appropriate GPxQSELn register bits to 11b. The internal pull-ups can be configured in the GPyPUD register.

See the *GPIO* chapter for more details on GPIO mux and settings.

## 27.3 Address/Data Bus Bridge

The CAN module uses a special addressing scheme to support byte accesses. It is recommended to only use 32-bit accesses to the CAN registers using the *HWREG_BP()* macro which uses the *__byte_peripheral_32()* intrinsic. If 16-bit accesses are to be used, the lower 16 bits should be written to the register's address, and the upper 16 bits should be written to the register's address plus 2.

Because of the bus bridge, the view of the CAN module's register space through a CCS memory window does not always match the actual addressing. When the view mode is 32-bit or 16-bit, even addresses are effectively duplicated. Odd addresses should be ignored. When the view mode is 8-bit, even addresses from within the CAN module are duplicated into the odd addresses in the CCS memory view. Odd addresses from the module are not displayed.

**Table 27-1. CAN Register Access From Software**

| CAN Register Space | | | C28x 8 Bit | |
|---|---|---|---|---|
| **Address** | **Reg. Name** | **Data** | **Access** | **Data** |
| 0x00 | CAN_CTL | 0x33221100 | __byte((int *)0x00,0) | 0x0000 |
| 0x04 | CAN_ES | 0x77665544 | __byte((int *)0x01,0) | 0x0011 |
| 0x08 | CAN_ERRC | 0xBBAA9988 | __byte((int *)0x02,0) | 0x0022 |
| 0x0C | CAN_BTR | 0xFFEEDDCC | __byte((int *)0x03,0) | 0x0033 |
| | | | __byte((int *)0x04,0) | 0x0044 |
| | | | __byte((int *)0x05,0) | 0x0055 |
| | | | __byte((int *)0x06,0) | 0x0066 |
| | | | __byte((int *)0x07,0) | 0x0077 |
| | | | __byte((int *)0x08,0) | 0x0088 |
| | | | __byte((int *)0x09,0) | 0x0099 |
| | | | __byte((int *)0x0A,0) | 0x00AA |
| | | | __byte((int *)0x0B,0) | 0x00BB |
| | | | __byte((int *)0x0C,0) | 0x00CC |
| | | | __byte((int *)0x0D,0) | 0x00DD |
| | | | __byte((int *)0x0E,0) | 0x00EE |
| | | | __byte((int *)0x0F,0) | 0x00FF |
| **C28x 16 Bit** | | | **C28x 32 Bit** | |
| **Access** | **Data** | | **Access** | **Data** |
| (*((short *)(0x00))) | 0x1100 | | (*((long *)(0x00))) | 0x33221100 |
| (*((short *)(0x01))) | 0x1100 | | (*((long *)(0x01))) | 0x33221100 |
| (*((short *)(0x02))) | 0x3322 | | (*((long *)(0x02))) | 0x33221100 |

**Table 27-1. CAN Register Access From Software (continued)**

| CAN Register Space | | | C28x 8 Bit | |
|---|---|---|---|---|
| (*((short *)(0x03))) | 0x3322 | | (*((long *)(0x03))) | 0x33221100 |
| (*((short *)(0x04))) | 0x5544 | | (*((long *)(0x04))) | 0x77665544 |
| (*((short *)(0x05))) | 0x5544 | | (*((long *)(0x05))) | 0x77665544 |
| (*((short *)(0x06))) | 0x7766 | | (*((long *)(0x06))) | 0x77665544 |
| (*((short *)(0x07))) | 0x7766 | | (*((long *)(0x07))) | 0x77665544 |
| (*((short *)(0x08))) | 0x9988 | | (*((long *)(0x08))) | 0xBBAA9988 |
| (*((short *)(0x09))) | 0x9988 | | (*((long *)(0x09))) | 0xBBAA9988 |
| (*((short *)(0x0A))) | 0xBBAA | | (*((long *)(0x0A))) | 0xBBAA9988 |
| (*((short *)(0x0B))) | 0xBBAA | | (*((long *)(0x0B))) | 0xBBAA9988 |
| (*((short *)(0x0C))) | 0xDDCC | | (*((long *)(0x0C))) | 0xFFEEDDCC |
| (*((short *)(0x0D))) | 0xDDCC | | (*((long *)(0x0D))) | 0xFFEEDDCC |
| (*((short *)(0x0E))) | 0xFFEE | | (*((long *)(0x0E))) | 0xFFEEDDCC |
| (*((short *)(0x0F))) | 0xFFEE | | (*((long *)(0x0F))) | 0xFFEEDDCC |

**Table 27-2. CAN Register Access From CCS**

| CCS 8 Bit | | CCS 16 Bit | | CCS 32 Bit | |
|---|---|---|---|---|---|
| **Address** | **Displayed Data** | **Address** | **Displayed Data** | **Address** | **Displayed Data** |
| 0x00 | 0x00 | 0x00 | 0x1100 | 0x00 | 0x11001100 |
| 0x01 | 0x00 | 0x01 | 0x1100 | 0x02 | 0x33223322 |
| 0x02 | 0x22 | 0x02 | 0x3322 | 0x04 | 0x55445544 |
| 0x03 | 0x22 | 0x03 | 0x3322 | 0x06 | 0x77667766 |
| 0x04 | 0x44 | 0x04 | 0x5544 | 0x08 | 0x99889988 |
| 0x05 | 0x44 | 0x05 | 0x5544 | 0x0A | 0xBBAABBAA |
| 0x06 | 0x66 | 0x06 | 0x7766 | 0x0C | 0xDDCCDDCC |
| 0x07 | 0x66 | 0x07 | 0x7766 | 0x0E | 0xFFEEFFEE |
| 0x08 | 0x88 | 0x08 | 0x9988 | | |
| 0x09 | 0x88 | 0x09 | 0x9988 | | |
| 0x0A | 0xAA | 0x0A | 0xBBAA | | |
| 0x0B | 0xAA | 0x0B | 0xBBAA | | |
| 0x0C | 0xCC | 0x0C | 0xDDCC | | |
| 0x0D | 0xCC | 0x0D | 0xDDCC | | |
| 0x0E | 0xEE | 0x0E | 0xFFEE | | |
| 0x0F | 0xEE | 0x0F | 0xFFEE | | |

## 27.4 Operating Modes

### *27.4.1 Initialization*

The initialization mode is entered either by software (by setting the **Init** bit in the CAN_CTL register), by hardware reset, or by going bus-off. While the Init bit is set, the message transfer from and to the CAN bus is stopped, and the status of the CAN_TX output is recessive (high). The CAN error counters are not updated. Setting the Init bit does not change any other configuration register.

To initialize the CAN Controller, the CPU has to configure the CAN Bit timing and those message objects which have to be used for CAN communication. Message objects which are not needed, can be deactivated with their MsgVal bits cleared.

The access to the Bit Timing Register for the configuration of the bit timing is enabled when both **Init** and CCE bits in the CAN Control register are set.

Clearing the Init bit finishes the software initialization. Afterwards the bit stream processor (BSP), synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (= Bus Idle) before it can take part in bus activities and start the message transfer (for more details see Section 27.13) .

The initialization of the message objects is independent of the Init bit, however all message objects should be configured to particular identifiers or set to "not-valid" before the message transfer is started.

It is possible to change the configuration of message objects during normal operation by the CPU. After setup and subsequent transfer of message object from interface registers to message RAM, the acceptance filtering will be applied to it, when the modified message object number is same or smaller than the previously found message object. This assures data consistency even when changing message objects, for example, while there is a pending CAN frame reception.

### *27.4.2 CAN Message Transfer (Normal Operation)*

Once the CAN is initialized and the **Init** bit is reset to zero, the CAN Core synchronizes itself to the CAN bus and is ready for communication.

Received messages are stored into their appropriate message objects if they pass acceptance filtering. The whole message (MSGID, DLC, and up to eight data bytes) is stored into the message object. As a consequence, when, for example, the identifier mask is used, the MSGID bits which are masked to "don't care" may change in the message object when a received message is stored.

The CPU may read or write each message at any time via the Interface registers, as the message handler guarantees data consistency in case of concurrent accesses.

Messages to be transmitted can be updated by the CPU. If a permanent message object (MSGID and control bits set up during configuration and leaving unchanged for multiple CAN transfers) exists for the message, it is possible to only update the data bytes. If several transmit messages should be assigned to one message object, the whole message object has to be configured before the transmission of this message is requested.

The transmission of multiple message objects may be requested at the same time. They are subsequently transmitted, according to their internal priority. Messages may be updated or set to not valid at any time, even if a requested transmission is still pending. However, the data bytes will be discarded if a message is updated before a pending transmission has started.

Depending on the configuration of the message object, a transmission may be automatically requested by the reception of a remote frame with a matching identifier.

#### 27.4.2.1 Disabled Automatic Retransmission

According to the CAN Specification (see ISO11898, 6.3.3 Recovery Management), the CAN provides a mechanism to automatically retransmit frames which have lost arbitration or have been disturbed by errors during transmission. The frame transmission service will not be confirmed to the user before the transmission is successfully completed.

By default, this automatic retransmission is enabled. It can be disabled by setting bit DAR in the CAN Control register. Further details to this mode are provided in Section 27.12.3.

#### 27.4.2.2 Auto-Bus-On

After the CAN has entered bus-off state, the CPU can start a bus-off-recovery sequence by resetting Init bit. If this is not done, the module will stay in bus-off state.

The CAN provides an automatic auto-bus-on feature which is enabled by bit ABO. If set, the CAN will automatically start the bus-off-recovery sequence. The sequence can be delayed by a user-defined number of clock cycles.

**NOTE:** If the CAN module goes Bus-Off due to massive occurrence of CAN bus errors, it stops all bus activities and automatically sets the Init bit. Once the Init bit is cleared by the application (or due to the auto-bus-on feature), the device will wait for 128 occurrences of Bus Idle (equal to 128 * 11 consecutive recessive bits) before resuming normal operation. The Bus-Off recovery sequence cannot be shortened by setting or resetting Init bit. At the end of the bus-off recovery sequence, the error counters will be reset. After the Init bit is reset, each time when a sequence of 11 recessive bits is monitored, a Bit0 Error code is written to the Error and Status Register, enabling 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.

### *27.4.3 Test Modes*

The CAN provides several test modes which are mainly intended for self test.

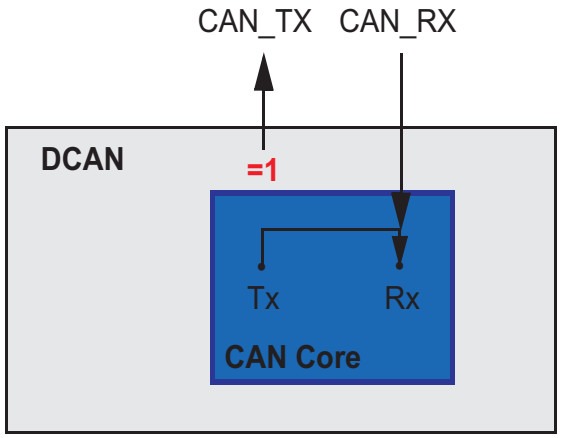
For all test modes, the **Test** bit in the CAN Control register needs to be set to 1. This enables write access to the Test register.

#### 27.4.3.1 Silent Mode

The silent mode may be used to analyze the traffic on the CAN bus without affecting it by sending dominant bits (for example, acknowledge bit, overload flag, active error flag). The CAN is still able to receive valid data frames and valid remote frames, but it will not send any dominant bits. However, the received frames se are internally routed to the CAN Core.

Figure 27-2 shows the connection of signals CAN_TX and CAN_RX to the CAN core in silent mode. Silent mode can be activated by setting the **Silent** bit in test register (CAN_TEST), to 1. In ISO 11898-1, the silent mode is called the bus monitoring mode.

**Figure 27-2. CAN Core in Silent Mode**




#### 27.4.3.2 Loopback Mode

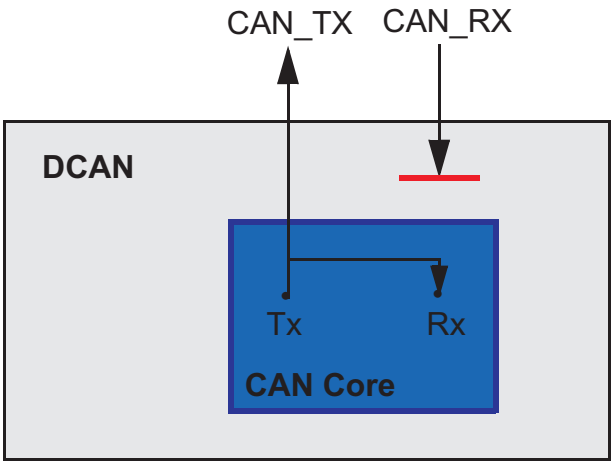
The loopback mode is mainly intended for hardware self-test functions. In this mode, the CAN core uses internal feedback from Tx output to Rx input. Transmitted messages are treated as received messages, and can be stored into message objects if they pass acceptance filtering. The actual value of the CAN_RX input pin is disregarded by the CAN core. Transmitted messages still can be monitored at the CAN_TX pin.

In order to be independent from external stimulation, the CAN core ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in loopback mode.

Figure 27-3 shows the connection of signals CAN_TX and CAN_RX to the CAN core in loopback mode. Loopback mode can be activated by setting bit **LBack** in the CAN_TEST register to 1.

**NOTE:** In loopback mode, the signal path from CAN core to Tx pin, the Tx pin itself, and the signal path from Tx pin back to CAN Core are disregarded. For including these into the testing, see Section 27.4.3.3.

**Figure 27-3. CAN Core in Loopback Mode**



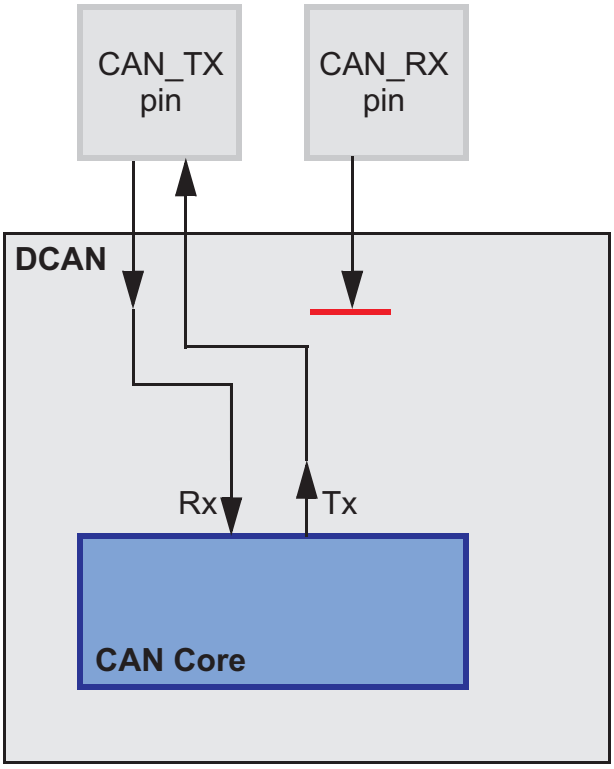

### 27.4.3.3 External Loopback Mode

The external loopback mode is similar to the loopback mode; however, it includes the signal path from CAN Core to Tx pin, the Tx pin itself, and the signal path from Tx pin back to the CAN Core. When the external loopback mode is selected, the CAN core is connected to the input buffer of the Tx pin. With this configuration, the Tx pin IO circuit can be tested. External loopback mode can be activated by setting bit **ExL** in Test Register to one.

Figure 27-4 shows the connection of signals CAN_TX and CAN_RX to the CAN Core in external loopback mode.

**NOTE:** When loopback mode is active (LBack bit set), the ExL bit will be ignored.

**Figure 27-4. CAN Core in External Loopback Mode**

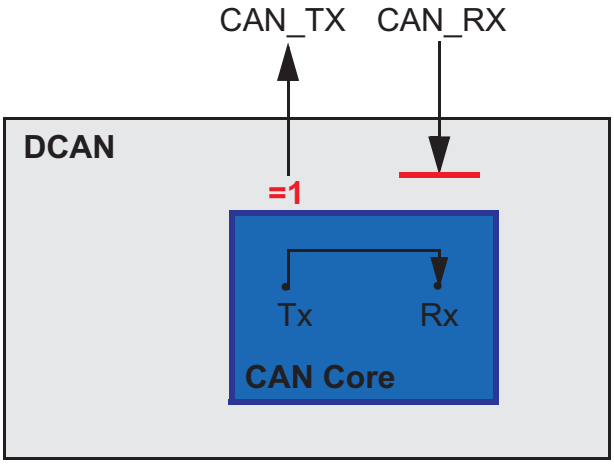



### 27.4.3.4 Loopback Combined with Silent Mode

It is also possible to combine loopback mode and silent mode by setting bits LBack and Silent at the same time. This mode can be used for a "Hot Selftest;" that is, the CAN hardware can be tested without affecting the CAN network. In this mode, the CAN_RX pin is disconnected from the CAN Core and no dominant bits will be sent on the CAN_TX pin.

Figure 27-5 shows the connection of the signals CAN_TX and CAN_RX to the CAN Core in case of the combination of loopback mode with silent mode.

**Figure 27-5. CAN Core in Loopback Combined with Silent Mode**



## 27.5 Multiple Clock Source

The CAN bit timing clock is normally derived from the system clock (SYSCLK). If desired, the bit clock can be derived directly from the external clock source (XTAL).

The System Control chapter of this document and the device data manual provide for more information on how to configure the relevant clock source registers in the system module.

**NOTE:** The CAN core has to be programmed to at least eight clock cycles per bit time. To achieve a transfer rate of 1 Mbps an oscillator frequency of 8 MHz or higher has to be used.

## 27.6 Interrupt Functionality

Interrupts can be generated on two interrupt lines: CAN0INT and CAN1INT. These lines can be enabled by setting the IE0 and IE1 bits, respectively, in the CAN Control register.

The CAN provides three groups of interrupt sources: message object interrupts, status change interrupts and error interrupts. The source of an interrupt can be determined by the interrupt identifiers Int0ID / Int1ID in the Interrupt register. When no interrupt is pending, the register will hold the value zero. Each interrupt line remains active until the dedicated field in the Interrupt register (Int0ID/Int1ID) again reach zero (this means the cause of the interrupt is reset), or until IE0 / IE1 are reset. The value 0x8000 in the Int0ID field indicates that an interrupt is pending because the CAN Core has updated (not necessarily changed) the Error and Status Register (Error Interrupt or Status Interrupt). This interrupt has the highest priority. The CPU can update (reset) the status bits RxOk, TxOk and LEC by reading the Error and Status Register, but a write access of the CPU will never generate or reset an interrupt.

Values between 1 and the number of the last message object indicates that the source of the interrupt is one of the message objects, Int0ID/Int1ID will point to the pending message interrupt with the highest priority. The Message Object 1 has the highest priority, the last message object has the lowest priority.

An interrupt service routine which reads the message that is the source of the interrupt, may read the message and reset the message object's IntPnd at the same time (ClrIntPnd bit in the IF1/IF2 Command register). When IntPnd is cleared, the Interrupt register will point to the next message object with a pending interrupt.

The CAN module features a module-level interrupt enable and acknowledge mechanism. To enable the CAN0 and CAN1 interrupts, set the appropriate bits in the CAN_GLB_INT_EN register. When handling an interrupt, the individual message or status change flag must be cleared prior to acknowledging the interrupt via CAN_GLB_INT_CLR and PIEACK.

### *27.6.1 Message Object Interrupts*

Message object interrupts are generated by events from the message objects. They are controlled by the flags IntPND, TxIE and RxIE which are described in Section 27.15.1. Message object interrupts can be routed to either CAN0INT or CAN1INT line, controlled by the Interrupt Multiplexer register.

### *27.6.2 Status Change Interrupts*

The events RxOk, TxOk and LEC in Error and Status register belong to the status change interrupts. The status change interrupt group can be enabled by bit SIE in CAN Control Register. If SIE is set, a status change interrupt will be generated at each CAN frame, independent of bus errors or valid CAN communication, and also independent of the Message RAM configuration. Status Change interrupts can only be routed to interrupt line CAN0INT which has to be enabled by setting IE0 in the CAN Control Register.

### *27.6.3 Error Interrupts*

The events PER, BOff and EWarn, belong to the error interrupts. The error interrupt group can be enabled by setting bit EIE. Also, error interrupts can only be routed to interrupt line CAN0INT which has to be enabled by setting IE0 in the CAN_CTL register.

### 27.6.4 PIE Nomenclature for DCAN Interrupts

Table 27-3 shows the PIE nomenclature for the interrupts.

**Table 27-3. PIE Nomenclature for Interrupts**

| Interrupt | CANA | CANB |
|---|---|---|
| CANINT0 | CANA_0 | CANB_0 |
| CANINT1 | CANA_1 | CANB_1 |

## 27.7 DMA Functionality

The CAN module provides three DMA trigger outputs, one for each of the three Interface Registers IF1, IF2 and IF3. These can be enabled using the DE1, DE2, and DE3 bits in the CAN_CTL register.

The Update of IF1 and IF2 registers will be initiated by a write access to the IF1 and IF2 Command Registers respectively. Once enabled, setting the DMAActive bit in the IF1CMD or IF2CMD registers will cause a DMA request the next time the corresponding interface becomes available.

The IF3 registers content can be automatically updated on reception of CAN messages in message objects which are programmed for automatic IF3 update. That is, when IF3 DMA requests are enabled, all IF3 updates will trigger a DMA request.

When a DCAN internal IFx update is complete, a DMA request will be activated and stays active until the first access to one of the relevant IFx registers; that is, DMA requests are cleared after the first read or write access to an IF register set.

## 27.8 Parity Check Mechanism

The CAN provides a parity check mechanism to ensure data integrity of message RAM data. For each word (32 bits) in Message RAM, one parity bit will be calculated.

Parity information is stored in the Message RAM on write accesses and will be checked against the stored parity bit from Message RAM on read accesses.

The parity check functionality can be enabled or disabled by PMD bit field in CAN Control register. In case of disabled parity check, the parity bits in message RAM will be left unchanged on write access to data area and no check will be done on read access.

If parity checking is enabled, parity bits will be automatically generated and checked by the CAN. A parity bit will be set, if the modulo-2-sum of the data bits is 1. This definition is equivalent to: the parity bit will be set, if the number of 1 bits in the data is odd.

### 27.8.1 Behavior on Parity Error

On any read access to Message RAM, for example, during the start of a CAN frame transmission, the parity of the message object will be checked. If a parity error is detected, the PER bit in Error and Status register will be set. If error interrupts are enabled, an interrupt would also be generated. In order to avoid the transmission of invalid data over the CAN bus, the MsgVal bit of the message object will be reset.

The message object data can be read by the CPU, independently of parity errors. Thus, the application has to ensure that the read data is valid, for example, by immediately checking the Parity Error Code register on parity error interrupt.

## 27.9 Debug Mode

The module supports the usage of an external debug unit by providing functions like pausing CAN activities and making message RAM content accessible from the debugger. Debug mode is entered automatically when an external debugger is connected and the core is halted.

Before entering Debug mode, the circuit will either wait until a started transmission or reception will be finished and Bus idle state is recognized, or immediately interrupt a current transmission or reception. This is depending on bit IDS. Afterwards, the CAN enters Debug mode, indicated by the InitDbg flag, in the CAN Control register. During debug mode, all CAN registers can be accessed. Reading reserved bits will return '0'. Writing to reserved bits will have no effect. Also, the message RAM will be memory mapped. This allows the external debug unit to read the message RAM. For the memory organization (see Section 27.15.3).

**NOTE:** During debug mode, the Message RAM cannot be accessed via the IFx register sets.

**NOTE:** Writing to control registers in Debug mode may influence the CAN state machine and further message handling.

For debug support, the auto clear functionality of the following CAN registers is disabled:

- Error and Status register (clear of status flags by read)
- IF1/IF2 Command registers (clear of DMAActive flag by r/w)

## 27.10 Module Initialization

After hardware reset, the Init bit in the CAN Control register is set and all CAN protocol functions are disabled. The configuration of the bit timing and of the message objects should be completed before the CAN protocol functions are enabled.

For the configuration of the message objects, see Section 27.11.

For the configuration of the Bit Timing, see Section 27.13.2.

The bits MsgVal, NewDat, IntPnd, and TxRqst of the message objects are reset to '0' by a hardware reset. The configuration of a message object is done by programming Mask, Arbitration, Control and Data bits of one of the IF1/IF2 Interface register sets to the desired values. By writing the message object number to bits [7:0] of the corresponding IF1/IF2 Command register, the IF1/IF2 Interface Register content is loaded into the addressed message object in the Message RAM.

The configuration of the bit timing requires that the CCE bit in the CAN Control register is set additionally to Init. This is not required for the configuration of the message objects.

When the Init bit in the CAN Control register is cleared, the CAN Protocol Controller state machine of the CAN Core and the message handler State Machine start to control the CAN's internal data flow. Received messages which pass the acceptance filtering are stored into the Message RAM; messages with pending transmission request are loaded into the CAN Core's Shift register and are transmitted via the CAN bus.

The CPU may enable the interrupt lines (setting IE0 and IE1 to '1') at the same time when it clears Init and CCE. The status interrupts EIE and SIE may be enabled simultaneously.

The CAN communication may be controlled interrupt-driven or in polling mode. The Interrupt Register points to those message objects with IntPnd = '1'. It is updated even if the interrupt lines to the CPU are disabled (IE0 / IE1 are zero).

The CPU may poll all MessageObject's NewDat and TxRqst bits in parallel from the NewData registers and the Transmission Request registers. Polling can be made easier if all Transmit Objects are grouped at the low numbers, all Receive Objects are grouped at the high numbers.

## 27.11 Configuration of Message Objects

The entire Message RAM should to be configured before the end of the initialization; however, it is also possible to change the configuration of message objects during CAN communication.

### *27.11.1 Configuration of a Transmit Object for Data Frames*

Figure 27-6 shows how a transmit object can be initialized.

**Figure 27-6. Initialization of a Transmit Object**

| MsgVal | Arb | Data | Mask | EoB | Dir | NewDat | MsgLst | RxIE | TxIE | IntPnd | RmtEn | TxRqst |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | appl. | appl. | appl. | 1 | 1 | 0 | 0 | 0 | appl. | 0 | appl. | 0 |

- The arbitration bits (ID[28:0] and Xtd bit) are given by the application. They define the identifier and type of the outgoing message. If an 11-bit Identifier (standard frame) is used (Xtd = '0'), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored.
- The data registers (DLC[3:0] and Data0-7) are given by the application, TxRqst and RmtEn should not be set before the data is valid.
- If the TxIE bit is set, the IntPnd bit will be set after a successful transmission of the message object.
- If the RmtEn bit is set, a matching received remote frame will cause the TxRqst bit to be set; the remote frame will autonomously be answered by a data frame.
- The Mask bits (Msk[28:0], UMask, MXtd, and MDir bits) may be used (UMask='1') to allow groups of remote frames with similar identifiers to set the TxRqst bit. The Dir bit should not be masked. For details see Section 27.12.8. Identifier masking must be disabled (UMask = '0') if no remote frames are allowed to set the TxRqst bit (RmtEn = '0').

### 27.11.2 Configuration of a Transmit Object for Remote Frames

It is not necessary to configure transmit objects for the transmission of remote frames. Setting TxRqst for a receive object will cause the transmission of a remote frame with the same identifier as the data frame for which this receive object is configured.

### 27.11.3 Configuration of a Single Receive Object for Data Frames

Figure 27-7 shows how a receive object for data frames can be initialized.

**Figure 27-7. Initialization of a Single Receive Object for Data Frames**

| MsgVal | Arb | Data | Mask | EoB | Dir | NewDat | MsgLst | RxIE | TxIE | IntPnd | RmtEn | TxRqst |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | appl. | appl. | appl. | 1 | 0 | 0 | 0 | appl. | 0 | 0 | 0 | 0 |

- The arbitration bits (ID[28:0] and Xtd bit) are given by the application. They define the identifier and type of accepted received messages. If an 11-bit Identifier (Standard Frame) is used (Xtd = '0'), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored. When a data frame with an 11-bit Identifier is received, ID[17:0] will be set to '0'.
- The data length code (DLC[3:0]) is given by the application. When the message handler stores a data frame in the message object, it will store the received data length code and eight data bytes. If the data length code is less than 8, the remaining bytes of the message object may be overwritten by non specified values.
- The mask bits (Msk[28:0], UMask, MXtd, and MDir bits) may be used (UMask = '1') to allow groups of data frames with similar identifiers to be accepted. The Dir bit should not be masked in typical applications. If some bits of the Mask bits are set to "don't care", the corresponding bits of the Arbitration Register will be overwritten by the bits of the stored data frame.
- If the RxIE bit is set, the IntPnd bit will be set when a received data frame is accepted and stored in the message object.
- If the TxRqst bit is set, the transmission of a remote frame with the same identifier as actually stored in the Arbitration bits will be triggered. The content of the Arbitration bits may change if the Mask bits are used (UMask = '1') for acceptance filtering.

### 27.11.4 Configuration of a Single Receive Object for Remote Frames

Figure 27-8 shows how a receive object for remote frames can be initialized.

**Figure 27-8. Initialization of a single Receive Object for Remote Frames**

| MsgVal | Arb | Data | Mask | EoB | Dir | NewDat | MsgLst | RxIE | TxIE | IntPnd | RmtEn | TxRqst |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | appl. | appl. | appl. | 1 | 1 | 0 | 0 | appl. | 0 | 0 | 0 | 0 |

- Receive objects for remote frames may be used to monitor remote frames on the CAN bus. The remote frame stored in the receive object will not trigger the transmission of a data frame. Receive objects for remote frames may be expanded to a FIFO buffer, see Section 27.11.5.
- UMask must be set to '1'. The Mask bits (Msk[28:0], UMask, MXtd, and MDir bits) may be set to "must-match" or to "don't care", to allow groups of remote frames with similar identifiers to be accepted. The Dir bit should not be masked in typical applications. For details see Section 27.12.8.
- The arbitration bits (ID[28:0] and Xtd bit) may be given by the application. They define the identifier and type of accepted received remote frames. If some bits of the Mask bits are set to "don't care", the corresponding bits of the arbitration bits will be overwritten by the bits of the stored remote frame. If an 11-bit Identifier (standard frame) is used (Xtd = '0'), it is programmed to ID[28:18]. In this case, ID[17:0] can be ignored. When a remote frame with an 11-bit Identifier is received, ID[17:0] will be set to '0'.
- The data length code (DLC[3:0]) may be given by the application. When the message handler stores a remote frame in the message object, it will store the received data length code. The data bytes of the message object will remain unchanged.
- If the RxIE bit is set, the IntPnd bit will be set when a received remote frame is accepted and stored in the message object.

### 27.11.5 Configuration of a FIFO Buffer

With the exception of the EoB bit, the configuration of receive objects belonging to a FIFO buffer is the same as the configuration of a single receive object.

To concatenate multiple message objects to a FIFO buffer, the identifiers and masks (if used) of these message objects have to be programmed to matching values. Due to the implicit priority of the message objects, the message object with the lowest number will be the first message object of the FIFO buffer. The EoB bit of all message objects of a FIFO buffer except the last one have to be programmed to zero. The EoB bits of the last message object of a FIFO buffer is set to one, configuring it as the end of the block.

## 27.12 Message Handling

When initialization is finished, the CAN module synchronizes itself to the traffic on the CAN bus. It does acceptance filtering on received messages and stores those frames that are accepted into the designated message objects. The application has to update the data of the messages to be transmitted and to enable and request their transmission. The transmission is requested automatically when a matching remote frame is received.

The application may read messages which are received and accepted. Messages that are not read before the next messages is accepted for the same message object will be overwritten. Messages may be read interrupt-driven or after polling of NewDat.

### 27.12.1 Message Handler Overview

The message handler state machine controls the data transfer between the Rx/Tx Shift Register of the CAN Core and the Message RAM. It performs the following tasks:

- Data transfer from Message RAM to CAN Core (messages to be transmitted).
- Data transfer from CAN Core to the Message RAM (received messages).
- Data transfer from CAN Core to the Acceptance Filtering unit.
- Scanning of Message RAM for a matching message object (acceptance filtering).
- Scanning the same message object after being changed by IF1/IF2 registers when priority is same or higher as message the object found by last scanning.
- Handling of TxRqst flags.
- Handling of interrupt flags.

The message handler registers contains status flags of all message objects grouped into the following topics:

- Transmission request flags

- New data flags
- Interrupt pending flags
- Message valid registers

Instead of collecting above listed status information of each message object via IFx registers separately, these message handler registers provide a fast and easy way to get an overview, for example, about all pending transmission requests.

All message handler registers are read-only.

### 27.12.2 Receive/Transmit Priority

The receive/transmit priority for the message objects is attached to the message number, not to the CAN identifier. Message object 1 has the highest priority, while message object 32 has the lowest priority. If more than one transmission request is pending, they are serviced due to the priority of the corresponding message object, so for example, messages with the highest priority can be placed in the message objects with the lowest numbers.

The acceptance filtering for received data frames or remote frames is also done in ascending order of message objects, so a frame that has been accepted by a message object cannot be accepted by another message object with a higher message number. The last message object may be configured to accept any data frame or remote frame that was not accepted by any other message object, for nodes that need to log the complete message traffic on the CAN bus.

### 27.12.3 Transmission of Messages in Event Driven CAN Communication

If the shift register of the CAN Core is ready for loading and if there is no data transfer between the IFx registers and Message RAM, the MsgVal bits in the Message Valid register and the TxRqst bits in the transmission request register are evaluated. The valid message object with the highest priority pending transmission request is loaded into the shift register by the message handler and the transmission is started. The message object's NewDat bit is reset.

After a successful transmission and if no new data was written to the message object (NewDat = '0') since the start of the transmission, the TxRqst bit will be reset. If TxIE is set, IntPnd will be set after a successful transmission. If the CAN has lost the arbitration or if an error occurred during the transmission, the message will be retransmitted as soon as the CAN bus is free again. If meanwhile the transmission of a message with higher priority has been requested, the messages will be transmitted in the order of their priority.

If automatic retransmission mode is disabled by setting the DAR bit in the CAN Control register, the behavior of bits TxRqst and NewDat in the Message Control register of the Interface register set is as follows:

- When a transmission starts, the TxRqst bit of the respective Interface register set is reset, while bit NewDat remains set.
- When the transmission has been successfully completed, the NewDat bit is reset.

When a transmission failed (lost arbitration or error) bit NewDat remains set. To restart the transmission, the application has to set TxRqst again.

Received remote frames do not require a receive object. They will automatically trigger the transmission of a data frame, if in the matching Transmit Object the RmtEn bit is set.

### 27.12.4 Updating a Transmit Object

The CPU may update the data bytes of a transmit object any time via the IF1/IF2 interface registers, neither MsgVal nor TxRqst have to be reset before the update.

Even if only a part of the data bytes are to be updated, all four bytes in the corresponding IF1/IF2 Data A register or IF1/IF2 Data B register have to be valid before the content of that register is transferred to the message object. Either the CPU has to write all four bytes into the IF1/IF2 Data register or the message object is transferred to the IF1/IF2 Data Register before the CPU writes the new data bytes.

When only the data bytes are updated, first 0x87 can be written to bits [23:16] of the Command register and then the number of the message object is written to bits [7:0] of the Command register, concurrently updating the data bytes and setting TxRqst with NewDat.

To prevent the reset of TxRqst at the end of a transmission that may already be in progress while the data is updated, NewDat has to be set together with TxRqst in event driven CAN communication. For details see Section 27.12.3.

When NewDat is set together with TxRqst, NewDat will be reset as soon as the new transmission has started.

### 27.12.5 Changing a Transmit Object

If the number of implemented message objects is not sufficient to be used as permanent message objects only, the transmit objects may be managed dynamically. The CPU can write the whole message (Arbitration, Control, and Data) into the Interface register. The bits [23:16] of the Command register can be set to 0xB7 for the transfer of the whole message object content into the message object. Neither MsgVal nor TxRqst have to be reset before this operation.

If a previously requested transmission of this message object is not completed but already in progress, it will be continued; however it will not be repeated if it is disturbed.

To only update the data bytes of a message to be transmitted, bits [23:16] of the Command register should be set to 0x87.

**NOTE:** After the update of the transmit object, the interface register set will contain a copy of the actual contents of the object, including the part that had not been updated.

### 27.12.6 Acceptance Filtering of Received Messages

When the arbitration and control bits (Identifier + IDE + RTR + DLC) of an incoming message is completely shifted into the shift register of the CAN Core, the message handler starts to scan of the message RAM for a matching valid message object:

- The acceptance filtering unit is loaded with the arbitration bits from the CAN Core shift register.
- Then the arbitration and mask bits (including MsgVal, UMask, NewDat, and EoB) of Message Object 1 are loaded into the Acceptance Filtering unit and are compared with the arbitration bits from the shift register. This is repeated for all following message objects until a matching message object is found, or until the end of the Message RAM is reached.
- If a match occurs, the scanning is stopped and the message handler proceeds depending on the type of the frame (data frame or remote frame) received.

### 27.12.7 Reception of Data Frames

The message handler stores the message from the CAN Core shift register into the respective message object in the Message RAM. Not only the data bytes, but all arbitration bits and the data length code are stored into the corresponding message object. This ensures that the data bytes stay associated to the identifier even if arbitration mask registers are used.

The NewDat bit is set to indicate that new data (not yet seen by the CPU) has been received. The CPU should reset the NewDat bit when it reads the message object. If at the time of the reception the NewDat bit was already set, MsgLst is set to indicate that the previous data (not seen by the CPU) is lost. If the RxIE bit is set, the IntPnd bit is set, causing the Interrupt Register to point to this message object.

The TxRqst bit of this message object is reset to prevent the transmission of a remote frame, while the requested data frame has just been received.

### 27.12.8 Reception of Remote Frames

When a remote frame is received, three different configurations of the matching message object have to be considered:

1. Dir = '1' (direction = transmit), RmtEn = '1', UMask = '1' or '0'

The TxRqst bit of this message object is set at the reception of a matching remote frame. The rest of the message object remains unchanged.

2. Dir = '1' (direction = transmit), RmtEn = '0', UMask = '0'

   The remote frame is ignored, this message object remains unchanged.

3. Dir = '1' (direction = transmit), RmtEn = '0', UMask = '1'

   The remote frame is treated similar to a received data frame. At the reception of a matching remote frame, the TxRqst bit of this message object is reset. The arbitration and control bits (Identifier + IDE + RTR + DLC) from the shift register are stored in the message object in the Message RAM and the NewDat bit of this message object is set. The data bytes of the message object remain unchanged

### 27.12.9 Reading Received Messages

The CPU may read a received message any time via the IFx interface registers, the data consistency is guaranteed by the message handler state machine.

Typically the CPU will write 0x7F to bits [23:16] and then the number of the message object to bits [7:0] of the Command Register. That combination will transfer the whole received message from the Message RAM into the Interface Register set. Additionally, the bits NewDat and IntPnd are cleared in the Message RAM (not in the Interface Register set). The values of these bits in the Message Control Register always reflect the status before resetting the bits.

If the message object uses masks for acceptance filtering, the arbitration bits show which of the different matching messages has been received.

The actual value of NewDat shows whether a new message has been received since last time when this message object was read. The actual value of MsgLst shows whether more than one message have been received since the last time when this message object was read. MsgLst will not be automatically reset.

### 27.12.10 Requesting New Data for a Receive Object

By means of a remote frame, the CPU may request another CAN node to provide new data for a receive object. Setting the TxRqst bit of a receive object will cause the transmission of a remote frame with the receive object's identifier. This remote frame triggers the other CAN node to start the transmission of the matching data frame. If the matching data frame is received before the remote frame could be transmitted, the TxRqst bit is automatically reset.

Setting the TxRqst bit without changing the contents of a message object requires the value 0x84 in bits [23:16] of the Command Register.

### 27.12.11 Storing Received Messages in FIFO Buffers

Several message objects may be grouped to form one or more FIFO Buffers. Each FIFO Buffer configured to store received messages with a particular (group of) Identifier(s). Arbitration and Mask registers of the FIFO Buffer's message objects are identical. The EoB (End of Buffer) bits of all but the last of the FIFO Buffer's message objects are '0', in the last one the EoB bit is '1'.

Received messages with identifiers matching to a FIFO Buffer are stored into a message object of this FIFO Buffer, starting with the message object with the lowest message number.

When a message is stored into a message object of a FIFO Buffer the NewDat bit of this message object is set. By setting NewDat while EoB is '0' the message object is locked for further write accesses by the message handler until the CPU has cleared the NewDat bit.

Messages are stored into a FIFO Buffer until the last message object of this FIFO Buffer is reached. If none of the preceding message objects is released by writing NewDat to '0', all further messages for this FIFO Buffer will be written into the last message object of the FIFO Buffer (EoB = '1') and therefore overwrite previous messages in this message object.

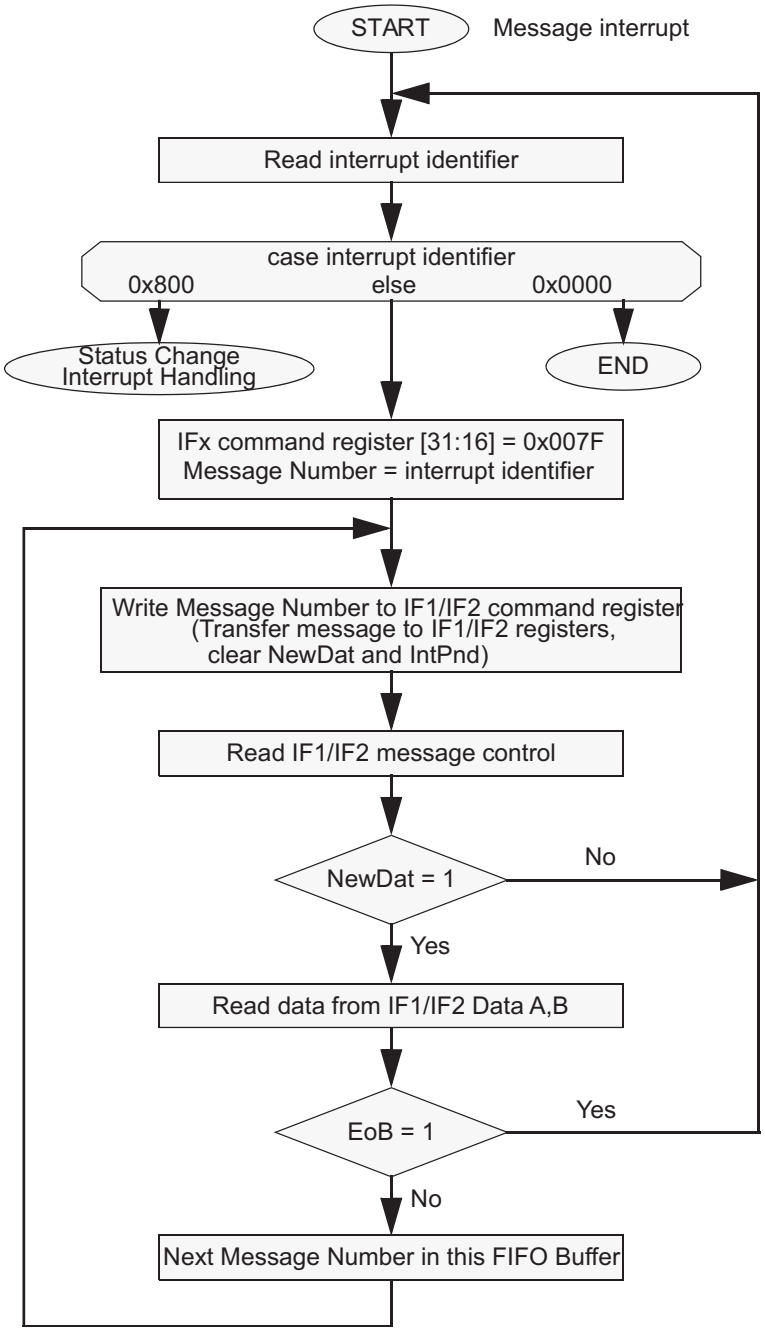
### 27.12.12 Reading from a FIFO Buffer

Several messages may be accumulated in a set of message objects which are concatenated to form a FIFO Buffer before the application program is required (in order to avoid the loss of data) to empty the buffer. A FIFO Buffer of length N will store N-1 plus the last received message since last time it was cleared. A FIFO Buffer is cleared by reading and resetting the NewDat bits of all its message objects, starting at the FIFO Object with the lowest message number. This should be done in a subroutine following the example shown in Figure 27-9.

**NOTE:** All message objects of a FIFO buffer needs to be read and cleared before the next batch of messages can be stored. Otherwise true FIFO functionality cannot be guaranteed, since the message objects of a partly read buffer will be re-filled according to the normal (descending) priority.

Reading from a FIFO Buffer message object and resetting its NewDat bit is handled the same way as reading from a single message object.

**Figure 27-9. CPU Handling of a FIFO Buffer (Interrupt Driven)**




## 27.13 CAN Bit Timing

The CAN supports bit rates between less than 1 kBit/s and 1000 kBit/s.

Each member of the CAN network has its own clock generator, typically derived from a crystal oscillator. The Bit timing parameters can be configured individually for each CAN node, creating a common Bit rate even though the CAN nodes' oscillator periods ($F_{osc}$) may be different.

The frequencies of these oscillators are not absolutely stable. Small variations are caused by changes in temperature or voltage and by deteriorating components. As long as the variations remain inside a specific oscillator tolerance range (df), the CAN nodes are able to compensate for the different bit rates by resynchronizing to the bit stream.

In many cases, the CAN bit synchronization will amend a faulty configuration of the CAN bit timing to such a degree that only occasionally an error frame is generated. In the case of arbitration however, when two or more CAN nodes simultaneously try to transmit a frame, a misplaced sample point may cause one of the transmitters to become error passive.

The analysis of such sporadic errors requires a detailed knowledge of the CAN bit synchronization inside a CAN node and of the CAN nodes' interaction on the CAN bus.

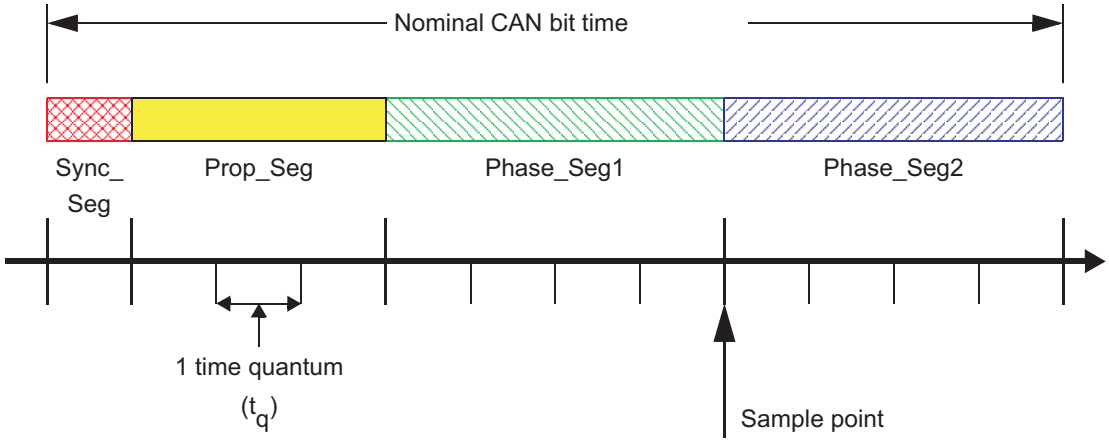
Even if minor errors in the configuration of the CAN bit timing do not result in immediate failure, the performance of a CAN network can be reduced significantly.

### 27.13.1 Bit Time and Bit Rate

According to the CAN specification, the Bit time is divided into four segments (see Figure 27-10):

- Synchronization Segment (Sync_Seg)
- Propagation Time Segment (Prop_Seg)
- Phase Buffer Segment 1 (Phase_Seg1)
- Phase Buffer Segment 2 (Phase_Seg2)

**Figure 27-10. Bit Timing**




Each segment consists of a specific number of time quanta. The length of one time quantum ($t_q$), which is the basic time unit of the bit time, is given by the CAN_CLK and the Baud Rate Prescalers (BRPE and BRP). With these two Baud Rate Prescalers combined, divider values from 1 to 1024 can be programmed:

$$t_q = \frac{\text{Baud Rate Prescaler}}{\text{CAN\_CLK}}$$

Apart from the fixed length of the synchronization segment, these numbers are programmable. Table 27-4 describes the minimum programmable ranges required by the CAN protocol.

A given bit rate may be met by different bit time configurations.

**Table 27-4. Programmable Ranges Required by CAN Protocol**

| Parameter | Range | Remark |
|---|---|---|
| Sync_Seg | 1 $t_q$ (fixed) | Synchronization of bus input to CAN_CLK |
| Prop_Seg | [1 … 8] $t_q$ | Compensates for the physical delay times |
| Phase_Seg1 | [1 … 8] $t_q$ | May be lengthened temporarily by synchronization |
| Phase_Seg2 | [1 … 8] $t_q$ | May be shortened temporarily by synchronization |
| Synchronization Jump Width (SJW) | [1 … 4] $t_q$ | May not be longer than either phase buffer segment |

**NOTE:** For proper functionality of the CAN network, the physical delay times and the oscillator's tolerance range have to be considered.

#### 27.13.1.1 Synchronization Segment

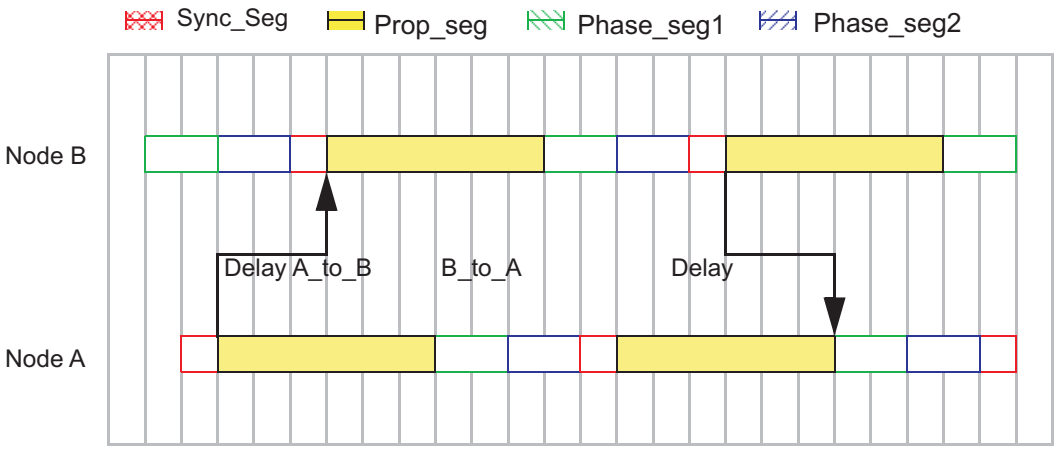
The Synchronization Segment (Sync_Seg) is the part of the bit time where edges of the CAN bus level are expected to occur. If an edge occurs outside of Sync_Seg, its distance to the Sync_Seg is called the phase error of this edge.

#### 27.13.1.2 Propagation Time Segment

This part of the bit time is used to compensate physical delay times within the CAN network. These delay times consist of the signal propagation time on the bus and the internal delay time of the CAN nodes.

Any CAN node synchronized to the bit stream on the CAN bus can be out of phase with the transmitter of the bit stream, caused by the signal propagation time between the two nodes. The CAN protocol's nondestructive bitwise arbitration and the dominant acknowledge bit provided by receivers of CAN messages require that a CAN node transmitting a bit stream must also be able to receive dominant bits transmitted by other CAN nodes that are synchronized to that bit stream. The example in Figure 27-11 shows the phase shift and propagation times between two CAN nodes.

**Figure 27-11. The Propagation Time Segment**




Delay A_to_B >= node output delay(A) + bus line delay(AÆB) + node input delay(B)

Prop_Seg >= Delay A_to_B + Delay B_to_A

Prop_Seg >= 2 • [max(node output delay+ bus line delay + node input delay)]

In this example, both nodes A and B are transmitters performing an arbitration for the CAN bus. The node A has sent its Start of Frame bit less than one bit time earlier than node B, therefore node B has synchronized itself to the received edge from recessive to dominant. Since node B has received this edge delay(A_to_B) after it has been transmitted, node B's bit timing segments are shifted with regard to node A. Node B sends an identifier with higher priority and so it will win the arbitration at a specific identifier bit when it transmits a dominant bit while node A transmits a recessive bit. The dominant bit transmitted by node B will arrive at node A after the delay(B_to_A).

Due to oscillator tolerances, the actual position of node A's Sample Point can be anywhere inside the nominal range of node A's Phase Buffer Segments, so the bit transmitted by node B must arrive at node A before the start of Phase_Seg1. This condition defines the length of Prop_Seg.

If the edge from recessive to dominant transmitted by node B would arrive at node A after the start of Phase_Seg1, it could happen that node A samples a recessive bit instead of a dominant bit, resulting in a bit error and the destruction of the current frame by an error flag.

This error only occurs when two nodes arbitrate for the CAN bus which have oscillators of opposite ends of the tolerance range and are separated by a long bus line; this is an example of a minor error in the Bit timing configuration (Prop_Seg too short) that causes sporadic bus errors.

Some CAN implementations provide an optional 3 Sample Mode. The CAN does not. In this mode, the CAN bus input signal passes a digital low-pass filter, using three samples and a majority logic to determine the valid bit value. This results in an additional input delay of 1 $t_q$, requiring a longer Prop_Seg.

#### 27.13.1.3 Phase Buffer Segments and Synchronization

The phase buffer segments (Phase_Seg1 and Phase_Seg2) and the synchronization jump width (SJW) are used to compensate for the oscillator tolerance.

The phase buffer segments surround the sample point. The phase buffer segments may be lengthened or shortened by synchronization.

The synchronization jump width (SJW) defines how far the resynchronizing mechanism may move the sample point inside the limits defined by the phase buffer segments to compensate for edge phase errors.

Synchronizations occur on edges from recessive to dominant. Their purpose is to control the distance between edges and sample points.

Edges are detected by sampling the actual bus level in each time quantum and comparing it with the bus level at the previous sample point. A synchronization may be done only if a recessive bit was sampled at the previous sample point and if the actual time quantum's bus level is dominant.

An edge is synchronous if it occurs inside of Sync_Seg, otherwise its distance to the Sync_Seg is the edge phase error, measured in time quanta. If the edge occurs before Sync_Seg, the phase error is negative, else it is positive.

Two types of synchronization exist: hard synchronization and resynchronizing. A hard synchronization is done once at the start of a frame; inside a frame only resynchronization is possible.

- Hard Synchronization

  After a hard synchronization, the bit time is restarted with the end of Sync_Seg, regardless of the edge phase error. Thus hard synchronization forces the edge which has caused the hard synchronization to lie within the synchronization segment of the restarted bit time.

- Bit Resynchronizations

  Resynchronization leads to a shortening or lengthening of the bit time such that the position of the sample point is shifted with regard to the edge.

  When the phase error of the edge which causes resynchronization is positive, Phase_Seg1 is lengthened. If the magnitude of the phase error is less than SJW, Phase_Seg1 is lengthened by the magnitude of the phase error, else it is lengthened by SJW.

  When the phase error of the edge which causes resynchronization is negative, Phase_Seg2 is shortened. If the magnitude of the phase error is less than SJW, Phase_Seg2 is shortened by the magnitude of the phase error, else it is shortened by SJW.

If the magnitude of the phase error of the edge is less than or equal to the programmed value of SJW, the results of hard synchronization and resynchronization are the same. If the magnitude of the phase error is larger than SJW, the resynchronization cannot compensate the phase error completely, and an error of (phase error - SJW) remains.

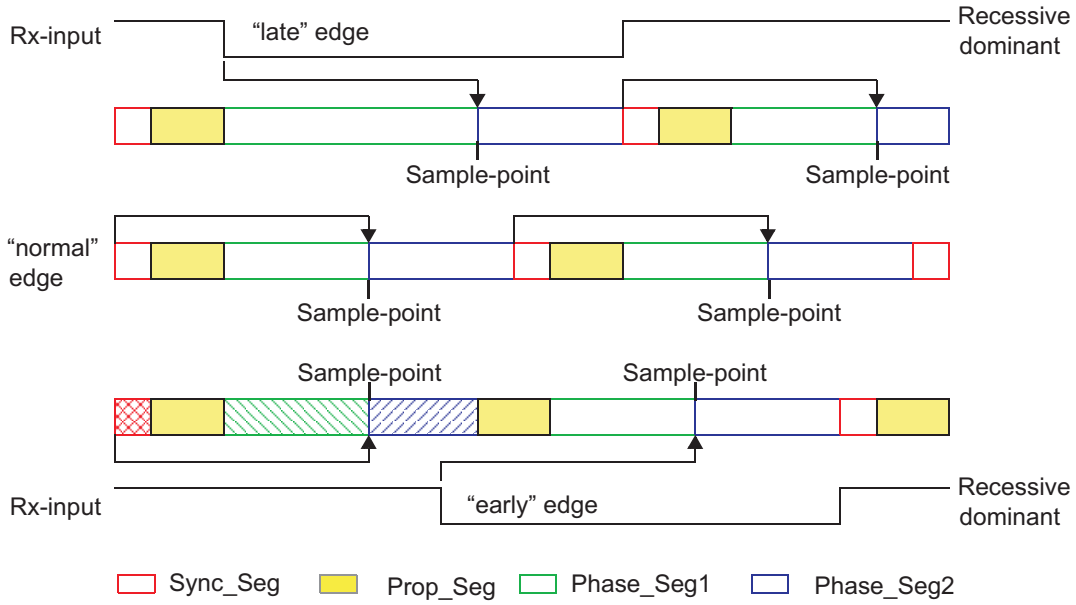
Only one synchronization may be done between two sample points. The synchronizations maintain a minimum distance between edges and sample points, giving the bus level time to stabilize and filtering out spikes that are shorter than (Prop_Seg + Phase_Seg1).

Apart from noise spikes, most synchronizations are caused by arbitration. All nodes synchronize "hard" on the edge transmitted by the "leading" transceiver that started transmitting first, but due to propagation delay times, they cannot become ideally synchronized. The "leading" transmitter does not necessarily win the arbitration, therefore the receivers have to synchronize themselves to different transmitters that subsequently "take the lead" and that are differently synchronized to the previously "leading" transmitter. The same happens at the acknowledge field, where the transmitter and some of the receivers will have to synchronize to that receiver that "takes the lead" in the transmission of the dominant acknowledge bit.

Synchronizations after the end of the arbitration will be caused by oscillator tolerance, when the differences in the oscillator's clock periods of transmitter and receivers sum up during the time between synchronizations (at most ten bits). These summarized differences may not be longer than the SJW, limiting the oscillator's tolerance range.

The examples in Figure 27-12 show how the phase buffer segments are used to compensate for phase errors. There are three drawings of each two consecutive bit timings. The upper drawing shows the synchronization on a "late" edge, the lower drawing shows the synchronization on an "early" edge, and the middle drawing is the reference without synchronization.

**Figure 27-12. Synchronization on Late and Early Edges**




In the first example, an edge from recessive to dominant occurs at the end of Prop_Seg. The edge is "late" since it occurs after the Sync_Seg. Reacting to the "late" edge, Phase_Seg1 is lengthened so that the distance from the edge to the sample point is the same as it would have been from the Sync_Seg to the sample point if no edge had occurred. The phase error of this "late" edge is less than SJW, so it is fully compensated and the edge from dominant to recessive at the end of the bit, which is one nominal bit time long, occurs in the Sync_Seg.

In the second example, an edge from recessive to dominant occurs during Phase_Seg2. The edge is "early" since it occurs before a Sync_Seg. Reacting to the "early" edge, Phase_Seg2 is shortened and Sync_Seg is omitted, so that the distance from the edge to the sample point is the same as it would have been from a Sync_Seg to the sample point if no edge had occurred. As in the previous example, the magnitude of this "early" edge's phase error is less than SJW, so it is fully compensated.

The phase buffer segments are lengthened or shortened temporarily only; at the next bit time, the segments return to their nominal programmed values.

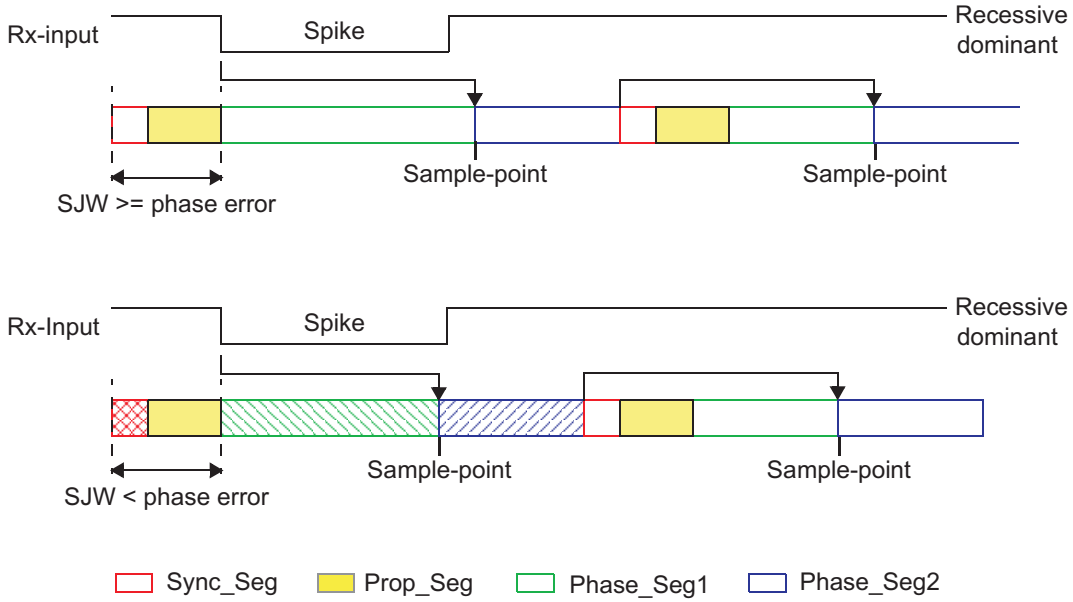
In these examples, the bit timing is seen from the point of view of the CAN implementation's state machine, where the bit time starts and ends at the sample points. The state machine omits Sync_Seg when synchronizing on an "*early*" edge because it cannot subsequently redefine that time quantum of Phase_Seg2 where the edge occurs to be the Sync_Seg.

The examples in Figure 27-13 show how short dominant noise spikes are filtered by synchronizations. In both examples, the spike starts at the end of Prop_Seg and has the length of (Prop_Seg + Phase_Seg1).

In the first example, the synchronization jump width is greater than or equal to the phase error of the spike's edge from recessive to dominant. Therefore the sample point is shifted after the end of the spike; a recessive bus level is sampled.

In the second example, SJW is shorter than the phase error, so the sample point cannot be shifted far enough; the dominant spike is sampled as actual bus level.

**Figure 27-13. Filtering of Short Dominant Spikes**




### 27.13.1.4 Oscillator Tolerance Range

With the introduction of CAN protocol version 1.2, the option to synchronize on edges from dominant to recessive became obsolete. Only edges from recessive to dominant are considered for synchronization. The protocol update to version 2.0 (A and B) had no influence on the oscillator tolerance.

The tolerance range df for an oscillator's frequency $f_{osc}$ around the nominal frequency $f_{nom}$ with

$(1 - df) \bullet f_{nom} \leq f_{osc} \leq (1 + df) \bullet f_{nom}$

depends on the proportions of Phase_Seg1, Phase_Seg2, SJW, and the bit time. The maximum tolerance df is the defined by two conditions (both shall be met):

(11)

$$df \leq \frac{\min(Tseg1, Tseg2)}{2((13 \times bit\ time) - Tseg2)}$$

$$df \leq \frac{SJW}{20 \times bit\_time}$$

(12)

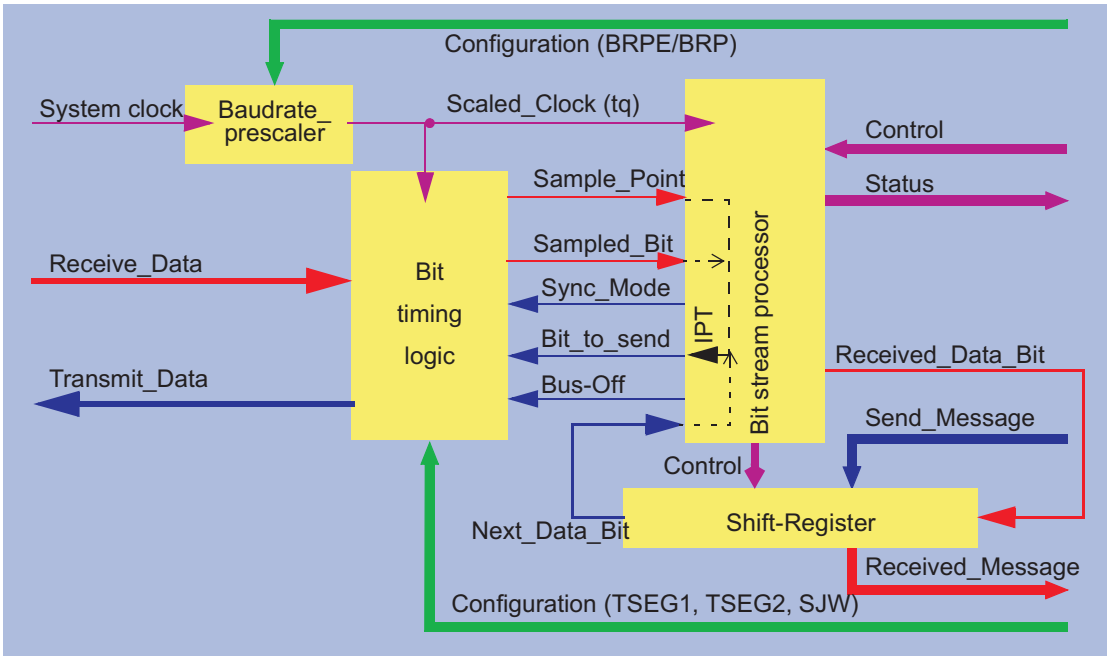
It has to be considered that SJW may not be larger than the smaller of the phase buffer segments and that the propagation time segment limits that part of the bit time that may be used for the phase buffer segments.

The combination Prop_Seg = 1 and Phase_Seg1 = Phase_Seg2 = SJW = 4 allows the largest possible oscillator tolerance of 1.58%. This combination with a Propagation Time Segment of only 10% of the bit time is not suitable for short bit times; it can be used for bit rates of up to 125 kBit/s (bit time = 8 µs) with a bus length of 40 m.

## *27.13.2 Configuration of the CAN Bit Timing*

In the CAN, the bit timing configuration is programmed in two register bytes, additionally a third byte for a baud rate prescaler extension of 4 bits (BRPE) is provided. The sum of Prop_Seg and Phase_Seg1 (as TSEG1) is combined with Phase_Seg2 (as TSEG2) in one byte, SJW and BRP (plus BRPE in third byte) are combined in the other byte (see Figure 27-14).

**Figure 27-14. Structure of the CAN Core's CAN Protocol Controller**




In this bit timing register, the components TSEG1, TSEG2, SJW and BRP have to be programmed to a numerical value that is one less than its functional value; so instead of values in the range of [1…n], values in the range of [0…n-1] are programmed. That way, for example, SJW (functional range of [1…4]) is represented by only two bits.

Therefore the length of the Bit time is (programmed values) [TSEG1 + TSEG2 + 3] $t_q$ or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] $t_q$.

The data in the Bit Timing Register is the configuration input of the CAN protocol controller. The baud rate prescaler (configured by BRPE/BRP) defines the length of the time quantum (the basic time unit of the bit time); the bit timing logic (configured by TSEG1, TSEG2, and SJW) defines the number of time quanta in the bit time.

The processing of the bit time, the calculation of the position of the Sample Point, and occasional synchronizations are controlled by the Bit timing state machine, which is evaluated once each time quantum. The rest of the CAN protocol controller, the Bit Stream Processor (BSP) state machine, is evaluated once each bit time, at the Sample Point.

The Shift register serializes the messages to be sent and parallelizes received messages. Its loading and shifting is controlled by the BSP.

The BSP translates messages into frames and vice versa. It generates and discards the enclosing fixed format bits, inserts and extracts stuff bits, calculates and checks the CRC code, performs the error management, and decides which type of synchronization is to be used. It is evaluated at the sample point and processes the sampled bus input bit. The time after the sample point that is needed to calculate the next bit to be sent (for example, data bit, CRC bit, stuff bit, error flag, or idle) is called the Information Processing Time (IPT), which is 0 $t_q$ for the CAN.

Generally, the IPT is CAN controller specific, but may not be longer than 2 $t_q$. The IPT length is the lower limit of the programmed length of Phase_Seg2. In case of a synchronization, Phase_Seg2 may be shortened to a value less than IPT, which does not affect bus timing.

### 27.13.2.1 Calculation of the Bit Timing Parameters

Usually, the calculation of the bit timing configuration starts with a desired bit rate or bit time. The resulting Bit time (1 / Bit rate) must be an integer multiple of the CAN clock period.

**NOTE:** 8 MHz is the minimum CAN clock frequency required to operate the CAN at a bit rate of 1 MBit/s.

The bit time may consist of 8 to 25 time quanta. The length of the time quantum $t_q$ is defined by the Baud Rate Prescaler with $t_q$ = (Baud Rate Prescaler) / CAN_CLK. Several combinations may lead to the desired bit time, allowing iterations of the following steps.

The first part of the bit time to be defined is the Prop_Seg. Its length depends on the delay times measured in the system. A maximum bus length as well as a maximum node delay has to be defined for expandible 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 may not be shorter than any CAN controller's Information Processing Time in the network, which is device dependent and can be in the range of [0…2] $t_q$.

The length of the synchronization jump width is set to its maximum value, which is the minimum of 4 and Phase_Seg1.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formulas given in Section 27.13.1.4.

If more than one configurations are possible to reach a certain Bit rate, it is recommended to choose the configuration which allows the highest oscillator tolerance range.

CAN nodes with different clocks require different configurations to come to the same bit rate. The calculation of the propagation time in the CAN network, based on the nodes with the longest delay times, is done once for the whole network.

The CAN system's oscillator tolerance range is limited by the node with the lowest tolerance range.

The calculation may show that bus length or bit rate have to be decreased or that the oscillator frequencies' stability has to be increased in order to find a protocol compliant configuration of the CAN bit timing.

The resulting configuration is written into the Bit Timing register:

(Phase_Seg2-1)&(Phase_Seg1+Prop_Seg-1)&

(SynchronizationJumpWidth-1)&(Prescaler-1)

### 27.13.2.2 Example for Bit Timing at High Baudrate

In this example, the frequency of CAN_CLK is 10 MHz, BRP is 0, the bit rate is 1 MBit/s.

| | | | | |
|---|---|---|---|---|
| $t_q$ | 100 | ns | = | $t_{CAN\_CLK}$ |
| delay of bus driver | 90 | ns | = | |
| delay of receiver circuit | 40 | ns | = | |
| delay of bus line (40m) | 220 | ns | = | |
| $t_{Prop}$ | 700 | ns | = | 2*delays = 7 • $t_q$ |
| $t_{SJW}$ | 100 | ns | = | 1 • $t_q$ |
| $t_{TSeg1}$ | 800 | ns | = | $t_{Prop}$ + $t_{SJW}$ |
| $t_{TSeg2}$ | 100 | ns | = | Information Processing Time + 1 • $t_q$ |
| $t_{Sync\text{-}Seg}$ | 100 | ns | = | 1 • $t_q$ |
| bit time | 1000 | ns | = | $t_{Sync\text{-}Seg}$ + $t_{TSeg1}$ + $t_{TSeg2}$ |

| | | | | |
|---|---|---|---|---|
| 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 = 0x00000700.

#### 27.13.2.3 Example for Bit Timing at Low Baudrate

In this example, the frequency of CAN_CLK is 2 MHz, BRP is 1, the bit rate is 100 KBit/s.

| | | | | |
|---|---|---|---|---|
| $t_q$ | 1 | µs | = | $2 \bullet t_{CAN\_CLK}$ |
| delay of bus driver | 200 | ns | = | |
| delay of receiver circuit | 80 | ns | = | |
| delay of bus line (40m) | 220 | ns | = | |
| $t_{Prop}$ | 1 | µs | = | $1 \bullet t_q$ |
| $t_{SJW}$ | 4 | µs | = | $4 \bullet t_q$ |
| $t_{TSeg1}$ | 5 | µs | = | $t_{Prop} + t_{SJW}$ |
| $t_{TSeg2}$ | 4 | µs | = | Information Processing Time + $4 \bullet t_q$ |
| $t_{Sync\text{-}Seg}$ | 1 | µs | = | $1 \bullet t_q$ |
| bit time | 10 | µs | = | $t_{Sync\text{-}Seg} + t_{TSeg1} + t_{TSeg2}$ |
| 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 = 0x000034C1.

## 27.14 Message Interface Register Sets

The interface register sets control the CPU read and write accesses to the Message RAM. There are two interface register sets for read / write access (IF1 and IF2) and one Interface Register Set for read access only (IF3).

Due to the structure of the Message RAM, it is not possible to change single bits or bytes of a message object. Instead, always a complete message object in the Message RAM is accessed. Therefore the data transfer from the IF1/IF2 registers to the Message RAM requires the message handler to perform a read-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 which are not selected in the Command Register, will be set to the actual contents of the selected message object. After the partial read of a message object, those parts of the Interface Register set which are not selected in the Command Register, will be left unchanged.

By buffering the data to be transferred, the Interface Register sets avoid conflicts between concurrent CPU accesses to the Message RAM and CAN message reception and transmission. A complete message object (see Section 27.15.1) or parts of the message object may be transferred between the Message RAM and the IF1/IF2 Register set in one single transfer. This transfer, performed in parallel on all selected parts of the message object, guarantees the data consistency of the CAN message.

That being said, there is one condition that can cause a write access to the message RAM to be lost. If MsgVal = 1 for the message object which is accessed and CAN communication is ongoing, a transfer from the IFx register to message RAM may be lost. The reason for this is that it might happen that the IFx register write to the message RAM occurs in between a read-modify-write access of the Host Message Handler when it is in the process of receiving a message for the same message object.

To avoid this issue with receive mail boxes, reset MsgVal before changing any of the following: Id28-0, Xtd, Dir, DLC3-0, RxIE, TxIE, RmtEn, EoB, Umask, Msk28-0, MXtd, and MDir.

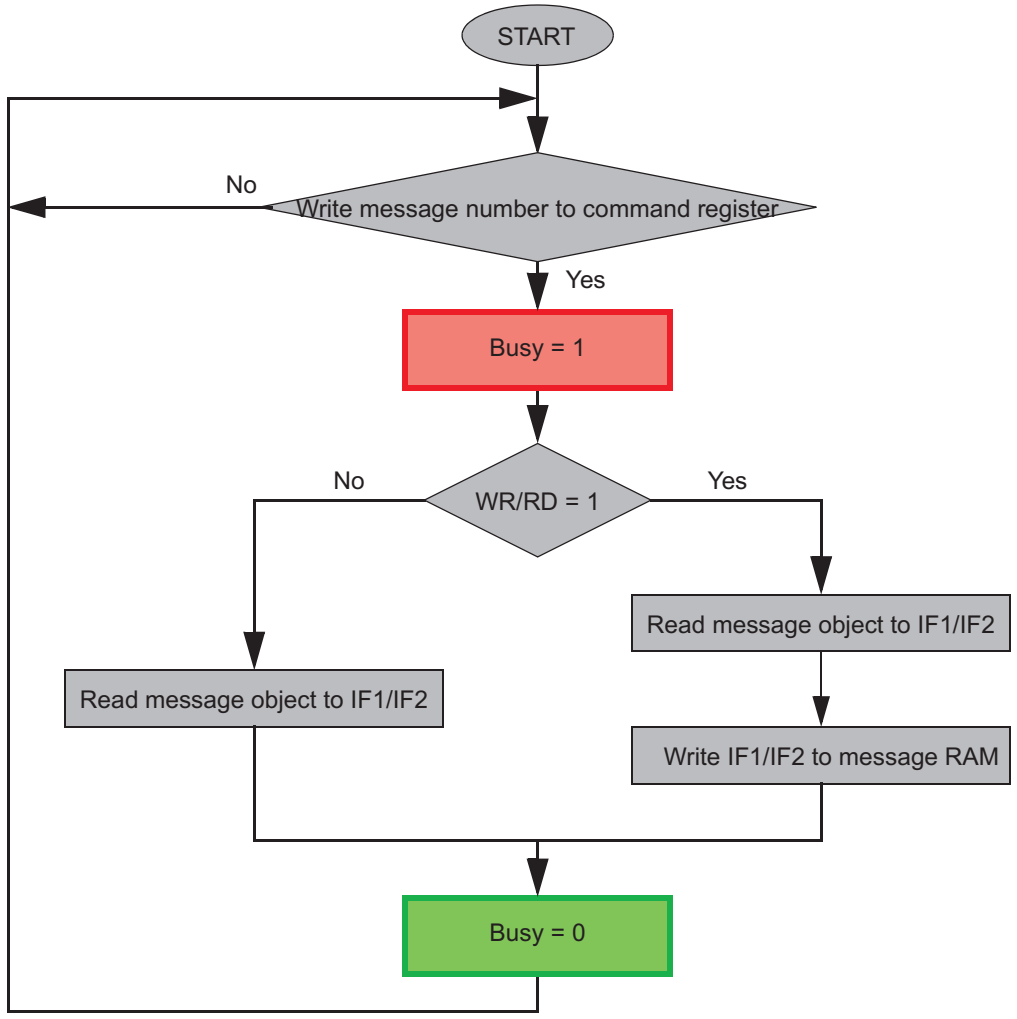
To avoid this issue with transmit mail boxes, reset MsgVal before changing any of the following: Dir, RxIE, TxIE, RmtEn, EoB, Umask, Msk28-0, MXtd, and MDir. Other fields not listed above, like Data, may be changed without fear of losing a write to the message RAM.

### 27.14.1 Message Interface Register Sets 1 and 2

The IF1 and IF2 register sets allow data transfers to and from the message objects. The IFxCMD register for an interface control the direction of the data transfer. If the IFxCMD register is set to write, then the message object fields selected by the IFxCMD register will be overwritten by values taken from the other IFx registers. Is the IFxCMD register is set to read, then the message object fields selected by the IFxCMD register will be copied from the message object to the other IFx registers. The interfaces allow for transfers of a complete message object as well as individual parts. The transfer begins with the desired message object number is written to bits 7:0 of the IFxCMD register.

When the CPU initiates a data transfer between the IF1/IF2 registers and Message RAM, the message handler sets the Busy bit in the respective Command Register to '1'. After the transfer has completed, the Busy bit is set back to '0' (see Figure 27-15 ).

**Figure 27-15. Data Transfer Between IF1 / IF2 Registers and Message RAM**




### *27.14.2 IF3 Register Set*

The IF3 register set can automatically be updated with received message objects without the need to initiate the transfer from Message RAM by CPU. The automatic update functionality can be programmed for each message object (see the IF3 Update Enable register).

All valid message objects in Message RAM which are configured for automatic update, will be checked for active NewDat flags. If such a message object is found, it will be transferred to the IF3 register (if no previous DMA transfers are ongoing), controlled by IF3 Observation register. If more than one NewDat flag is active, the message object with the lowest number has the highest priority for automatic IF3 update.

The NewDat bit in the message object will be reset by a transfer to IF3.

If DCAN internal IF3 update is complete, a DMA will be requested. The DMA request stays active until the first read access to one of the IF3 registers. The DMA functionality has to be enabled by setting bit DE3 in the CAN Control register.

**NOTE:** The IF3 register set can not be used for transferring data into message objects.

## 27.15 Message RAM

The CAN Message RAM contains message objects and parity bits for the message objects. There are 32 message objects in the Message RAM.

During normal operation, accesses to the Message RAM are performed via the Interface Register sets, and the CPU cannot directly access the Message RAM.

The Interface Register sets IF1 and IF2 provide indirect read/write access from the CPU to the Message RAM. The IF1 and IF2 register sets can buffer control and user data to be transferred to and from the message objects.

The third Interface Register set IF3 can be configured to automatically receive control and user data from the Message RAM when a message object has been updated after reception of a CAN message. The CPU does not need to initiate the transfer from Message RAM to IF3 Register set.

The message handler avoids potential conflicts between concurrent accesses to Message RAM and CAN frame reception/transmission.

The message RAM can only be accessed in debug mode. The message RAM base address is 0x1000 above the base address of the CAN peripheral.

### *27.15.1 Structure of Message Objects*

Figure 27-16 shows the structure of a message object.

The grayed fields are those parts of the message object which are represented in dedicated registers. For example, the transmit request flags of all message objects are represented in centralized transmit request registers.

**Figure 27-16. Structure of a Message Object**

| Message Object | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UMask | Msk[28:0] | MXtd | MDir | EoB | unused | NewDat | MsgLst | RxIE | TxIE | IntPnd | RmtEn | TxRqst |
| MsgVal | ID[28:0] | Xtd | Dir | DLC[3:0] | Data 0 | Data 1 | Data 2 | Data 3 | Data 4 | Data 5 | Data 6 | Data 7 |

**Table 27-5. 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 may be kept at level '1' even when the identifier bits ID[28:0], the control bits Xtd, Dir, or the data length code DLC[3:0] are changed. It should be reset if the Messages Object is no longer required. |
| UMask | | 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 one, the message object's mask bits have to be programmed during initialization of the message object before MsgVal is set to one. |
| ID[28:0] | | Message Identifier |
| | ID[28:0] | 29-bit ("extended") identifier bits |
| | ID[28:18] | 11-bit ("standard") identifier bits |
| 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 C28xx devices, where a "1" means the corresponding bit is NOT used for filtering, and "0" means it is used. |

**Table 27-5. Message Object Field Descriptions (continued)**

| Name | Value | Description |
|---|---|---|
| Xtd | | Extended Identifier |
| | 0 | The 11-bit ("standard") identifier will be used for this message object. |
| | 1 | The 29-bit ("extended") identifier will be used for this message object. |
| MXtd | | 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 = one). |
| 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 one. |
| 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 will not be triggered after the successful reception of a frame. |
| | 1 | IntPnd will be triggered after the successful reception of a frame. |
| TxIE | | Transmit Interrupt Enable |
| | 0 | IntPnd will not be triggered after the successful transmission of a frame. |
| | 1 | IntPnd will be triggered after the successful transmission of a frame. |
| 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 will point to this message object if there is no other interrupt source with higher priority. |
| 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 27.12.8 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. |

**Table 27-5. Message Object Field Descriptions (continued)**

| Name | Value | Description |
|---|---|---|
| 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, it will write the DLC to the value given by the received message. |
| Data 0 | | 1st data byte of a CAN data frame |
| Data 1 | | 2nd data byte of a CAN data frame |
| Data 2 | | 3rd data byte of a CAN data frame |
| Data 3 | | 4th data byte of a CAN data frame |
| Data 4 | | 5th data byte of a CAN data frame |
| Data 5 | | 6th data byte of a CAN data frame |
| Data 6 | | 7th data byte of a CAN data frame |
| Data 7 | | 8th data byte of a CAN data frame |
| | | Note: Byte Data 0 is the first data byte shifted into the shift register of the CAN Core during a reception, byte Data 7 is the last. When the message handler stores a data frame, it will write all the eight data bytes into a message object. If the data length code is less than 8, the remaining bytes of the message object may be overwritten by undefined values. |

### 27.15.2 Addressing Message Objects in RAM

The starting location of a particular message object in RAM is:

Message RAM base address + (message object number) * 0x20.

This means that Message Object 1 starts at offset 0x0020; Message Object 2 starts at offset 0x0040, etc.

**NOTE:** '0' is not a valid message object number. At address 0x0000, the last message object (32) (with the lowest priority) is located. Writing to the address of an unimplemented message object may overwrite an implemented message object.

Message Object number 1 has the highest priority.

**Table 27-6. 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 |

(1) See Section 27.15.3.

**Table 27-6. Message RAM Addressing in Debug Mode (continued)**

| Message Object Number | Offset From Base Address | Word Number | Debug Mode [1] |
|---|---|---|---|
| 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 |

### 27.15.3 Message RAM Representation in Debug Mode

In debug mode, the Message RAM will be memory mapped. This allows the external debug unit to access the Message RAM.

**NOTE:** During debug mode, the Message RAM cannot be accessed via the IFx register sets.

**Figure 27-17. Message RAM Representation in Debug Mode**

| 31/15 | 30/14 | 29/13 | 28/12 | 27/11 | 26/10 | 25/9 | 24/8 | 23/7 | 22/6 | 21/5 | 20/4 | 19/3 | 18/2 | 17/1 | 16/0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **MsgAddr + 0x00** | | | | | | | | | | | | | | | |
| Reserved | | | | | | | | | | | | | | | |
| Reserved | | | | | | | | | | | | | | Parity[4:0] | |
| **MsgAddr + 0x04** | | | | | | | | | | | | | | | |
| MXtd | MDir | Rsvd | Msk[28:16] | | | | | | | | | | | | |
| Msk[15:0] | | | | | | | | | | | | | | | |
| **MsgAddr + 0x08** | | | | | | | | | | | | | | | |
| Rsvd | Xtd | Dir | ID[28:16] | | | | | | | | | | | | |
| ID[15:0] | | | | | | | | | | | | | | | |
| **MsgAddr + 0x0C** | | | | | | | | | | | | | | | |
| Reserved | | | | | | | | | | | | | | | |
| Rsvd | 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 | | | | | | | |

## 27.16 Registers

### 27.16.1 Controller Area Network Base Addresses

**Table 27-7. 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 |

### 27.16.1.1 CAN_REGS Registers

Table 27-8 lists the memory-mapped registers for the CAN_REGS. All register offset addresses not listed in Table 27-8 should be considered as reserved locations and the register contents should not be modified.

**Table 27-8. 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 |

Complex bit access types are encoded to fit into small table cells. Table 27-9 shows the codes that are used for access types in this section.

**Table 27-9. CAN_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| **Write Type** | | |
| W | W | Write |
| W=1 | 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. |

#### 27.16.1.1.1 CAN_CTL Register (Offset = 0h) [reset = 1401h]

CAN_CTL is shown in Figure 27-18 and described in Table 27-10.

Return to Summary Table.

This register is used for configuring the CAN module in terms of interrupts, parity, debug-mode behavior etc.

**Figure 27-18. CAN_CTL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | RESERVED | RESERVED |
| R-0h | | | | | | R-0h | R-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | DE3 | DE2 | DE1 | IE1 | INITDBG |
| R-0h | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| SWR | RESERVED | PMD | | | | ABO | IDS |
| R/W-0h | R-0h | R/W-5h | | | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Test | CCE | DAR | RESERVED | EIE | SIE | IE0 | Init |
| R/W-0h | R/W-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-1h |

**Table 27-10. CAN_CTL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-26 | RESERVED | R | 0h | Reserved |
| 25 | RESERVED | R | 0h | Reserved |
| 24 | RESERVED | R | 0h | Reserved |
| 23-21 | RESERVED | R | 0h | Reserved |
| 20 | DE3 | R/W | 0h | Enable DMA request line for IF3<br>0 Disabled<br>1 Enabled<br>Note: A pending DMA request for IF3 remains active until first access to one of the IF3 registers.<br>Reset type: SYSRSn |
| 20 | RESERVED | R | 0h | Reserved |
| 19 | DE2 | R/W | 0h | Enable DMA request line for IF2<br>0 Disabled<br>1 Enabled<br>Note: A pending DMA request for IF1 remains active until first access to one of the IF2 registers.<br>Reset type: SYSRSn |
| 19 | RESERVED | R | 0h | Reserved |
| 18 | DE1 | R/W | 0h | Enable DMA request line for IF1<br>0 Disabled<br>1 Enabled<br>Note: A pending DMA request for IF1 remains active until first access to one of the IF1 registers.<br>Reset type: SYSRSn |
| 18 | RESERVED | R | 0h | Reserved |

**Table 27-10. CAN_CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 15 | SWR | R/W | 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>This bit is EALLOW protected.<br>Note: This bit is write-protected by Init bit<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 |

**Table 27-10. CAN_CTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 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 RxOk, TxOk and LEC bits cannot generate an interrupt.<br>1 RxOk, TxOk and LEC can generate an interrupt on the CANINT0 line<br>Reset type: SYSRSn |
| 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 |

### 27.16.1.1.2 CAN_ES Register (Offset = 4h) [reset = 7h]

CAN_ES is shown in Figure 27-19 and described in Table 27-11.

Return to 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 27-19. CAN_ES Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | 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 27-11. 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 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 |

**Table 27-11. CAN_ES Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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.<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 |

#### 27.16.1.1.3 CAN_ERRC Register (Offset = 8h) [reset = 0h]

CAN_ERRC is shown in Figure 27-20 and described in Table 27-12.

Return to Summary Table.

This register reflects the value of the Transmit and Receive error counters

**Figure 27-20. CAN_ERRC Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RP | REC | | | | | | | TEC | | | | | | | |
| R-0h | R-0h | | | | | | | R-0h | | | | | | | |

**Table 27-12. 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 |

#### 27.16.1.1.4 CAN_BTR Register (Offset = Ch) [reset = 2301h]

CAN_BTR is shown in Figure 27-21 and described in Table 27-13.

Return to 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 27-21. CAN_BTR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | BRPE | | | |
| R-0h | | | | R/W-0h | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | TSEG2 | | | TSEG1 | | | |
| R-0h | R/W-2h | | | R/W-3h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SJW | | BRP | | | | | |
| R/W-0h | | R/W-1h | | | | | |

**Table 27-13. 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 |

**Table 27-13. CAN_BTR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5-0 | BRP | R/W | 1h | Baud Rate Prescaler-<br><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.<br><br>Valid programmed values are 0 to 63.<br><br>The actual BRP value interpreted for the Bit Timing will be the programmed BRP value + 1.<br><br>Note: This bit is Write Protected by CCE bit.<br>Reset type: SYSRSn |

#### 27.16.1.1.5 CAN_INT Register (Offset = 10h) [reset = 0h]

CAN_INT is shown in Figure 27-22 and described in Table 27-14.

Return to Summary Table.

This register is used to identify the source of the interrupt(s).

**Figure 27-22. CAN_INT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | INT1ID | | | | | | | | INT0ID | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | R-0h | | | | | | | | R-0h | | | | | | | | | | | | | | | |

**Table 27-14. 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 INT0ID 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 |

### 27.16.1.1.6 CAN_TEST Register (Offset = 14h) [reset = 0h]

CAN_TEST is shown in Figure 27-23 and described in Table 27-15.

Return to 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 27-23. CAN_TEST Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | RDA | EXL |
| R-0h | | | | | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RX | TX | | LBACK | SILENT | RESERVED | | |
| R-0h | R/W-0h | | R/W-0h | R/W-0h | R-0h | | |

**Table 27-15. 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 |

**Table 27-15. CAN_TEST Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 4 | LBACK | R/W | 0h | Loop Back Mode:<br>0 Disabled.<br>1 Enabled.<br>Reset type: SYSRSn |
| 3 | SILENT | R/W | 0h | Silent Mode:<br>0 Disabled.<br>1 Enabled.<br>Reset type: SYSRSn |
| 2-0 | RESERVED | R | 0h | Reserved |

#### 27.16.1.1.7 CAN_PERR Register (Offset = 1Ch) [reset = 100h]

CAN_PERR is shown in Figure 27-24 and described in Table 27-16.

Return to 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 27-24. CAN_PERR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | WORD_NUM | | | MSG_NUM | | | | | | | |
| R-0h | | | | | R-1h | | | R-0h | | | | | | | |

**Table 27-16. CAN_PERR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-11 | RESERVED | R | 0h | Reserved |
| 10-8 | WORD_NUM | R | 1h | 0x01-0x05 Word number where parity error has been detected.<br>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 | 0h | 0x01-0x21 Mailbox number where parity error has been detected<br>Reset type: SYSRSn |

#### 27.16.1.1.8 CAN_RAM_INIT Register (Offset = 40h) [reset = 5h]

CAN_RAM_INIT is shown in Figure 27-25 and described in Table 27-17.

Return to Summary Table.

This register is used to initialize the Mailbox RAM. It clears the entire mailbox RAM, including the MsgVal bits.

**Figure 27-25. 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 27-17. 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 |

#### 27.16.1.1.9 CAN_GLB_INT_EN Register (Offset = 50h) [reset = 0h]

CAN_GLB_INT_EN is shown in Figure 27-26 and described in Table 27-18.

Return to Summary Table.

This register is used to enable the interrupt lines to the PIE.

**Figure 27-26. 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 27-18. 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 |

#### 27.16.1.1.10 CAN_GLB_INT_FLG Register (Offset = 54h) [reset = 0h]

CAN_GLB_INT_FLG is shown in Figure 27-27 and described in Table 27-19.

Return to Summary Table.

This register indicates if and when the interrupt line to the PIE is active.

**Figure 27-27. 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 27-19. 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 |

#### 27.16.1.1.11 CAN_GLB_INT_CLR Register (Offset = 58h) [reset = 0h]

CAN_GLB_INT_CLR is shown in Figure 27-28 and described in Table 27-20.

Return to Summary Table.

This register is used to clear the interrupt to the PIE.

**Figure 27-28. 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 27-20. 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 |

#### 27.16.1.1.12 CAN_ABOTR Register (Offset = 80h) [reset = 0h]

CAN_ABOTR is shown in Figure 27-29 and described in Table 27-21.

Return to Summary Table.

This register is used to introduce a variable delay before the Bus-off recovery sequence is started.

**Figure 27-29. 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 27-21. CAN_ABOTR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | ABO_Time | R/W | 0h | Auto-Bus-On Timer<br><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><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><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><br>NOTE: During Debug mode, running Auto-Bus-On timer will be paused.<br>Reset type: SYSRSn |

#### 27.16.1.1.13 CAN_TXRQ_X Register (Offset = 84h) [reset = 0h]

CAN_TXRQ_X is shown in Figure 27-30 and described in Table 27-22.

Return to 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 27-30. 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 27-22. 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 |

#### 27.16.1.1.14 CAN_TXRQ_21 Register (Offset = 88h) [reset = 0h]

CAN_TXRQ_21 is shown in Figure 27-31 and described in Table 27-23.

Return to Summary Table.

The bits in this register indicate if a transmission has been requested for a mailbox.

**Figure 27-31. 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 27-23. 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 |

#### 27.16.1.1.15 CAN_NDAT_X Register (Offset = 98h) [reset = 0h]

CAN_NDAT_X is shown in Figure 27-32 and described in Table 27-24.

Return to 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 27-32. 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 27-24. 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 |

#### 27.16.1.1.16 CAN_NDAT_21 Register (Offset = 9Ch) [reset = 0h]

CAN_NDAT_21 is shown in Figure 27-33 and described in Table 27-25.

Return to Summary Table.

The bits in this register indicate if the message handler or the CPU has written new data into the data portion of this mailbox.

**Figure 27-33. 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 27-25. 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 |

### 27.16.1.1.17 CAN_IPEN_X Register (Offset = ACh) [reset = 0h]

CAN_IPEN_X is shown in Figure 27-34 and described in Table 27-26.

Return to 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 27-34. 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 27-26. 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 |

#### 27.16.1.1.18 CAN_IPEN_21 Register (Offset = B0h) [reset = 0h]

CAN_IPEN_21 is shown in Figure 27-35 and described in Table 27-27.

Return to Summary Table.

The bits in this register indicate if an interrupt is pending for the corresponsding mailbox.

**Figure 27-35. 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 27-27. 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 |

### 27.16.1.1.19 CAN_MVAL_X Register (Offset = C0h) [reset = 0h]

CAN_MVAL_X is shown in Figure 27-36 and described in Table 27-28.

Return to 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 27-36. 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 27-28. 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 |

#### 27.16.1.1.20 CAN_MVAL_21 Register (Offset = C4h) [reset = 0h]

CAN_MVAL_21 is shown in Figure 27-37 and described in Table 27-29.

Return to Summary Table.

The bits in this register are used to enable/disable mailboxes as needed.

**Figure 27-37. 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 27-29. 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 |

#### 27.16.1.1.21 CAN_IP_MUX21 Register (Offset = D8h) [reset = 0h]

CAN_IP_MUX21 is shown in Figure 27-38 and described in Table 27-30.

Return to 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. The IntPnd bit of a specific message object can be set or reset by the CPU via the IF1/IF2 Interface Register sets, or by the Message Handler after reception or successful transmission of a frame. This will also affect the Int0ID or Int1ID flags in the Interrupt Register.

**Figure 27-38. 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 27-30. 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 |

### 27.16.1.1.22 CAN_IF1CMD Register (Offset = 100h) [reset = 1h]

CAN_IF1CMD is shown in Figure 27-39 and described in Table 27-31.

Return to Summary Table.

The IF1/IF2 Command Registers configure and initiate the transfer between the IF1/IF2 Register sets and the Message RAM. It is configurable which portions of the message object should be transferred. A transfer is started when the CPU writes the message number to bits [7:0] of the IF1/IF2 Command Register. With this write operation, the Busy bit is automatically set to '1' to indicate that a transfer is in progress. After 4 to 14 clock cycles, the transfer between the Interface Register and the Message RAM will be completed and the Busy bit is cleared. The maximum number of cycles is needed when the message transfer coincides with a CAN message transmission, acceptance filtering, or message storage.

If the CPU writes to both IF1/IF2 Command Registers consecutively (request of a second transfer while first transfer is still in progress), the second transfer will start after the first one has been completed.

Note: While Busy bit is one, IF1/IF2 Register sets are write protected.

Note: For debug support, the auto clear functionality of the IF1/IF2 Command Registers (clear of DMAactive flag by R/W, for devices with DMA support) is disabled during Debug/Suspend mode.

Note: If an invalid Message Number is written to bits [7:0] of the IF1/IF2 Command Register, the Message Handler may access an implemented (valid) message object instead.

**Figure 27-39. CAN_IF1CMD Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DIR | Mask | Arb | Control | ClrIntPnd | TXRQST | DATA_A | DATA_B |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Busy | DMAactive | RESERVED | | | | | |
| R-0h | R/W-0h | R-0h | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MSG_NUM | | | | | | | |
| R/W-1h | | | | | | | |

**Table 27-31. 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.<br>1 Direction = Write: Transfer direction is 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 |

**Table 27-31. 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>Direction = Write: The Mask bits (Identifier Mask + MDir + MXtd) will be transferred from theIF1/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>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/IF2 Register set.<br>Direction = Write: The Message Control bits will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]). If the TxRqst/NewDat bit in this register (Bit [18]) is set, the TxRqst/ NewDat bit in the IF1/IF2 Message Control Register will be ignored.<br>Note: This bit is write protected by 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. Copying of IntPnd flag from IF1/IF2 registers to Message RAM can only be controlled by the Control flag (Bit [20]).<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |

**Table 27-31. CAN_IF1CMD Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18 | TXRQST | R/W | 0h | Access Transmission Request 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 |
| 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 |

**Table 27-31. CAN_IF1CMD Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 14 | DMAactive | R/W | 0h | DMA trigger status due to IF1 update.<br>0 No IF1 DMA request is active.<br>1 DMA is requested after a completed transfer between IF1 and the message RAM. The DMA request remains active until the first read or write to one of the IF1 registers<br>an exception is a write to Message Number (Bits [7:0]) when DMAactive is one.<br>Note: Due to the auto reset feature of the DMAactive bit, this bit has to be set for each subsequent DMA cycle separately.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 14 | RESERVED | R | 0h | Reserved |
| 13-8 | RESERVED | R | 0h | Reserved |
| 7-0 | MSG_NUM | R/W | 1h | Number of message object in Message RAM which is used for data transfer<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 |

### 27.16.1.1.23 CAN_IF1MSK Register (Offset = 104h) [reset = FFFFFFFFh]

CAN_IF1MSK is shown in Figure 27-40 and described in Table 27-32.

Return to 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 27-40. 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 27-32. CAN_IF1MSK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MXtd | R/W | 1h | Mask Extended Identifier<br>0 The extended identifier bit (IDE) has no effect on the acceptance filtering.<br>1 The extended identifier bit (IDE) is used for acceptance filtering.<br>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 |

### 27.16.1.1.24 CAN_IF1ARB Register (Offset = 108h) [reset = 0h]

CAN_IF1ARB is shown in Figure 27-41 and described in Table 27-33.

Return to 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 27-41. 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 27-33. 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. This bit must also be reset before the identifier ID[28:0], the control bits Xtd, Dir or DLC[3:0] are modified, or if the messages object is no longer required.<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 |

**Table 27-33. CAN_IF1ARB Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 29 | Dir | R/W | 0h | Message Direction<br><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><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><br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 28-0 | ID | R/W | 0h | Message Identifier<br><br>ID[28:0] 29-bit Identifier ("Extended Frame")<br><br>ID[28:18] 11-bit Identifier ("Standard Frame")<br><br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |

### 27.16.1.1.25 CAN_IF1MCTL Register (Offset = 10Ch) [reset = 0h]

CAN_IF1MCTL is shown in Figure 27-42 and described in Table 27-34.

Return to 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 27-42. 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 27-34. 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 |

**Table 27-34. CAN_IF1MCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 27-34. CAN_IF1MCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |

#### 27.16.1.1.26 CAN_IF1DATA Register (Offset = 110h) [reset = 0h]

CAN_IF1DATA is shown in Figure 27-43 and described in Table 27-35.

Return to 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 27-43. 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 27-35. 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 |

### 27.16.1.1.27 CAN_IF1DATB Register (Offset = 114h) [reset = 0h]

CAN_IF1DATB is shown in Figure 27-44 and described in Table 27-36.

Return to 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 27-44. 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 27-36. 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 |

#### 27.16.1.1.28 CAN_IF2CMD Register (Offset = 120h) [reset = 1h]

CAN_IF2CMD is shown in Figure 27-45 and described in Table 27-37.

Return to Summary Table.

The IF1/IF2 Command Registers configure and initiate the transfer between the IF1/IF2 Register sets and the Message RAM. It is configurable which portions of the message object should be transferred. A transfer is started when the CPU writes the message number to bits [7:0] of the IF1/IF2 Command Register. With this write operation, the Busy bit is automatically set to '1' to indicate that a transfer is in progress. After 4 to 14 clock cycles, the transfer between the Interface Register and the Message RAM will be completed and the Busy bit is cleared. The maximum number of cycles is needed when the message transfer coincides with a CAN message transmission, acceptance filtering, or message storage.

If the CPU writes to both IF1/IF2 Command Registers consecutively (request of a second transfer while first transfer is still in progress), the second transfer will start after the first one has been completed.

Note: While Busy bit is one, IF1/IF2 Register sets are write protected.

Note: For debug support, the auto clear functionality of the IF1/IF2 Command Registers (clear of DMAactive flag by R/W, for devices with DMA support) is disabled during Debug/Suspend mode.

Note: If an invalid Message Number is written to bits [7:0] of the IF1/IF2 Command Register, the Message Handler may access an implemented (valid) message object instead.

**Figure 27-45. CAN_IF2CMD Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DIR | Mask | Arb | Control | ClrIntPnd | TxRqst | DATA_A | DATA_B |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Busy | DMAactive | RESERVED | | | | | |
| R-0h | R/W-0h | R-0h | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MSG_NUM | | | | | | | |
| R/W-1h | | | | | | | |

**Table 27-37. 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.<br>1 Direction = Write: Transfer direction is 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 |

**Table 27-37. 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 theIF1/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>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/IF2 Register set.<br>1 Direction = Write: The Message Control bits will be transferred from the IF1/IF2 Register set to the message object addressed by Message Number (Bits [7:0]). If the TxRqst/NewDat bit in this register (Bit [18]) is set, the TxRqst/ NewDat bit in the IF1/IF2 Message Control Register will be ignored.<br>Note: This bit is write protected by 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. Copying of IntPnd flag from IF1/IF2 registers to Message RAM can only be controlled by the Control flag (Bit [20]).<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |

**Table 27-37. CAN_IF2CMD Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 18 | TxRqst | R/W | 0h | Access Transmission Request 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 |
| 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 |

**Table 27-37. CAN_IF2CMD Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 14 | DMAactive | R/W | 0h | DMA trigger status due to IF2 update.<br>0 No IF2 DMA request is active.<br>1 DMA is requested after a completed transfer between IF2 and the message RAM. The DMA request remains active until the first read or write to one of the IF2 registers<br>an exception is a write to Message Number (Bits [7:0]) when DMAactive is one.<br>Note: Due to the auto reset feature of the DMAactive bit, this bit has to be set for each subsequent DMA cycle separately.<br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 14 | RESERVED | R | 0h | Reserved |
| 13-8 | RESERVED | R | 0h | Reserved |
| 7-0 | MSG_NUM | R/W | 1h | Number of message object in Message RAM which is used for data transfer<br>0x00 Invalid message number<br>0x01-0x20 Valid message numbers<br>0x21-0xFF Invalid message numbers<br>Reset type: SYSRSn |

### 27.16.1.1.29 CAN_IF2MSK Register (Offset = 124h) [reset = FFFFFFFFh]

CAN_IF2MSK is shown in Figure 27-46 and described in Table 27-38.

Return to 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 27-46. 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 27-38. CAN_IF2MSK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MXtd | R/W | 1h | Mask Extended Identifier<br>0 The extended identifier bit (IDE) has no effect on the acceptance filtering.<br>1 The extended identifier bit (IDE) is used for acceptance filtering.<br>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 |

### 27.16.1.1.30 CAN_IF2ARB Register (Offset = 128h) [reset = 0h]

CAN_IF2ARB is shown in Figure 27-47 and described in Table 27-39.

Return to 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 27-47. 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 27-39. 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. This bit must also be reset before the identifier ID[28:0], the control bits Xtd, Dir or DLC[3:0] are modified, or if the messages object is no longer required.<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 |

**Table 27-39. CAN_IF2ARB Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 29 | Dir | R/W | 0h | Message Direction<br><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><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><br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |
| 28-0 | ID | R/W | 0h | Message Identifier<br><br>ID[28:0] 29-bit Identifier ("Extended Frame")<br><br>ID[28:18] 11-bit Identifier ("Standard Frame")<br><br>Note: This bit is write protected by Busy bit.<br>Reset type: SYSRSn |

### 27.16.1.1.31 CAN_IF2MCTL Register (Offset = 12Ch) [reset = 0h]

CAN_IF2MCTL is shown in Figure 27-48 and described in Table 27-40.

Return to 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 27-48. 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 27-40. 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 |

**Table 27-40. CAN_IF2MCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

**Table 27-40. CAN_IF2MCTL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |

#### 27.16.1.1.32 CAN_IF2DATA Register (Offset = 130h) [reset = 0h]

CAN_IF2DATA is shown in Figure 27-49 and described in Table 27-41.

Return to 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 27-49. 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 27-41. 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 |

#### 27.16.1.1.33 CAN_IF2DATB Register (Offset = 134h) [reset = 0h]

CAN_IF2DATB is shown in Figure 27-50 and described in Table 27-42.

Return to 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 27-50. 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 27-42. 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 |

### 27.16.1.1.34 CAN_IF3OBS Register (Offset = 140h) [reset = 0h]

CAN_IF3OBS is shown in Figure 27-51 and described in Table 27-43.

Return to 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 27-51. 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 27-43. 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 |

**Table 27-43. CAN_IF3OBS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 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 |
| 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 |

### 27.16.1.1.35 CAN_IF3MSK Register (Offset = 144h) [reset = FFFFFFFFh]

CAN_IF3MSK is shown in Figure 27-52 and described in Table 27-44.

Return to Summary Table.

This register provides a window to the acceptance mask for the chosen mailbox.

**Figure 27-52. 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 27-44. CAN_IF3MSK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | MXtd | R | 1h | Mask Extended Identifier<br>0 The extended identifier bit (IDE) has no effect on the acceptance filtering.<br>1 The extended identifier bit (IDE) 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 |

### 27.16.1.1.36 CAN_IF3ARB Register (Offset = 148h) [reset = 0h]

CAN_IF3ARB is shown in Figure 27-53 and described in Table 27-45.

Return to Summary Table.

The bits of the IF3 Arbitration Register mirrors the arbitration bits of a message object.

**Figure 27-53. 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 27-45. 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. This bit must also be reset before the identifier ID[28:0], the control bits Xtd, Dir or DLC[3:0] are modified, or if the messages object is no longer required.<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 |

### 27.16.1.1.37 CAN_IF3MCTL Register (Offset = 14Ch) [reset = 0h]

CAN_IF3MCTL is shown in Figure 27-54 and described in Table 27-46.

Return to Summary Table.

The bits of the IF3 Message Control Register mirrors the message control bits of a message object.

**Figure 27-54. 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 27-46. 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 |

**Table 27-46. 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 |

#### 27.16.1.1.38 CAN_IF3DATA Register (Offset = 150h) [reset = 0h]

CAN_IF3DATA is shown in Figure 27-55 and described in Table 27-47.

Return to Summary Table.

This register provides a window to the data bytes of the CAN message.

**Figure 27-55. 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 27-47. 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 |

### 27.16.1.1.39 CAN_IF3DATB Register (Offset = 154h) [reset = 0h]

CAN_IF3DATB is shown in Figure 27-56 and described in Table 27-48.

Return to Summary Table.

This register provides a window to the data bytes of the CAN message.

**Figure 27-56. 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 27-48. 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 |

#### 27.16.1.1.40 CAN_IF3UPD Register (Offset = 160h) [reset = 0h]

CAN_IF3UPD is shown in Figure 27-57 and described in Table 27-49.

Return to 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 27-57. 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 27-49. 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 |

# Local Interconnect Network (LIN) Module

This chapter describes the local interconnect network (LIN) module. Since this module can also operate like a conventional serial communications interface (SCI) port, it is referred to as the SCI/LIN module in this document. In SCI compatibility mode, it is functionally compatible to the standalone SCI module on this devices. However, since the SCI/LIN module uses a different register/bit structure, code written for this module cannot be directly ported to the standalone SCI module and vice versa.

**Topic** **Page**

## 28.1 Introduction and Features

The SCI/LIN is compliant to the LIN 2.1 protocol specified in the *LIN Specification Package*. The SCI/LIN module can be programmed to work either as an SCI or as a LIN. The SCI's hardware features are augmented to achieve LIN functionality

The SCI module is a universal asynchronous receiver-transmitter that implements the standard non-return to zero format. The SCI can be used to communicate, for example, through an RS-232 port or over a K-line.

The LIN standard is based on the SCI (UART) serial data link format. The communication concept is single-master/multiple-slave with a message identification for multi-cast transmission between any network nodes.

### *28.1.1 LIN Features*

The following are the features of the LIN module:

- Compatibility with LIN 1.3, 2.0, and 2.1protocols
- Configurable Baud Rate up to 20 Kbits/s
- Two external pins: LINRX and LINTX.
- Multi-buffered receive and transmit units
- Identification masks for message filtering
- Automatic master header generation
  - Programmable synchronization break field
  - Synchronization field
  - Identifier field
- Slave automatic synchronization
  - Synchronization break detection
  - Optional baud rate update
  - Synchronization validation
- $2^{31}$ programmable transmission rates with 7 fractional bits
- Wakeup on LINRX dominant level from transceiver
- Automatic wakeup support
  - Wakeup signal generation
  - Expiration times on wakeup signals
- Automatic bus idle detection
- Error detection
  - Bit error
  - Bus error
  - No-response error
  - Checksum error
  - Synchronization field error
  - Parity error
- Capability to use Direct Memory Access (DMA) for transmit and receive data.
- 2 Interrupt lines with priority encoding for:
  - Receive
  - Transmit
  - ID, error, and status
- Support for LIN 2.0 checksum
- Enhanced synchronizer finite state machine (FSM) support for frame processing

- Enhanced handling of extended frames
- Enhanced baud rate generator
- Update wakeup/go to sleep

### 28.1.2 Block Diagram

The SCI/LIN module contains core SCI block with added sub-blocks to support LIN protocol.

Three Major components of the SCI Module are:

- Transmitter
- Baud Clock Generator
- Receiver

**Transmitter** (TX) contains two major registers to perform the double- buffering:

- The transmitter data buffer register (SCITD) contains data loaded by the CPU to be transferred to the shift register for transmission.
- The transmitter shift register (SCITXSHF) loads data from the data buffer (SCITD) and shifts data onto the LINTX pin, one bit at a time.

**Baud Clock Generator**

- A programmable baud generator produces either a baud clock scaled from the input clock.

**Receiver** (RX) contains two major registers to perform the double- buffering:

- The receiver shift register (SCIRXSHF) shifts data in from the LINRX pin one bit at a time and transfers completed data into the receive data buffer.
- The receiver data buffer register (SCIRD) contains received data transferred from the receiver shift register

The SCI receiver and transmitter are double-buffered, and each has its own separate enable and interrupt bits. The receiver and transmitter may each be operated independently or simultaneously in full duplex mode.

To ensure data integrity, the SCI checks the data it receives for breaks, parity, overrun, and framing errors. The bit rate (baud) is programmable to over 16 million different rates through a 24-bit baud-select register. Figure 28-1 shows the detailed SCI block diagram.

The SCI/LIN module is based on the standalone SCI with the addition of an error detector (parity calculator, checksum calculator, and bit monitor), a mask filter, a synchronizer, and a multi-buffered receiver and transmitter. The SCI interface, the DMA control subblocks and the baud generator are modified as part of the hardware enhancements for LIN compatibility. Figure 28-2 shows the SCI/LIN block diagram.

**Figure 28-1. SCI Block Diagram**

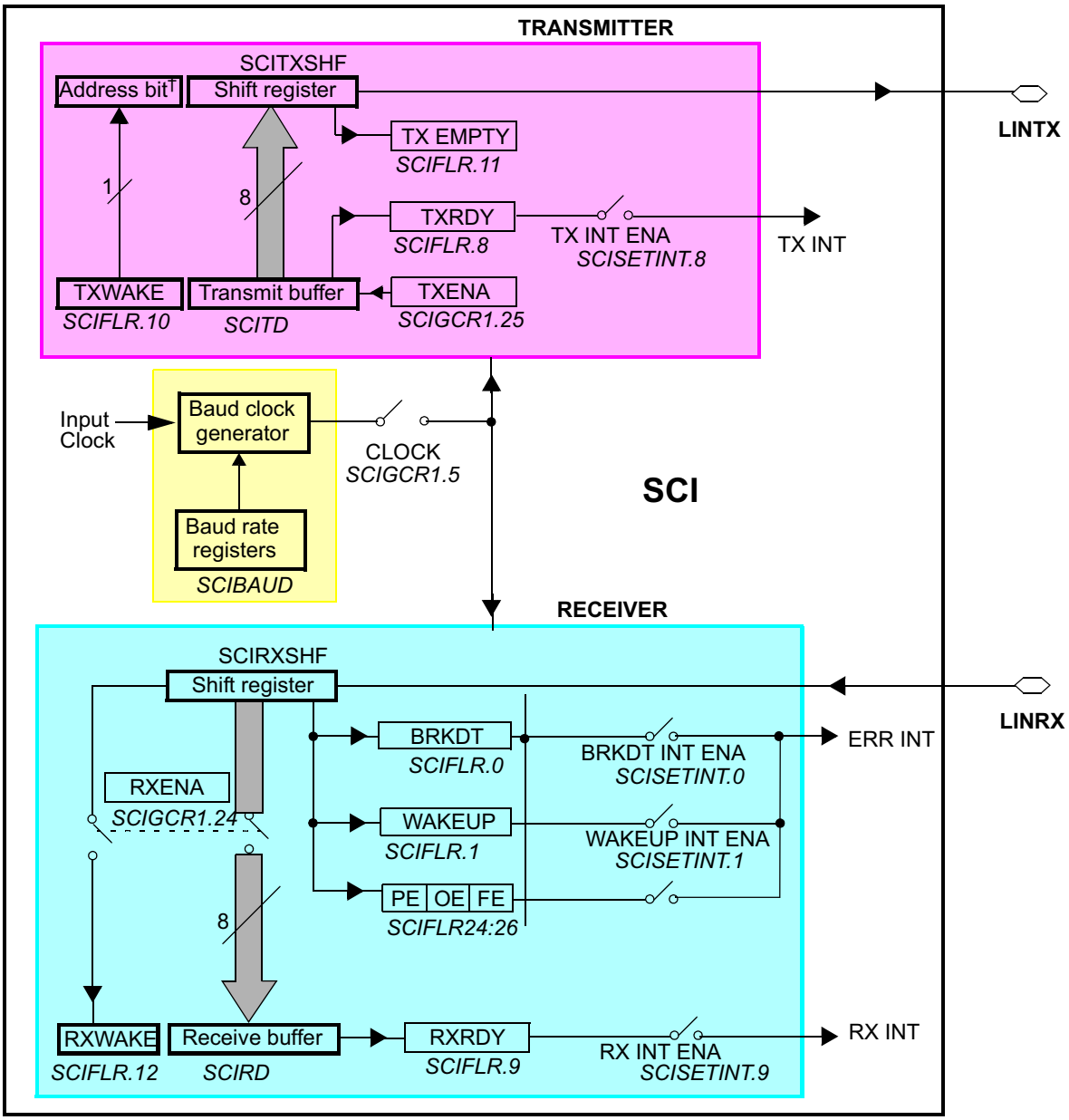
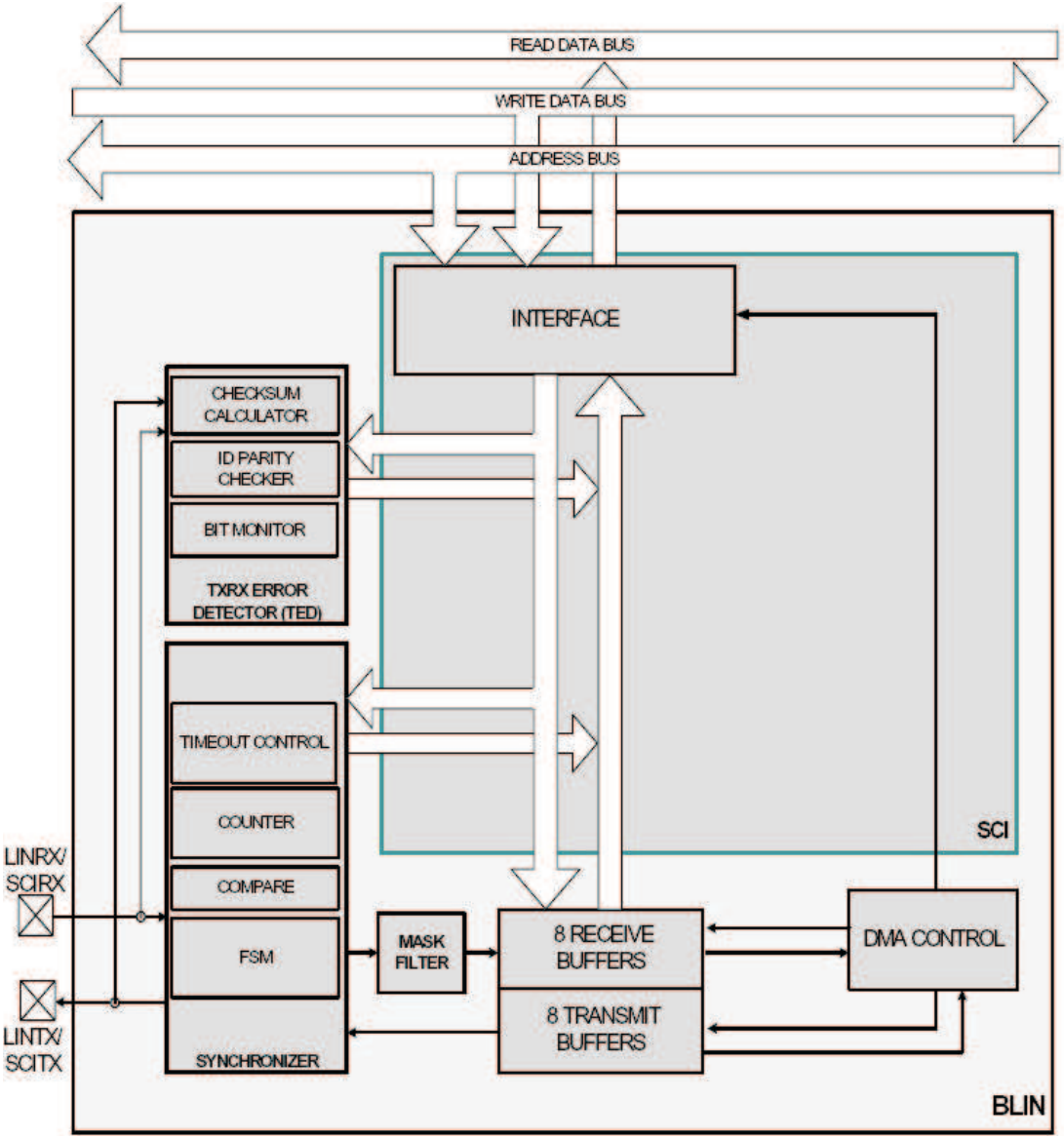


**Figure 28-2. SCI/LIN Block Diagram**




## 28.2 LIN Communication Formats

The SCI/LIN module can be used in LIN mode or SCI mode. The enhancements for baud generation, DMA controls and additional receive/transmit buffers necessary for LIN mode operation are also part of the enhanced buffered SCI module. LIN mode is selected by enabling LIN MODE bit in SCIGCR1 register.

**NOTE:** The SCI/LIN is built around the SCI platform and uses a similar sampling scheme: 16 samples for each bit with majority vote on samples 8, 9, and 10. For START bit, the first three samples are used.

The SCI/LIN control registers are located at the SCI/LIN base address. For a detailed description of each register, see the Registers Section.

### *28.2.1 LIN Standards*

For compatibility with LIN2.0 standard the following additional features are implemented over LIN1.3:

i. Support for LIN 2.0 checksum
ii. Enhanced synchronizer FSM support for frame processing
iii. Enhanced handling of extended frames
iv. Enhanced baudrate generator
v. Update wakeup/go to sleep

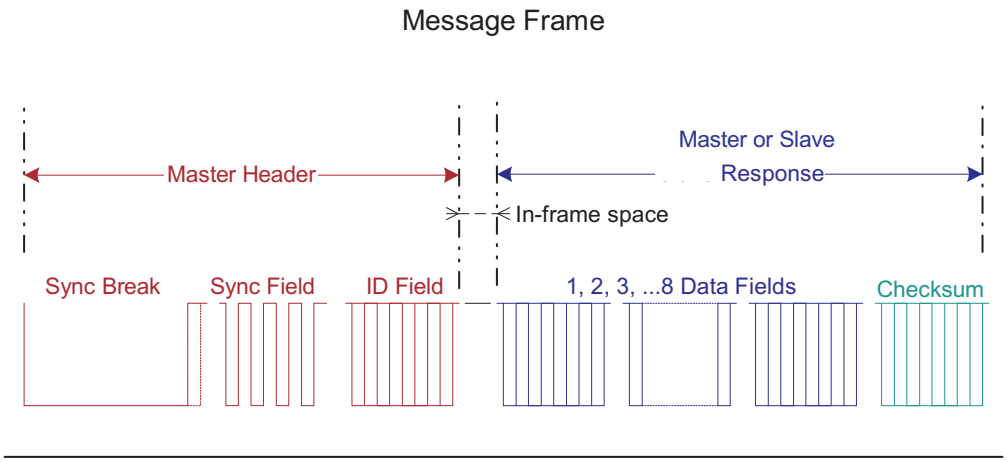
The LIN module covers the CPU performance-consuming features, defined in the *LIN Specification Package* Revision 1.3 and 2.0 by hardware.

The Master Mode of LIN module is compatible with LIN 2.1 standard.

### 28.2.2 Message Frame

The LIN protocol defines a message frame format, illustrated in Figure 28-3. Each frame includes one master header, one response, one in-frame response space, and inter-byte spaces. In-frame-response and inter-byte spaces may be 0.

**Figure 28-3. LIN Protocol Message Frame Format: Master Header and Slave Response**




There is no arbitration in the definition of the LIN protocol; therefore, multiple slave nodes responding to a header might be detected as an error.

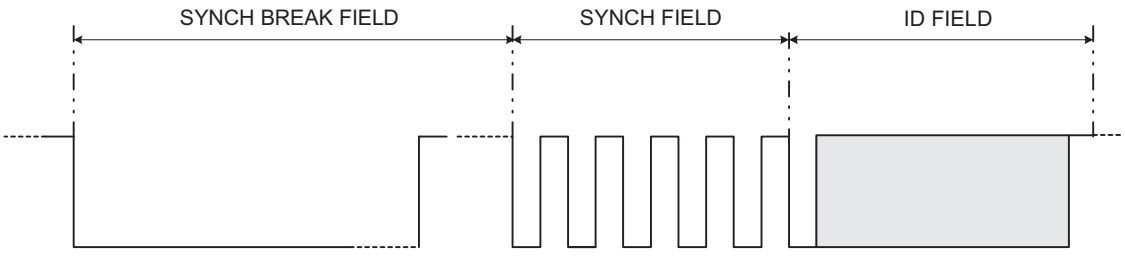
The LIN bus is a single channel wired-AND. The bus has a binary level: either dominant for a value of 0, or recessive for a value of 1.

#### 28.2.2.1 Message Header

The header of a message is initiated by a master (see Figure 28-4) and consists of a three field-sequence:

- The synch break field signaling the beginning of a message
- The synch field conveying bit rate information of the LIN bus
- The ID field denoting the content of a message

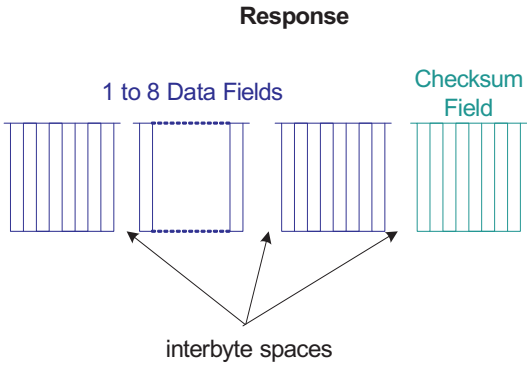
**Figure 28-4. Header 3 Fields: Synch Break, Synch, and ID**



### 28.2.2.2 Response

The format of the response is as illustrated in Figure 28-5. There are two types of fields in a response: data and checksum. The data field consists of exactly one data byte, one start bit, and one stop bit, for a total of 10 bits. The LSB is transmitted first. The checksum field consists of one checksum byte, one start bit and one stop bit. The checksum byte is the inverted modulo-256 sum over all data bytes in the data fields of the response.

**Figure 28-5. Response Format of LIN Message Frame**




The format of the response is a stream of N data fields and one checksum field. Typically N is from 1 to 8, with the exception of the extended command frames (Section 28.2.6). The length N of the response is indicated either with the optional length control bits of the ID Field (this is used in standards earlier than LIN 1.x); see Table 28-1, or by LENGTH value in SCIFORMAT[18:16] register; see Table 28-2. The SCI/LIN module supports response lengths from 1 to 8 bytes in compliance with LIN 2.0.

**Table 28-1. Response Length Info Using IDBYTE Field Bits [5:4] for LIN Standards Earlier than 1.3**

| ID5 | ID4 | Number of Data bytes |
|---|---|---|
| 0 | 0 | 2 |
| 0 | 1 | 2 |
| 1 | 0 | 4 |
| 1 | 1 | 8 |

**Table 28-2. Response Length with SCIFORMAT[18:16] Programming**

| SCIFORMAT[18:16] | No. of Bytes |
|---|---|
| 000 | 1 |
| 001 | 2 |
| 010 | 3 |
| 011 | 4 |
| 100 | 5 |
| 101 | 6 |
| 110 | 7 |
| 111 | 8 |

### 28.2.3 Synchronizer

The synchronizer has three major functions in the messaging between master and slave nodes. It generates the master header data stream, it synchronizes to the LIN bus for responding, and it locally detects timeouts. A bit rate is programmed using the prescalers in the BRSR register to match the indicated LIN_speed value in the LIN description file.

The LIN synchronizer will perform the following functions: master header signal generation, slave detection and synchronization to message header with optional baud rate adjustment, response transmission timing and timeout control.

The LIN synchronizer is capable of detecting an incoming break and initializing communication at all times.

### 28.2.4 Baud Rate

The transmission baud rate of any node is configured by the CPU at the beginning; this defines the bit time $T_{bit}$. The bit time is derived from the fields P and M in the baud rate selection register (BRSR). There is an additional 3-bit fractional divider value, field U in the BRSR, which further fine-tunes the data field baud rate.

The ranges for the prescaler values in the BRSR register are:

P = 0, 1, 2, 3, . . . , $2^{24}$ - 1

M = 0, 1, 2, . . . , 15

U = 0, 1, 2, 3, 4, 5, 6, 7

The P, M, and U values in the BRSR register are user programmable. The P and M dividers could be used for both SCI mode and LIN mode to select a baud rate. The U value is an additional 3-bit value determining that "**a TVCLK**" (with **a** = 0,1) is added to each $T_{bit}$ as explained in Section 28.2.4.2. If the ADAPT bit is set and the LIN slave is in adaptive baud rate mode, then all these divider values are automatically obtained during header reception when the synchronization field is measured.

The LIN protocol defines baud rate boundaries as follows:

$1kHz \le F_{LINCLK} \le 20kHz$

All transmitted bits are shifted in and out at $T_{bit}$ periods.

#### 28.2.4.1 Fractional Divider

The M field of the BRSR register modifies the integer prescaler P for finer tuning of the baud rate. The M value adds in increments of 1/16 of the P value.

The bit time, $T_{bit}$ is expressed in terms of the VCLK period $T_{VCLK}$ as follows:

For all P other than 0, and all M,

$$T_{bit} = 16\left(P + 1 + \frac{M}{16}\right)T_{VCLK} \quad (13)$$

For P= 0 : $T_{bit} = 32T_{VCLK}$

Therefore, the LINCLK frequency is given by:

$$F_{LINCLK} = \frac{F_{VCLK}}{16(P + 1 + \frac{M}{16})} \quad \text{For all P other than zero}$$

$$F_{LINCLK} = \frac{F_{VCLK}}{32} \quad \text{For } P = 0 \quad (14)$$

#### 28.2.4.2 Superfractional Divider

The superfractional divider scheme applies to the following modes:

- LIN master mode (synch field + identifier field + response field + checksum field)
- LIN slave mode (response field + checksum field)

#### 28.2.4.3 Superfractional Divider In LIN Mode

Building on the 4-bit fractional divider M (BRSR[27:24], the superfractional divider uses an additional 3-bit modulating value, illustrated in Table 28-3. The sync field (0x55), the identifier field and the response field can all be seen as 8-bit data bytes flanked by a start bit and a stop bit. The bits with a 1 in the table will have an additional VCLK period added to their $T_{bit}$.

**Table 28-3. Superfractional Bit Modulation for LIN Master Mode and Slave Mode (1)**

| BRS[30:28] | Start Bit | D[0] | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] | Stop Bit |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 2 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 3 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 4 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 5 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 6 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |

(1)

1. In LIN master mode bit modulation applies to synch field + identifier field + response field
2. In LIN slave mode bit modulation applies to identifier field + response field

The baud rate will vary over a LIN data field to average according to the BRSR[30:28] value by a *d* fraction of the peripheral internal clock: 0<d<1.

The instantaneous bit time is expressed in terms of $T_{VCLK}$ *as follows:*

For all P other than 0, and all M and d (0 or 1),

$$T^{i}bit = \left[16\left(P + 1 + \frac{M}{16}\right) + d\right]T_{VCLK} \quad (15)$$

For P = 0 $T_{bit} = 32T_{VCLK}$

The averaged bit time is expressed in terms of $T_{VCLK}$ as follows*:*

For all P other than 0, and all M and d (0<d<1),

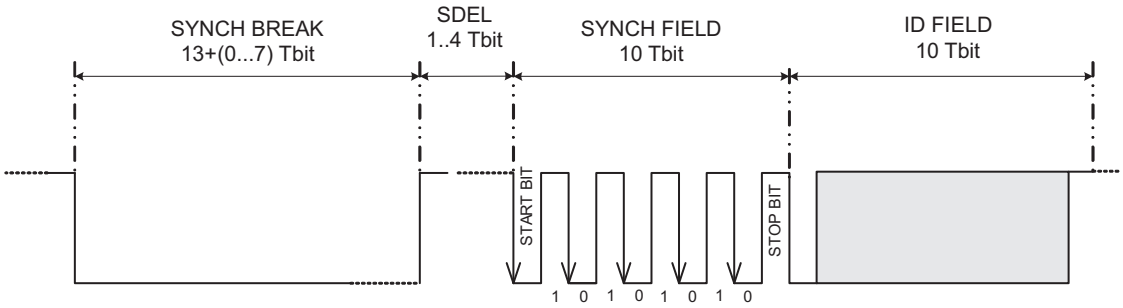
$$T^{a}bit = \left[16\left(P + 1 + \frac{M}{16}\right) + d\right]T_{VCLK} \quad (16)$$

For P = 0 $T_{bit} = 32T_{VCLK}$

### *28.2.5 Header Generation*

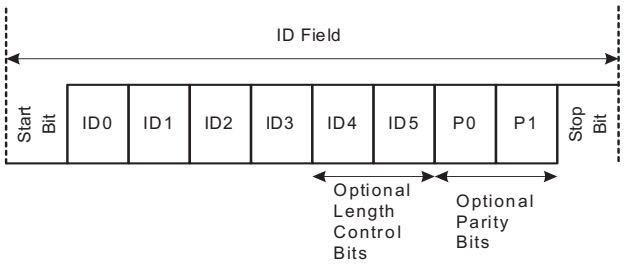
Automatic generation of the LIN protocol header data stream is supported without CPU interaction. The CPU pr the DMA will trigger a message header generation and the LIN state machine will handle the generation itself. A master node initiates header generation on CPU or DMA writes to the IDBYTE in the LINID register. The header is always sent by the master to initiate a LIN communication and consists of three fields: break field, synchronization field, and identification field, as seen in Figure 28-6.

**Figure 28-6. Message Header in Terms of $T_{bit}$**




- The break field consists of two components:
  - The synchronization break (SYNCH BREAK) consists of a minimum of 13 (dominant) low bits to a maximum of 20 dominant bits. The synch break length may be extended from the minimum with the 3-bit SBREAK value in the LINCOMP register.
  - The synchronization break delimiter (SDEL) consists of a minimum of 1 (recessive) high bit to a maximum of 4 recessive bits. The delimiter marks the end of the synchronization break field. The synch break delimiter length depends on the 2-bit SDEL value in the LINCOMP register.
- The synchronization field (SYNCH FIELD) consists of one start bit, byte 0x55, and a stop bit. It is used to convey $T_{bit}$ information and resynchronize LIN bus nodes.
- The identifier field's ID byte may use six bits as an identifier, with optional length control (see the note below), and two optional bits as parity of the identifier. The identifier parity is used and checked if the PARITY ENA bit is set. If length control bits are not used, then there can be a total of 64 identifiers plus parity. If neither length control or parity are used there can be up to 256 identifiers. See Figure 28-7 for an illustration of the ID field.

**Figure 28-7. ID Field**




**NOTE: Optional Control Length Bits**

The control length bits only apply to LIN standards prior to LIN 1.3. IDBYTE field conveys response length information if compliant to standards earlier than LIN1.3. The SCIFORMAT register stores the length of the response for later versions of the LIN protocol.

**NOTE:** If the LIN module, configured as Slave in multi-buffer mode, is in the process of transmitting data while a new header comes in, the module might end up in responding with the data from the previous interrupted response (not the data corresponding to the new ID). To avoid this scenario the following procedure could be used:

1. Check for the Bit Error (BE) during the response transmission. If the BE flag is set, this indicates that a collision has happened on the LIN bus (here because of the new Synch Break).
2. In the Bit Error ISR, configure the TD0 and TD1 registers with the next set of data to be transmitted on a TX Match for the incoming ID. Before writing to TD0/TD1 make sure that there was not already an update because of a Bit Error; otherwise TD0/TD1 might be written twice for one ID.
3. Once the complete ID is received, based on the match, the newly configured data will be transmitted by the node.

#### 28.2.5.1 Event Triggered Frame Handling Proposal

The LIN 2.0 protocol uses event-triggered frames that may occasionally cause collisions. Event-triggered frames have to be handled in software.

If no slave answers to an event triggered frame header, the master node will set the NRE flag, and a NRE interrupt will occur if enabled. If a collision occurs, a frame error and checksum error may arise before the NRE error. Those errors are flagged and the appropriate interrupts will occur, if enabled.

Frame errors and checksum errors depend on the behavior and synchronization of the responding slaves. If the slaves are totally synchronized and stop transmission once the collision occurred, it is possible that only the NRE error is flagged despite the occurrence of a collision. To detect if there has been a reception of one byte before the NRE error is flagged, the BUS BUSY flag can be used as an indicator.

The bus busy flag is set on the reception of the first bit of the header and remains set until the header reception is complete, and again is set on the reception of the first bit of the response. In the case of a collision the flag is cleared in the same cycle as the NRE flag is set.

Software could implement the following sequence:

- Once the reception of the header is done (poll for RXID flag), wait for the bus busy flag to get set or NRE flag to get set.
- If bus busy flag is not set before NRE flag, then it is a true no response case (no data has been transmitted onto the bus).
- If bus busy flag gets set, then wait for NRE flag to get set or for successful reception. If NRE flag is set, then in this case a collision has occurred on the bus.

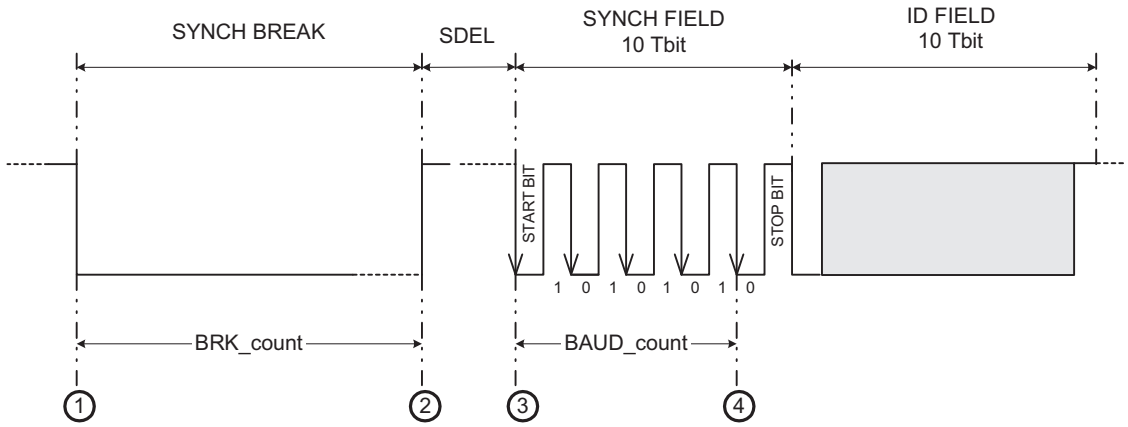
Even in the case of a collision, the received (corrupted) data is accessible in the RX buffers; registers LINRD0 and LINRD1.

### 28.2.5.2 Header Reception and Adaptive Baud Rate

A slave node baud rate can optionally be adjusted to the detected bit rate as an option to the LIN module. The adaptive baud rate option is enabled by setting the ADAPT bit. During header reception, a slave measures the baud rate during detection of the synch field. If ADAPT bit is set, then the measured baud rate is compared to the slave node's programmed baud rate and adjusted to the LIN bus baud rate if necessary.

The LIN synchronizer determines two measurements: BRK_count and BAUD_count (Figure 28-8). These values are always calculated during the Header reception for synch field validation (Figure 28-9).

**Figure 28-8. Measurements for Synchronization**




By measuring the values BRK_count and BAUD_count, a valid synch break sequence can be detected as described in Figure 28-9. The four numbered events in Figure 28-8 signal the start/stop of the synchronizer counter. The synchronizer counter uses VCLK as the time base.

The synchronizer counter is used to measure the synch break relative to the detecting node $T_{bit}$. For a slave node receiving the synch break, a threshold of 11 $T_{bit}$ is used as required by the LIN protocol. For detection of the dominant data stream of the synch break, the synchronizer counter is started on a falling edge and stopped on a rising edge of the LINRX. On detection of the synch break delimiter, the synchronizer counter value is saved and then reset.

On detection of five consecutive falling edges, the BAUD_count is measured. Bit timing calculation and consistency to required accuracy is implemented following the recommendations of LIN revision 2.0. A slave node can calculate a single $T_{bit}$ time by division of BAUD_count by 8. In addition, for consistency between the detected edges the following is evaluated:

BAUD_count + BAUD_count » 2 + BAUD_count » 3 ≤ BRK_count

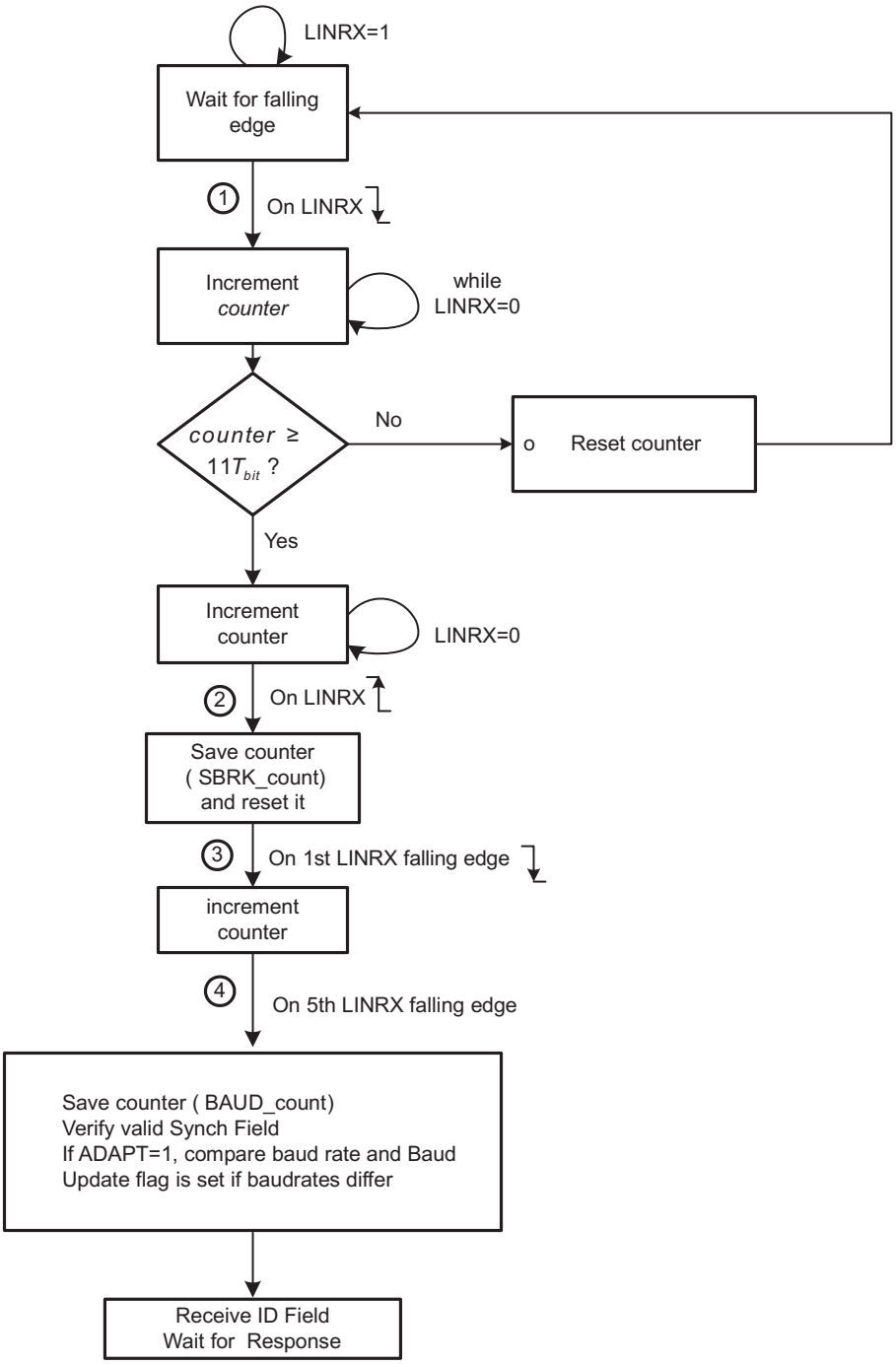
The BAUD_count value is shifted 3 times to the right and rounded using the first insignificant bit to obtain a $T_{bit}$ unit. If the ADAPT bit is set, then the detected baud rate is compared to the programmed baud rate.

During the header reception processing as illustrated in Figure 28-9, if the measured BRK_count value is less than 11 $T_{bit}$, the synch break is not valid according to the protocol for a fixed rate. If the ADAPT bit is set, then the MBRS is used for measuring BRK_count and BAUD_count values and automatically adjusts to any allowed LIN bus rate (refer to *LIN Specification Package 2.0*).

**NOTE:** In adaptive mode the MBRS divider should be set to allow a maximum baud rate that is not more than 10% above the expected operating baud rate in the LIN network. Otherwise, a 0x00 data byte could mistakenly be detected as a synch break.

The break-threshold relative to the slave node is 11 $T_{bit}$. The break is 13 $T_{bit}$ as specified in LIN version 1.3.

**Figure 28-9. Synchronization Validation Process and Baud Rate Adjustment**




If the synch field is not detected within the given tolerances, the inconsistent-synch-field-error (ISFE) flag will be set. An ISFE interrupt will be generated, if enabled by its respective bit in the SCISETINT register. The ID byte should be received after the synch field validation was successful. Any time a valid break (larger than 11 $T_{bit}$) is detected, the receiver's state machine should reset to reception of this new frame. This reset condition is only valid during response state, not if an additional synch break occurs during header reception.

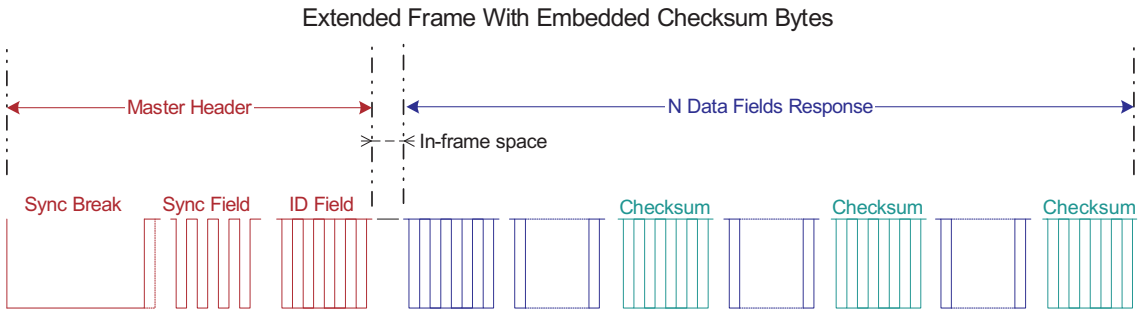
**NOTE:** When an inconsistent synch field (ISFE) error occurs, suggested action for the application is to Reset the SWnRST bit and set the SWnRST bit to make sure that the internal state machines are back to their normal states

### 28.2.6 Extended Frames Handling

The LIN protocol 2.0 and prior includes two extended frames with identifiers 62 (user-defined) and 63 (reserved extended). The response data length of the user-defined frame (ID 62, or 0x3E) is unlimited. The length for this identifier will be set at network configuration time to be shared with the LIN bus nodes.

Extended frame communication is triggered on reception of a header with identifier 0x3E; see Figure 28-10. Once the extended frame communication is triggered, unlike normal frames, this communication needs to be stopped before issuing another header. To stop the extended frame communication the STOP EXT FRAME bit must be set.

**Figure 28-10. Optional Embedded Checksum in Response for Extended Frames**



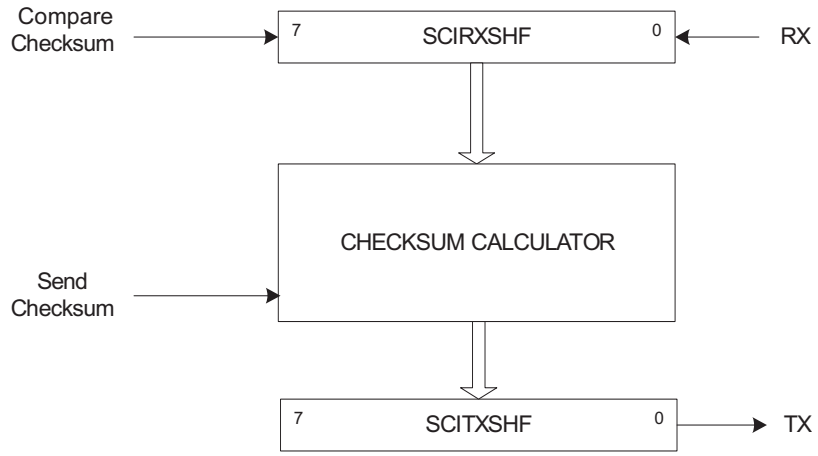

An ID interrupt will be generated (if enabled and there is a match) on reception of ID 62 (0x3E). This interrupt allows the CPU using a software counter to keep track of the bytes that are being sent out and decides when to calculate and insert a checksum byte (recommended at periodic rates). To handle this procedure, SC bit is used. A write to the send checksum bit SC will initiate an automatic send of the checksum byte. The last data field should always be a checksum in compliance with the LIN protocol.

The periodicity of the checksum insertion, defined at network configuration time, is used by the receiving node to evaluate the checksum of the ongoing message, and has the benefit of enhanced reliability.

For the sending node, the checksum is automatically embedded each time the send checksum bit SC is set. For the receiving node, the checksum is compared each time the compare checksum bit CC is set; see Figure 28-11.

**NOTE:** The LIN 2.0 enhanced checksum does not apply to the reserved identifiers. The reserved identifiers always use the classic checksum.

**Figure 28-11. Checksum Compare and Send for Extended Frames**




### *28.2.7 Timeout Control*

Any LIN node listening to the bus and expecting a response initiated from a master node could flag a no-response error timeout event. The LIN protocol defines four types of timeout events, which are all handled by the hardware of the LIN module. The four LIN protocol events are:

- No-response timeout error
- Bus idle detection
- Timeout after wakeup signal
- Timeout after three wakeup signals

#### 28.2.7.1 No-Response Error (NRE)

The no-response error will occur when any node expecting a response waits for $T_{FRAME\_MAX}$ time and the message frame is not fully completed within the maximum length allowed, $T_{FRAME\_MAX}$. After this time a no-response error (NRE) is flagged in the NRE bit of the SCIFLR register. An interrupt is triggered if enabled.

As specified in the LIN 1.3 standard, the minimum time to transmit a frame is:

$T_{FRAME\_MIN} = T_{HEADER\_MIN} + T_{DATA\_FIELD} + T_{CHECKSUM\_FIELD}$

$= 44 + 10N$

where N = number of data fields.

And the maximum time frame is given by:

$T_{FRAME\_MAX} = T_{FRAME\_MIN} * 1.4$

$= (44 + 10N) * 1.4$

The timeout value $T_{FRAME\_MAX}$ is derived from the *N* number of data fields value. The *N* value is either embedded in the header's ID field for messages or is part of the description file. In the latter case, the 3-bit CHAR value in SCIFORMAT register, will indicate the value for *N*.

**NOTE:** The length coding of the ID field does not apply to two extended frame identifiers, ID fields of 0x3E (62) and 0x3F (63). In these cases, the ID field can be followed by an arbitrary number of data byte fields. Also, the LIN 2.0 protocol specification mentions that ID field 0x3F (63) cannot be used. For these two cases, the NRE will not be handled by the LIN controller hardware.

**Table 28-4. Timeout Values in $T_{bit}$ Units**

| N | $T_{DATA\_FIELD}$ | $T_{FRAME\_MIN}$ | $T_{FRAME\_MAX}$ |
|---|---|---|---|
| 1 | 10 | 54 | 76 |
| 2 | 20 | 64 | 90 |
| 3 | 30 | 74 | 104 |
| 4 | 40 | 84 | 118 |
| 5 | 50 | 94 | 132 |
| 6 | 60 | 104 | 146 |
| 7 | 70 | 114 | 160 |
| 8 | 80 | 124 | 174 |

#### 28.2.7.2 Bus Idle Detection

The second type of timeout can occur when a node detects an inactive LIN bus: no transitions between recessive and dominant values are detected on the bus. This happens after a minimum of 4 s (this is 80,000 $F_{LINCLK}$ cycles with the fastest bus rate of 20 kbps). If a node detects no activity in the bus as the TIMEOUT bit is set, then it can be assumed that the LIN bus is in sleep mode. Application software can use the Timeout flag to determine when the LIN bus is inactive and put the LIN into sleep mode by writing the POWERDOWN bit.

**NOTE:** After the timeout was flagged, a SW nRESET should be asserted before entering low power mode. This is required to reset the receiver in case that an incomplete frame was on the bus before the idle period.

#### 28.2.7.3 Timeout after Wakeup Signal and Timeout after Three Wakeup Signals

The third and fourth types of timeout are related to the wakeup signal. A node initiating a wakeup should expect a header from the master within a defined amount of time: timeout after wakeup signal. See Section 28.4.3.3 for more details.

### *28.2.8 TXRX Error Detector (TED)*

The following sources of error are detected by the TXRX error detector logic (TED). The TED logic consists of a bit monitor, an ID parity checker, and a checksum error. The following errors are detected:

- Bit errors (BE)
- Physical bus errors (PBE)
- Identifier parity errors (PE)
- Checksum errors (CE)

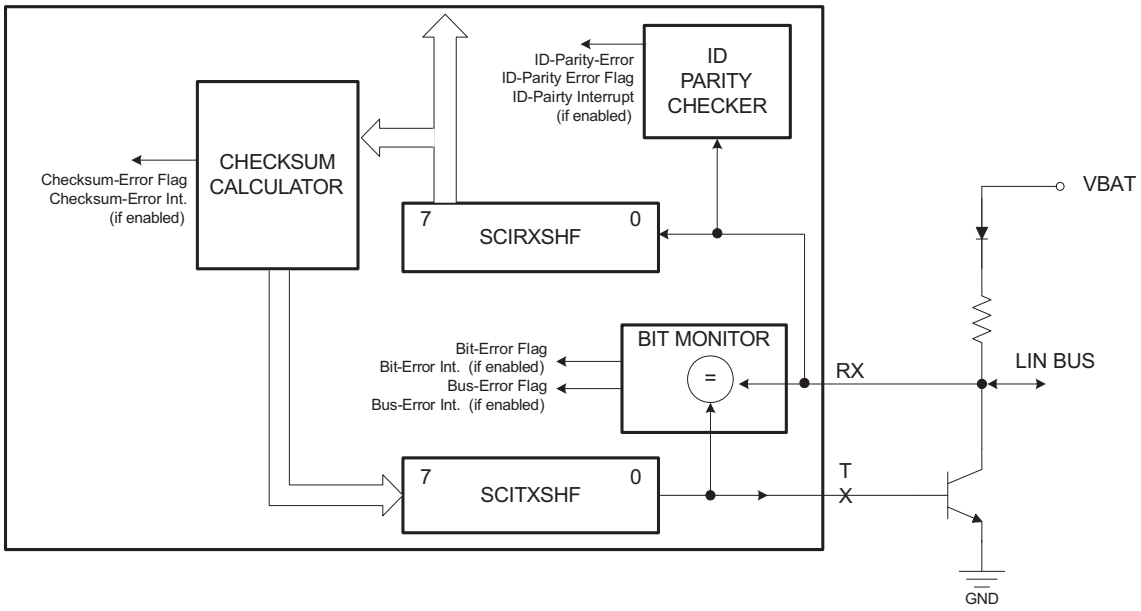
All of these errors (BE, PBE, PE, CE) are flagged. An interrupt for the flagged errors will be generated if enabled. A message is valid for both the transmitter and the receiver if there is no error detected until the end of the frame.

#### 28.2.8.1 Bit Errors

A bit error (BE) is detected at the bit time when the bit value that is monitored is different from the bit value that is sent. A bit error is indicated by the BE flag in SCIFLR. After signaling a BE, the transmission is aborted no later than the next byte. The bit monitor ensures that the transmitted bit in LINTX is the correct value on the LIN bus by reading back on the LINRX pin as shown in Figure 28-12.

**NOTE:** If BE Occurs due to New Header reception during a slave response, the NRE/TIMEOUT flag will not be set for the new frame.

**Figure 28-12. TXRX Error Detector**




#### 28.2.8.2 Physical Bus Errors

A Physical Bus Error (PBE) has to be detected by a master if no valid message can be generated on the bus (Bus shorted to GND or VBAT). The bit monitor detects a PBE during the header transmission, if no Synch Break can be generated (for example, because of a bus shortage to VBAT) or if no Synch break Delimiter can be generated (for example, because of a bus shortage to GND). Once the Synch Break Delimiter was validated, all other deviations between the monitored and the sent bit value are flagged as Bit Errors (BE) for this frame.

#### 28.2.8.3 ID Parity Errors

If parity is enabled, an ID parity error (PE) is detected if any of the two parity bits of the sent ID byte are not equal to the calculated parity on the receiver node. The two parity bits are generated using the following mixed parity algorithm.

$$P0 = ID0 \oplus ID1 \oplus ID2 \oplus ID4 \ (even\ Parity)$$
$$P1 = ID1 \oplus ID3 \oplus ID4 \oplus ID5 \ (odd\ Parity) \quad (17)$$

If an ID-parity error is detected, the ID-parity error is flagged, and the received ID is not valid. See Section 28.2.9 for details.

### 28.2.8.4 Checksum Errors

A checksum error (CE) is detected and flagged at the receiving end if the calculated modulo-256 sum over all received data bytes (including the ID byte if it is the enhanced checksum type) plus the checksum byte does not result in 0xFF. The modulo-256 sum is calculated over each byte by adding with carry, where the carry bit of each addition is added to the LSB of its resulting sum.

For the transmitting node, the checksum byte sent at the end of a message is the inverted sum of all the data bytes (see Figure 28-13) for classic checksum implementation. The checksum byte is the inverted sum of the identifier byte and all the data bytes (see Figure 28-14) for the LIN 2.0 compliant enhanced checksum implementation. The classic checksum implementation should always be used for reserved identifiers 60 to 63; therefore, the CTYPE bit will be overridden in this case. For signal-carrying-frame identifiers (0 to 59) the type of checksum used depends on the CTYPE bit.

**Figure 28-13. Classic Checksum Generation at Transmitting Node**

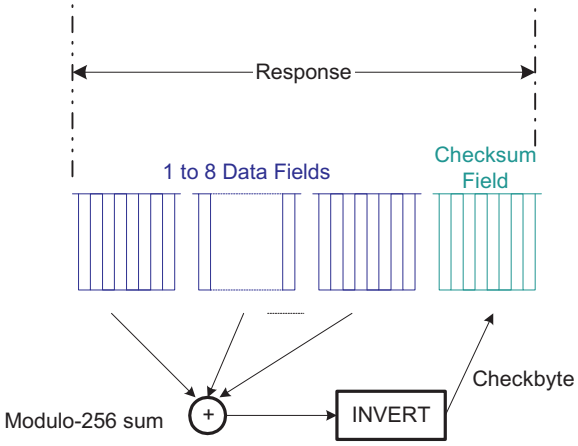
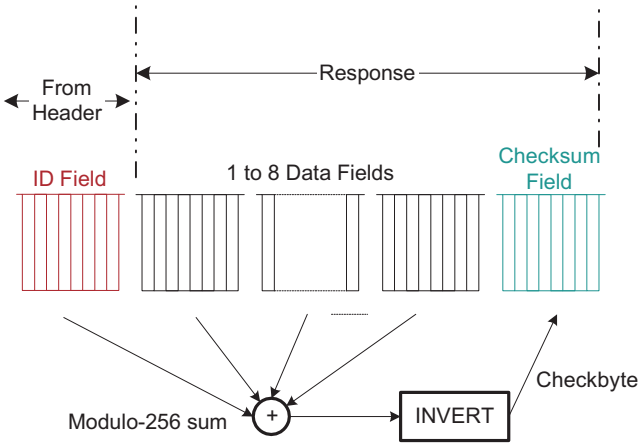



**Figure 28-14. LIN 2.0-Compliant Checksum Generation at Transmitting Node**



### 28.2.9 Message Filtering and Validation

Message filtering uses the entire identifier to determine which nodes will participate in a response, either receiving or transmitting a response. Therefore, two acceptance masks are used as shown in Figure 28-15. During header reception, all nodes filter the ID-Field (ID-Field is the part of the header explained in Figure 28-7) to determine whether they transmit a response or receive a response for the current message. There are two masks for message ID filtering: one to accept a response reception, the other to initiate a response transmission. See Figure 28-15. All nodes compare the received ID to the identifier stored in the ID-SlaveTask BYTE of the LINID register and use the RX ID MASK and the TX ID MASK fields in the LINMASK register to filter the bits of the identifier that should not be compared.

If there is an RX match with no parity error and the RXENA bit is set, there will be an ID RX flag and an interrupt will be triggered if enabled. If there is a TX match with no parity error and the TXENA bit is set, there will be an ID TX flag and an interrupt will be triggered if enabled in the SCISETINT register.

The masked bits become don't cares for the comparison. To build a mask for a set of identifiers, an XOR function could be used.

For example, to build a mask to accept IDs 0x26 and 0x25 using LINID[7:0] = 0x20; that is, compare 5 most significant bits (MSBs) and filter 3 least significant bits (LSBs), the acceptance mask could be:

$$(0x26 + 0x25) \oplus 0x20 = 0x07 \quad (18)$$

A mask of all zeros will compare all bits of the received identifier in the shift register with the ID-BYTE in LINID[7:0]. If HGEN CTRL is set to 1, a mask of 0xFF will always cause a match. A mask of all 1s will filter all bits of the received identifier, and thus there will be an ID match regardless of the content of the ID-SlaveTask BYTE field in the LINID register.

**NOTE:** When the HGEN CTRL bit = 0, the LIN nodes compare the received ID to the ID-BYTE field in the LINID register, and use the RX ID MASK and the TX ID MASK in the LINMASK register to filter the bits of the identifier that should not be compared.

If there is an RX match with no parity error and the RXENA bit is set, there will be an ID RX flag and an interrupt will be triggered if enabled. A mask of all 0s will compare all bits of the received identifier in the shift register with the ID-BYTE field in LINID[7:0]. A mask of all 1s will filter all bits of the received identifier and there will be no match.

During header reception, the received identifier is copied to the Received ID field LINID[23:16]. If there is no parity error and there is either a TX match or an RX match, then the corresponding TX or RX ID flag is set. If the ID interrupt is enabled, then an ID interrupt is generated.

After the ID interrupt is generated, the CPU may read the Received ID field LINID[23:16] and determine what response to load into the transmit buffers.

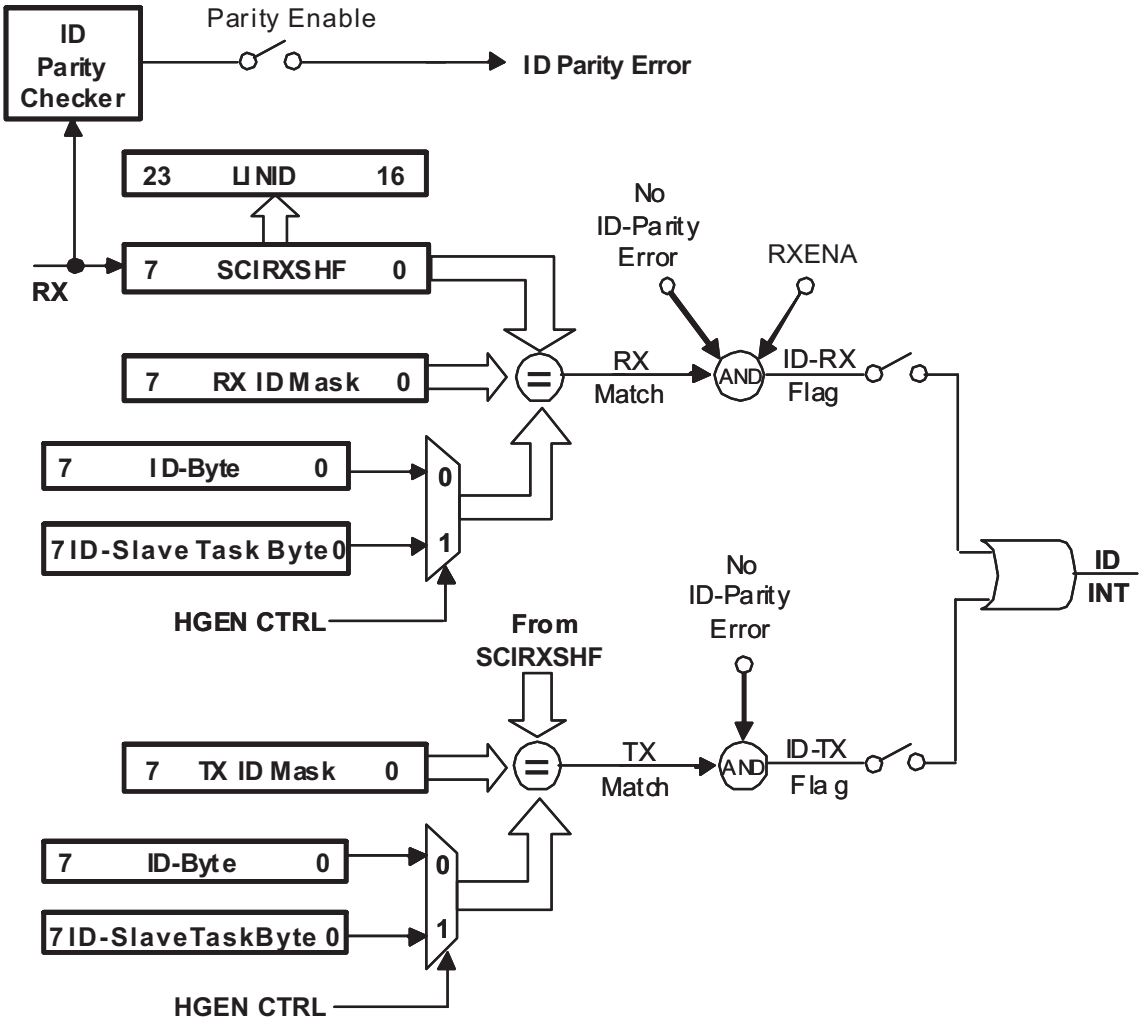
**NOTE:** When byte 0 is written to TD0 (LINTD0[31:24]), the response transmission is automatically generated.

In multi-buffer mode, the TXRDY flag will be set when all the response data bytes and checksum byte are copied to the shift register SCITXSHF. In non-multi-buffer mode, the TXRDY flag is set each time a byte is copied to the SCITXSHF register, and also for the last byte of the frame after the checksum byte is copied to the SCITXSHF register.

In multi-buffer mode, the TXEMPTY flag is set when both the transmit buffer(s) TDy and the SCITXSHF shift register are emptied and the checksum has been sent. In non-multi-buffer mode, TXEMPTY is set each time TD0 and SCITXSHF are emptied, except for the last byte of the frame where the checksum byte must also be transmitted.

If parity is enabled, all slave receiving nodes will validate the identifier using all eight bits of the received ID byte. The SCI/LIN will flag a corrupted identifier if an ID-parity error is detected.

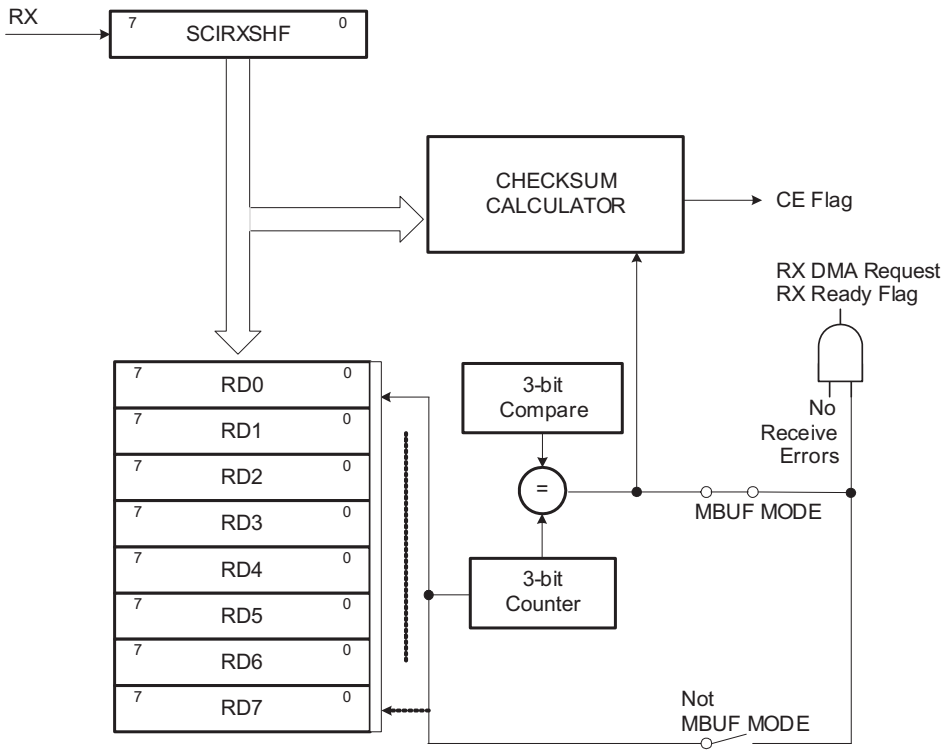
**Figure 28-15. ID Reception, Filtering and Validation**



### 28.2.10 Receive Buffers

To reduce CPU load when receiving a LIN N-byte (with N = 1–8) response in interrupt mode or DMA mode, the SCI/LIN module has eight receive buffers. These buffers can store an entire LIN response in the RDy receive buffers. Figure 28-16 illustrates the receive buffers.

**Figure 28-16. Receive Buffers**




The checksum byte following the data bytes is validated by the internal checksum calculator. The checksum error (CE) flag indicates a checksum error and a CE interrupt will be generated if enabled in the SCISETINT register.

The multi-buffer 3-bit counter counts the data bytes transferred from the SCIRXSHF register to the RDy receive buffers if multi-buffer mode is enabled, or to RD0 if multi-buffer mode is disabled. The 3-bit compare register contains the number of data bytes expected to be received. In cases where the ID BYTE field does not convey message length (see *Note:Optional Control Length Bits* in Section 28.2.5), the LENGTH value, indicates the expected length and is used to load the 3-bit compare register. Whether the length control field or the LENGTH value is used is selectable with the COMM MODE bit.

A receive interrupt, and a receive ready RXRDY flag set as well as a DMA request (RXDMA) could occur after receiving a response if there are no response receive errors for the frame (such as, there is no checksum error, frame error, and overrun error). The checksum byte will be compared before acknowledging a reception. A DMA request can be generated for each received byte or for the entire response depending on whether the multi-buffer mode is enabled or not (MBUF MODE bit).

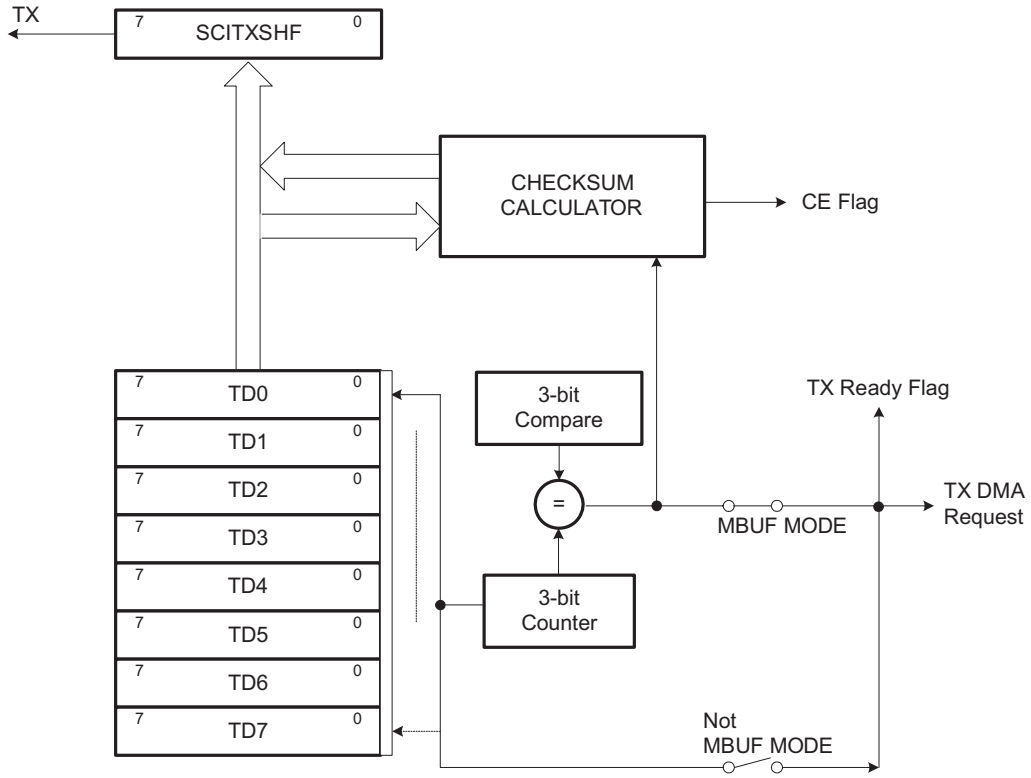
**NOTE:** In multi-buffer mode following are the scenarios associated with clearing the "RXRDY" flag bit:

1. The RXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register.
2. For LENGTH less than or equal to 4, Read to RD0 register will clear the “RXRDY” flag.
3. For LENGTH greater than 4, Read to RD1 register will clear the “RXRDY” flag.

### 28.2.11 Transmit Buffers

To reduce the CPU load when transmitting a LIN N-byte (with N = 1–8) response in interrupt mode or DMA mode, the SCI/LIN module has eight transmit buffers, TD0–TD7 in LINTD0 and LINTD1. With these transmit buffers, an entire LIN response field can be preloaded in the TXy transmit buffers. Optionally, a DMA transfer could be done on a byte-per-byte basis when multi-buffer mode is not enabled (MBUF MODE bit). Figure 28-17 illustrates the transmit buffers.

**Figure 28-17. Transmit Buffers**




The multi-buffer 3-bit counter counts the data bytes transferred from the TDy transmit buffers register if multi-buffer mode is enabled, or from TD0 to SCITXSHF if multi-buffer mode is disabled. The 3-bit compare register contains the number of data bytes expected to be transmitted. If the ID field is not used to convey message length (see *Note:Optional Control Length Bits* in Section 28.2.5), the LENGTH value indicates the expected length and is used instead to load the 3-bit compare register. Whether the length control field or the LENGTH value is used is selectable with the COMM MODE bit.

A transmit interrupt (TX interrupt), and a transmit ready flag (TXRDY flag), and a DMA request (TXDMA) could occur after transmitting a response. A DMA request can be generated for each transmitted byte or for the entire response depending on whether multi-buffer mode is enabled or not (MBUF MODE bit).

The checksum byte will be automatically generated by the checksum calculator and sent after the data-fields transmission is finished. The multi-buffer 3-bit counter counts the data bytes transferred from the TDy buffers into the SCITXSHF register.

**NOTE:** The transmit interrupt request can be eliminated until the next series of data is written into the transmit buffers LINTD0 and LINTD1, by disabling the corresponding interrupt via the SCICLRINT register or by disabling the transmitter via the TXENA bit.

## 28.3 LIN Interrupts

The SCI/LIN module has two interrupt lines, level 0 and level 1, to the peripheral interrupt expansion (PIE) module (see Figure 28-18). Two offset registers SCIINTVECT0 and SCIINTVECT1 determine which flag triggered the interrupt according to the respective priority encoders. Each interrupt condition has a bit to enable/disable the interrupt in the SCISETINT and SCICLRINT registers respectively.

Each interrupt also has a bit that can be set as interrupt level 0(INT0) or as interrupt level 1(INT1). By default, interrupts are in interrupt level 0. SCISETINTLVL sets a given interrupt to level1. SCICLEARINTLVL resets a given interrupt level to the default level 0.

The interrupt vector registers SCIINTVECT0 and SCIINTVECT1 return the vector of the pending interrupt line INT0 or INT1. If more than one interrupt is pending, the interrupt vector register holds the highest priority interrupt.

**Figure 28-18. General Interrupt Scheme**

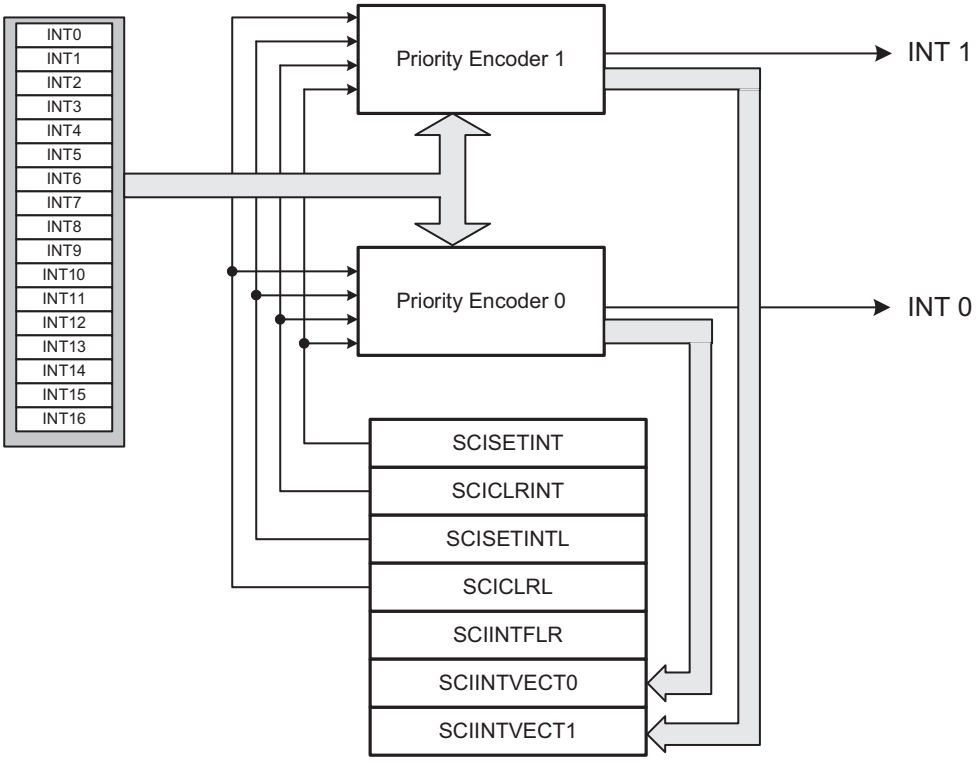
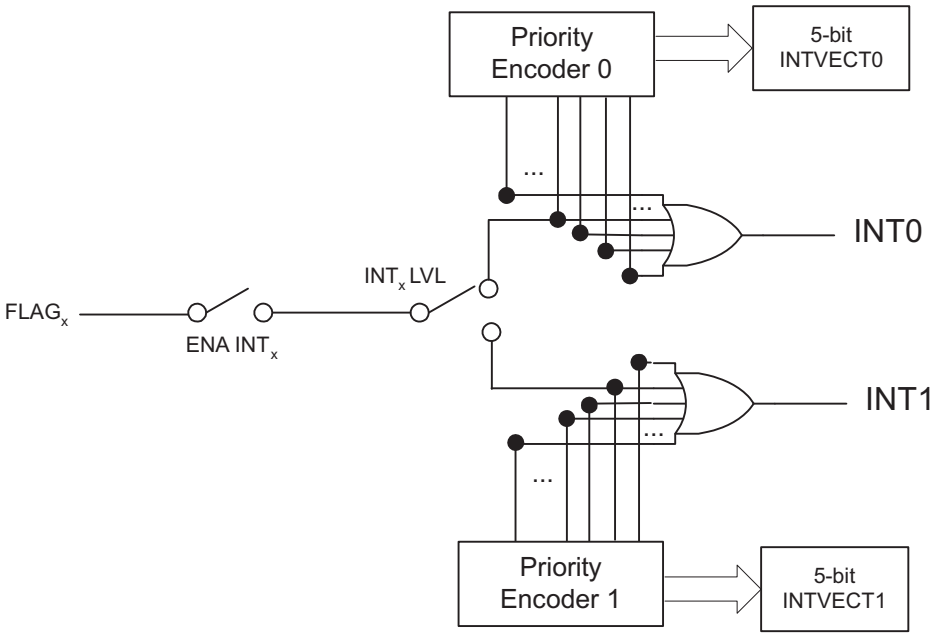


**Figure 28-19. Interrupt Generation for Given Flags**




There are 16 interrupt sources in the SCI/LIN module, In SCI mode 8 interrupts are supported, as seen in Table 28-5.

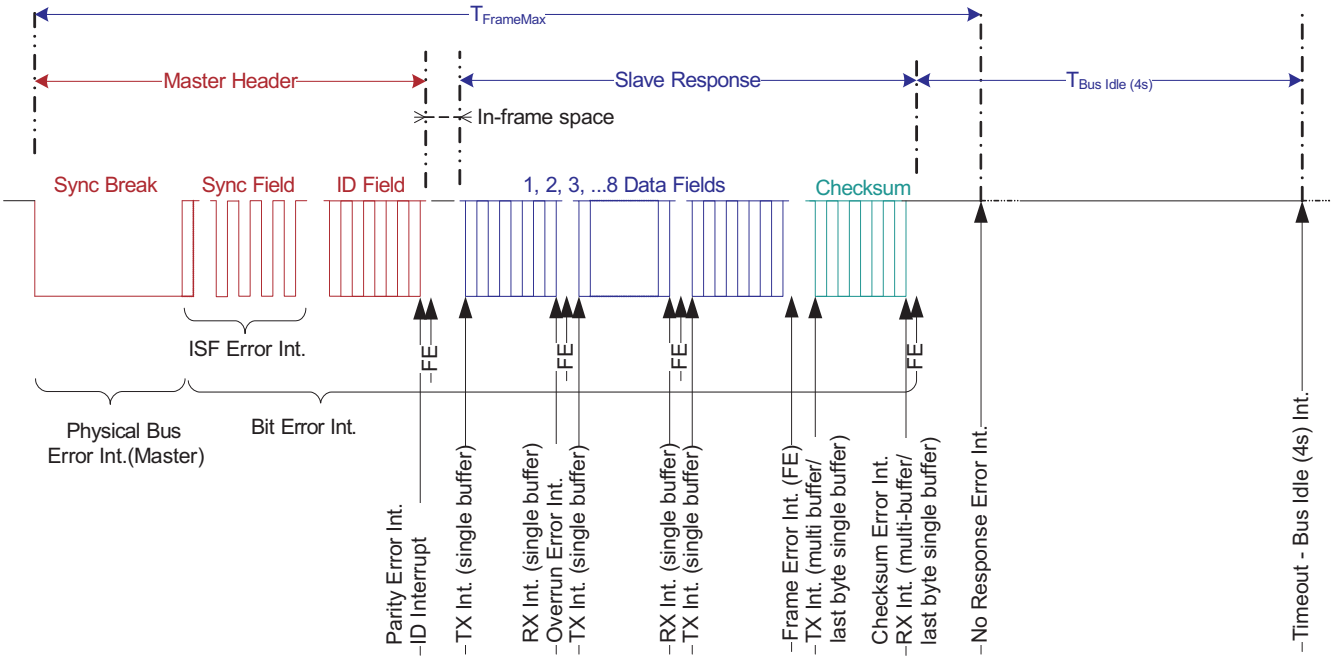
**Table 28-5. SCI/LIN Interrupts**

| Offset [1] | Interrupt | Applicable to SCI | Applicable to LIN |
|---|---|---|---|
| 0 | No interrupt | | |
| 1 | Wakeup | Yes | Yes |
| 2 | Inconsistent-synch-field error | No | Yes |
| 3 | Parity error | Yes | Yes |
| 4 | ID | No | Yes |
| 5 | Physical bus error | No | Yes |
| 6 | Frame error | Yes | Yes |
| 7 | Break detect | Yes | No |
| 8 | Checksum error | No | Yes |
| 9 | Overrun error | Yes | Yes |
| 10 | Bit error | Yes | Yes |
| 11 | Receive | Yes | Yes |
| 12 | Transmit | Yes | Yes |
| 13 | No-response error | No | Yes |
| 14 | Timeout after wakeup signal (150 ms) | No | Yes |
| 15 | Timeout after three wakeup signals (1.5 s) | No | Yes |
| 16 | Timeout (Bus Idle, 4s) | No | Yes |

(1) Offset 1 is the highest priority. Offset 16 is the lowest priority.

A LIN message frame indicating the timing and sequence of the LIN interrupts that could occur is shown in Figure 28-20.

**Figure 28-20. LIN Message Frame Showing LIN Interrupt Timing and Sequence**




### 28.3.1 Servicing Interrupts

When servicing an interrupt, clear the corresponding flag in the flag register (SCIFLR) before clearing the global interrupt flag (LIN_GLB_INT_CLR). The ISR should follow the guidelines below. This will prevent any spurious or duplicate interrupt from occurring.

- Clear the LIN interrupt flag in the SCIFLR register.
- Read the LIN interrupt status register to make sure the flag is cleared.
- Clear the global interrupt flag bit in LIN_GLB_INT_CLR.

**NOTE:** The transmit interrupt is generated before the LIN transmitter is ready to accept new data. Inside of the LIN Transmit Interrupt Handler, the software should wait until the buffer is completely empty before loading the next data. This can be done by polling for the Bus Busy Flag (SCIFLR.BUSY) to be 0.

### 28.3.2 LIN DMA Interface

LIN DMA Interface uses the SCI DMA interface logic. DMA requests for receive (RXDMA request) and transmit (TXDMA request) are available for the SCI/LIN module. There are two modes for DMA transfers depending on whether multi-buffer mode is enabled or not via the multi-buffer enable control bit (MBUF MODE).

**NOTE:** Do not use the DMA to transmit data to multiple slave IDs. Writing to the LINID register will initiate a new transmission. The DMA will write to the LINID register before the LIN state machine is ready to accept the new ID. Doing so will cause the LIN to miss this transmission.

#### 28.3.2.1 LIN Receive DMA Requests

In LIN mode, when the multi-buffer option is enabled, if a received response (up to eight data bytes) is transferred to the receive buffers (RDy), then a DMA request is generated. If the multi-buffer option is disabled, then DMA requests will be generated on a byte-per-byte basis until all the expected response data fields are received. This DMA functionality is enabled and disabled using the SET RX DMA and CLR RX DMA bits, respectively.

#### 28.3.2.2 LIN Transmit DMA Requests

In LIN mode with the multi-buffer option enabled, after a transmission (up to eight data bytes stored in the transmit buffer(s) TDy in the LINTD0 and LINTD1 registers), a DMA request is generated in order to reload the transmit buffer for the next transmission. If the multi-buffer option is disabled, then DMA requests will be generated on a byte-per-byte basis until all bytes are transferred. This DMA functionality is enabled and disabled using the SET TX DMA and CLR TX DMA bits, respectively.

## 28.4 LIN Configurations

The following list details the configuration steps that software should perform prior to the transmission or reception of data in LIN mode. As long as SWnRST is held low the entire time that the LIN is being configured, the order in which the registers are programmed is not important.

- Enable LIN by setting RESET bit.
- Clear SWnRST to 0 before configuring the LIN.
- Configure the LINRX and LINTX pins as SCI functional by setting the RX FUNC and TX FUNC bit.
- Select LIN mode by programming LIN MODE bit.
- Select Master or Slave mode by programming the CLOCK bit.
- Select the desired frame format (checksum, parity, length control) by programming SCIGCR1.
- Select multi-buffer mode by programming MBUF MODE bit.
- Select the baud rate to be used for communication by programming BRSR.
- Set the Maximum baud rate to be used for communication by programming BRSR.
- Set the CONT bit to make LIN not to halt for an emulation breakpoint until its current reception or transmission is complete (this bit is used only in an emulation environment).
- Set LOOP BACK bit to connect the transmitter to the receiver internally (this feature is used to perform a self-test).
- Select the receiver enable RXENA bit if data is to be received.
- Select the transmit enable TXENA bit if data is to be transmitted.
- Select the RX ID MASK and the TX ID MASK fields in the LINMASK register.
- Set SWnRST to 1 after the SCI is configured.
- Perform Receive or Transmit data (see Section 28.2.9, Section 28.2.10, and Section 28.4.2).

**NOTE:** If TXENA is set and the SWnRST is released, the LIN will only generate a new DMA request. The LIN hardware will not generate a new Transmit interrupt request in this situation. If using interrupts, the first transmission must be started by software by writing the data to transmit followed by writing to LIN TX to initiate the transmission.

### *28.4.1 Receiving Data*

The LIN receiver is enabled to receive messages if the RX FUNC bit and the RXENA bit are set to 1. If the RX FUNC bit is not set, the LINRX pin functions as a general purpose I/O pin rather than as an SCI/LIN function pin.

ID RX flag is set after a valid LIN ID is received with RX Match, generated ID interrupt if enabled.

#### 28.4.1.1 Receiving Data in Single-Buffer Mode

Single Buffer Mode is selected when MBUF MODE bit is 0. In this mode SCI/LIN sets the RXRDY bit when it transfers newly received data from SCIRXSHF to RD0. The SCI clears the RXRDY bit after the new data in RD0 has been read. Also, as data is transferred from SCIRXSHF to RD0, the LIN sets FE, OE, or PE if any of these error conditions were detected in the received data. These error conditions are supported with configurable Interrupt capability.

You can read the Received data by:

1. Polling Receive Ready Flag
2. Receive Interrupt
3. DMA

In polling method, software can poll for RXRDY bit and read the data from RD0 byte of LINRD0 register once RXRDY is set high. CPU is unnecessarily overloaded by selecting Polling mode. To avoid this, you can use either Interrupt or DMA method. To use interrupt method SET RX INT bit should be set and to use DMA SET RX DMA bit should be set. Either an Interrupt or a DMA request is generated the moment RXRDY is set. If checksum scheme is used once Compare Checksum CC bit is set, the checksum will be compared on the byte that is currently being received, expected to be the checkbyte be enabled during the last byte of the data. CC bit will be cleared once Checksum is received. A CE will immediately be flagged if there is a checksum error.

#### 28.4.1.2 Receiving Data in Multi-Buffer Mode

Multi-Buffer Mode is selected when MBUF MODE bit is 1. In this mode SCI/LIN sets the RXRDY bit after receiving the programmed number of data in the receive buffer and the checksum field, the complete frame. The error condition detection logic is same as Single Buffer mode, except that it monitors for the complete frame. Like Single Buffer mode, you can use either Interrupt, DMA or polling method to read the data. The received data has to be read from the LINRD0 and LINRD1 register, based on the number of bytes. For LENGTH less than or equal to 4, Read to LINRD0 register will clear the “RXRDY” flag. For LENGTH greater than 4, Read to LINRD1 register will clear the “RXRDY” flag. If checksum scheme is enabled by setting CC bit during the reception of the data, then the byte that is received after the reception of the programmed no. of data bytes indicated by LENGTH is treated as a checksum byte and compared.

### *28.4.2 Transmitting Data*

The SCI transmitter is enabled if the TX FUNC bit and the TXENA bit are set to 1. If the TX FUNC bit is not set, the LINTX pin functions as a general purpose I/O pin rather than as an SCI function pin. Any value written to the TD0 before TXENA is set to 1 is not transmitted. Both of these control bits allow for the SCI transmitter to be held inactive independently of the receiver.

ID TX flag is set after a valid LIN ID is received with TX Match, generated ID interrupt if enabled.

#### 28.4.2.1 Transmitting Data in Single-Buffer Mode

Single Buffer Mode is selected when MBUF MODE bit is 0. In this mode LIN waits for data to be written to TD0, transfers it to SCITXSHF, and transmits it. The flags TXRDY and TX EMPTY indicate the status of the transmit buffers. That is, when the transmitter is ready for data to be written to TD0, the TXRDY bit is set. Additionally, if both TD0 and SCITXSHF are empty, then the TX EMPTY bit is also set.

You can transmit data by:

1. Polling Transmit Ready Flag
2. Receive Interrupt
3. DMA

In polling method, software can poll for TXRDY bit to go high before writing the data to TD0 register. CPU is unnecessarily overloaded by doing this Polling method. To avoid this, you can use either Interrupt or DMA method. To use interrupt method SET TX INT bit should be set and to use DMA SET TX DMA bit should be set. Either an Interrupt or a DMA request is generated the moment TXRDY is set. When the SCI has completed transmission of all pending frames, the SCITXSHF register and TD0 are empty, the TXRDY bit is set, and an interrupt/DMA request is generated, if enabled. Because all data has been transmitted, the interrupt/DMA request should be halted. This can be done by either disabling the transmit interrupt (CLR TX INT) / DMA request (CLR TX DMA bit) or by disabling the transmitter (clear TXENA bit). In checksum scheme once Send Checksum SC bit is set, the checksum will be sent after the current transmission.

**NOTE:** The TXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0 or SCIINTVECT1 register.

#### 28.4.2.2 Transmitting Data in Multi-Buffer Mode

Multi-buffer mode is selected when MBUF MODE bit is 1. Similar to single-buffer mode the software can use polling, Interrupt or DMA method to write the data to be transmitted. The data to be transmitted has to be written to LINTD0 and LINTD1 register, based on the number of bytes. SCI/LIN waits for data to be written to Byte 0(TD0) of LINTD0 register and transfers the programmed number of bytes to SCITXSHF to transmit one by one automatically. In checksum scheme once Send Checksum SC bit is set, the checksum will be sent after transmission of programmed no. of data bytes indicated by LENGTH field.

### *28.4.3 LIN Low-Power Mode*

The SCI/LIN module can be put in either local or global low-power mode. Global low-power mode (HALT) is asserted by the system and is not controlled by the SCI/LIN module. During global low-power mode, all clocks to the SCI/LIN are turned off so the module is completely inactive.

The LIN module may enter low power mode either when there was no activity on the LINRX pin for more than 4s (this can be either a constant recessive or dominant level) or when a Sleep Command frame was received. Once the Timeout flag (SCIFLR.4) was set or once a Sleep Command was received, the POWERDOWN bit (SCIGCR2.0) must be set by the application software to make the module enter local low-power mode. A wakeup signal will terminate the sleep mode of the LIN bus.

**NOTE: Enabling Local Low-Power Mode During Receive and Transmit**

If the wakeup interrupt is enabled and low-power mode is requested while the receiver is receiving data, then the SCI/LIN immediately generates a wake-up interrupt to clear the powerdown bit. Thus, the SCI/LIN is prevented from entering low-power mode and completes the current reception. Otherwise, if the wakeup interrupt is disabled, the SCI/LIN completes the current reception and then enters the low-power mode.

#### 28.4.3.1 Entering Sleep Mode

In LIN protocol, a sleep command is used to broadcast the sleep mode to all nodes. The sleep command consists of a diagnostic master request frame with identifier 0x3C (60), with the first data field as 0x00. There should be no activity in the bus once all nodes receive the sleep command: the bus is in sleep mode.

Local low-power mode is asserted by setting the POWERDOWN bit; setting this bit stops the clocks to the SCI/LIN internal logic and registers. Clearing the POWERDOWN bit causes SCI/LIN to exit from local low-power mode. All the registers are accessible during local power-down mode. If a register is accessed in low-power mode, this access results in enabling the clock to the module for that particular access alone.

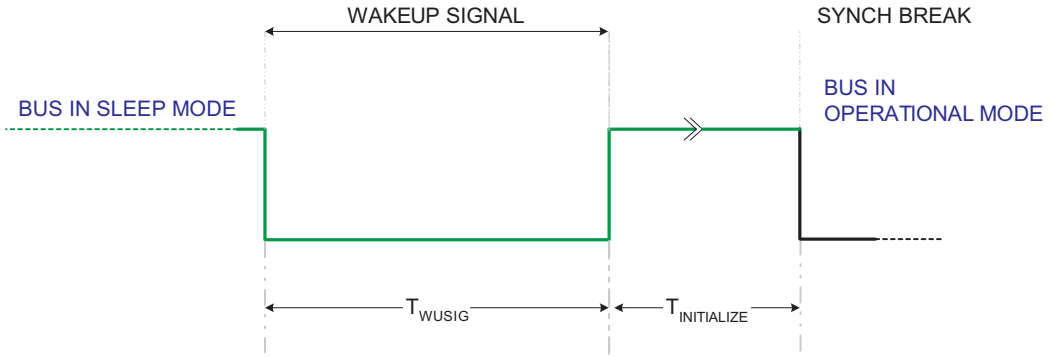
### 28.4.3.2 Wakeup

The wakeup interrupt is used to allow the SCI/LIN module to automatically exit low-power mode. A SCI/LIN wakeup is triggered when a low level is detected on the receive RX pin, and this clears the POWERDOWN bit.

**NOTE:** If the wakeup interrupt is disabled then the SCI/LIN enters low-power mode whenever it is requested to do so, but a low level on the receive RX pin does NOT cause the SCI/LIN to exit low-power mode.

In LIN mode, any node can terminate sleep mode by sending a wakeup signal; see Figure 28-21. A slave node that detects the bus in sleep mode, and with a wakeup request pending, will send a wakeup signal. The wakeup signal is a dominant value on the LIN bus for $T_{WUSIG}$; this is at least 5 $T_{bits}$ for the LIN bus baud rates. The wakeup signal is generated by sending an 0xF0 byte containing 5 dominant $T_{bits}$ and 5 recessive $T_{bits}$.

**Figure 28-21. Wakeup Signal Generation**




$$0.25\text{ms} \le T_{WUSIG} \le 5\text{ms} \quad (19)$$

Assuming a perfect bus with no noise or loading effects, a write of 0xF0 to TD0 will load the transmitter to meet the wakeup signal timing requirement for $T_{WUSIG}$. Then, setting the GENWU bit will transmit the preloaded value in TD0 for a wakeup signal transmission.

**NOTE:** The GENWU bit can be set/reset only when SWnRST is set to'1' and the node is in power down mode. The bit will be cleared on a valid synch break detection. A master sending a wakeup request, will exit power down mode upon reception of the wakeup pulse. The bit will be cleared on a SWnRST. This can be used to stop a master from sending further wakeup requests.

The TI TPIC1021 LIN transceiver, upon receiving a wakeup signal, will translate it to the microcontroller for wakeup with a dominant level on the RX pin, or a signal to the voltage regulator. While the POWERDOWN bit is set, if the LIN module detects a recessive-to-dominant edge (falling edge) on the RX pin, it will generate a wakeup interrupt if enabled in the SCISETINT register.

According to LIN protocol 2.0, the TI TPIC1021 LIN transceiver detecting a dominant level on the bus longer than 150 ms will detect it as a wakeup request. The LIN controller's slave is ready to listen to the bus in less than 100 ms ($T_{INITIALIZE}$<100ms) after a dominant-to-recessive edge (end-of-wakeup signal).

#### 28.4.3.3 Wakeup Timeouts

The LIN protocol defines the following timeouts for a wakeup sequence. After a wakeup signal has been sent to the bus, all nodes wait for the master to send a header. If no synch field is detected before 150 ms (3,000 cycles at 20 kHz) after wakeup signal is transmitted, a new wakeup is sent by the same node that requested the first wakeup. This sequence is not repeated more than two times. After three attempts to wake up the LIN bus, wakeup signal generation is suspended for a 1.5 s (30,000 cycles at 20 kHz) period after three breaks.

**NOTE:** To achieve compatibility to LIN1.3 timeout conditions, the MBRS register must be set to assure that the LIN 2.0 (real-time-based) timings meet the LIN 1.3 bit time base. A node triggering the wakeup should set the MBRS register accordingly to meet the targeted time as 128 Tbits x programmed prescaler.

The LIN controller handles the wakeup expiration times defined by the LIN protocol with a hardware implementation.

#### 28.4.3.4 Emulation Mode

In emulation mode, the CONT bit determines how the SCI/LIN operates when the program is suspended. The SCI/LIN counters are affected by this bit during debug mode. when set, the counters are not stopped and when cleared, the counters are stopped debug mode.

Any reads in emulation mode to a SCI/LIN register will not have any effect on the flags in the SCIFLR register.

**NOTE:** When emulation mode is entered during the Frame transmission or reception of the frame and CONT bit is not set, Communication is not expected to be successful. The suggested usage is to set CONT bit during emulation mode for successful communication.

## 28.5 Registers

### *28.5.1 Local Interconnect Network Base Addresses*

**Table 28-6. I2C Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| LinaRegs | LIN_REGS | 0000 6A00 | 0000 6AFF |

### 28.5.1.1 LIN_REGS Registers

Table 28-7 lists the memory-mapped registers for the LIN_REGS. All register offset addresses not listed in Table 28-7 should be considered as reserved locations and the register contents should not be modified.

**Table 28-7. LIN_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | SCIGCR0 | Global Control Register 0 | | Go |
| 2h | SCIGCR1 | Global Control Register 1 | | Go |
| 4h | SCIGCR2 | Global Control Register 2 | | Go |
| 6h | SCISETINT | Interrupt Enable Register | | Go |
| 8h | SCICLEARINT | Interrupt Disable Register | | Go |
| Ah | SCISETINTLVL | Set Interrupt Level Register | | Go |
| Ch | SCICLEARINTLVL | Clear Interrupt Level Register | | Go |
| Eh | SCIFLR | Flag Register | | Go |
| 10h | SCIINTVECT0 | Interrupt Vector Offset Register 0 | | Go |
| 12h | SCIINTVECT1 | Interrupt Vector Offset Register 1 | | Go |
| 14h | SCIFORMAT | Length Control Register | | Go |
| 16h | BRSR | Baud Rate Selection Register | | Go |
| 18h | SCIED | Emulation buffer Register | | Go |
| 1Ah | SCIRD | Receiver data buffer Register | | Go |
| 1Ch | SCITD | Transmit data buffer Register | | Go |
| 1Eh | SCIPIO0 | Pin control Register 0 | | Go |
| 22h | SCIPIO2 | Pin control Register 2 | | Go |
| 30h | LINCOMP | Compare register | | Go |
| 32h | LINRD0 | Receive data register 0 | | Go |
| 34h | LINRD1 | Receive data register 1 | | Go |
| 36h | LINMASK | Acceptance mask register | | Go |
| 38h | LINID | LIN ID Register | | Go |
| 3Ah | LINTD0 | Transmit Data Register 0 | | Go |
| 3Ch | LINTD1 | Transmit Data Register 1 | | Go |
| 3Eh | MBRSR | Baud Rate Selection Register | | Go |
| 48h | IODFTCTRL | IODFT for LIN | | Go |
| 70h | LIN_GLB_INT_EN | LIN Global Interrupt Enable Register | | Go |
| 72h | LIN_GLB_INT_FLG | LIN Global Interrupt Flag Register | | Go |
| 74h | LIN_GLB_INT_CLR | LIN Global Interrupt Clear Register | | Go |

Complex bit access types are encoded to fit into small table cells. Table 28-8 shows the codes that are used for access types in this section.

**Table 28-8. LIN_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** | | |

**Table 28-8. LIN_REGS Access Type Codes (continued)**

| Access Type | Code | Description |
|---|---|---|
| i,j,k,l,m,n | | When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula. |
| y | | When this variable is used in a register name, an offset, or an address it refers to the value of a register array. |

#### 28.5.1.1.1 SCIGCR0 Register (Offset = 0h) [reset = 0h]

SCIGCR0 is shown in Figure 28-22 and described in Table 28-9.

Return to Summary Table.

Global Control Register 0

**Figure 28-22. SCIGCR0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | RESET |
| R-0h | | | | | | | R/W-0h |

**Table 28-9. SCIGCR0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-1 | RESERVED | R | 0h | Reserved |
| 0 | RESET | R/W | 0h | This bit resets the SCI/BLIN module.<br>This bit is effective in SCI and LIN mode<br>Reset type: SYSRSn<br>0h (R/W) = EALLOW-protected Write mode:<br>Reset module<br>EALLOW-protected and normal mode Read:<br>Module is under reset.<br>1h (R/W) = EALLOW-protected Write mode:<br>Pull module out of reset<br>EALLOW-protected and normal mode Read:<br>Module is out of reset. |

#### 28.5.1.1.2 SCIGCR1 Register (Offset = 2h) [reset = 0h]

SCIGCR1 is shown in Figure 28-23 and described in Table 28-10.

Return to Summary Table.

Global Control Register 1

**Figure 28-23. SCIGCR1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | TXENA | RXENA |
| R-0h | | | | | | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | CONT | LOOPBACK |
| R-0h | | | | | | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | STOPEXTFRAME | HGENCTRL | CTYPE | MBUFMODE | ADAPT | SLEEP |
| R-0h | | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SWnRST | LINMODE | CLK_MASTER | STOP | PARITY | PARITYENA | TIMINGMODE | COMMMODE |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 28-10. SCIGCR1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-26 | RESERVED | R | 0h | Reserved |
| 25 | TXENA | R/W | 0h | Transmit enable.<br>This bit is effective in LIN and SCI modes. Data is transferred from SCITD or the TDy (with y=0, 1,...7) buffers in LIN mode to the SCITXSHF shift out register only when the TXENA bit is set.<br>Reset type: SYSRSn<br>0h (R/W) = Disable transfers from SCITD or TDy to SCITXSHF<br>1h (R/W) = Enable transfers of data from SCITD or TDy to SCITXSHF |
| 24 | RXENA | R/W | 0h | Receive enable.<br>This bit is effective in LIN and SCI modes. RXENA allows or prevents the transfer of data from SCIRXSHF to SCIRD or the receive multibuffers.<br>Note: Clearing RXENA stops received characters from being transferred into the receive buffer or multi-buffers, prevents the RX status flags (see Table 7) from being updated by receive data, and inhibits both receive and error interrupts. However, the shift register continues to assemble data regardless of the state of RXENA.<br>Note: If RXENA is cleared before the time the reception of a frame is complete, the data from the frame is not transferred into the receive buffer.<br>Note: If RXENA is set before the time the reception of a frame is complete, the data from the frame is transferred into the receive buffer. If RXENA is set while SCIRXSHF is in the process of assembling a frame, the status flags are not guaranteed to be accurate for that frame. To ensure that the status flags correctly reflect what was detected on the bus during a particular frame, RXENA should be set before the detection of that frame<br>Reset type: SYSRSn<br>0h (R/W) = Prevents the receiver from transferring data from the shift buffer to the receive buffer or multi-buffers<br>1h (R/W) = Allows the receiver to transfer data from the shift buffer to the receive buffer or multi-buffers |

**Table 28-10. SCIGCR1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 23-18 | RESERVED | R | 0h | Reserved |
| 17 | CONT | R/W | 0h | Continue on suspend.<br>This bit has an effect only when a program is being debugged with an emulator, and it determines how the SCI/BLIN operates when the program is suspended.<br>This bit affects the BLIN counters. When this bit is set, the counters are not stopped during debug. When this bit is cleared, the counters are stopped during debug.<br>Reset type: SYSRSn<br>0h (R/W) = When debug mode is entered, the SCI/BLIN state machine is frozen. Transmissions and LIN counters are halted and resume when debug mode is exited.<br>1h (R/W) = When debug mode is entered, the SCI/BLIN continues to operate until the current transmit and receive functions are complete |
| 16 | LOOPBACK | R/W | 0h | Loopback bit.<br>This bit is effective in LIN and SCI modes. The selfchecking option for the SCI/BLIN can be selected with this bit. If the LINTX and LINRX pins are configured with SCI/BLIN functionality, then the LINTX pin is internally connected to the LINRX pin. Externally, during loop back operation, the<br>LINTX pin outputs a high value and the LINRX pin is in a high-impedance state. If this bit value is changed while the SCI/BLIN is transmitting or receiving data, errors may result.<br>Reset type: SYSRSn<br>0h (R/W) = Loop back mode is disabled.<br>1h (R/W) = Loop back mode is enabled. |
| 15-14 | RESERVED | R | 0h | Reserved |
| 13 | STOPEXTFRAME | R/W | 0h | Stop extended frame communication.<br>This bit is effective in LIN mode only. This bit can be written only during extended frame communication. When the extended frame communication is stopped, this bit is cleared automatically.<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Extended frame communication will be stopped, once current frame transmission/reception is completed. |
| 12 | HGENCTRL | R/W | 0h | LIN mode bit.<br>This bit controls the type of Mask filtering comparison.<br>Reset type: SYSRSn<br>0h (R/W) = ID filtering using ID-Byte.<br>RECID and ID-BYTE fields in LINID register are used for detecting a match (using TX/RX MASK values). Mask of 0xFF in LINMASK register will result in NO match.<br>1h (R/W) = ID filtering using ID-SlaveTask byte (Recommended).<br>RECID and ID-Slave task byte are used for detecting a match (using TX/RX Mask values).Mask of 0xFF in LINMASK register will result in ALWAYS match |

**Table 28-10. SCIGCR1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 11 | CTYPE | R/W | 0h | Checksum type.<br>Reset type: SYSRSn<br>0h (R/W) = Classic checksum is used.<br>This checksum is compatible with LIN 1.3 slave nodes. The Classic Checksum contains the modulo-256 sum with carry over all data bytes.<br>Frames sent with Identifier 60 (0x3C) to 63 (0x3F) must allways use the classic checksum.<br>1h (R/W) = Enhanced checksum is used.<br>The enhanced checksum is compatible with LIN 2.0 and newer slave nodes. The Enhanced Checksum contains the modulo-256 sum with carry over all data bytes AND the protected Identifier. |
| 10 | MBUFMODE | R/W | 0h | Multibuffer mode.<br>This bit is effective in LIN and SCI modes.This bit controls receive/transmit buffer usage, i.e., whether the RX/TX multibuffers are used or a single register, RD0/TD0, is used.<br>Reset type: SYSRSn<br>0h (R/W) = The multi-buffer mode is disabled.<br>1h (R/W) = The multi-buffer mode is enabled. |
| 9 | ADAPT | R/W | 0h | Adapt mode enable.<br>This mode is effective in LIN mode only. This bit has an effect during the detection of the Synch Field. There are two LIN protocol bit rate modes that could be enabled with this bit according to the Node capability file definition: automatic or select. Software and network configuration will decide which of the previous two modes. When this bit is cleared, the LIN 2.0 protocol fixed bit rate should be used. If the ADAPT bit is set, a BLIN slave node detecting the baudrate will compare it to the prescalers in BRSR register and update it if they are different. The BRSR register will be updated with the new value. If this bit is not set there will be no adjustment to the BRSR register.<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = EALLOW-protected and normal mode Read:<br>Automatic baudrate adjustment is disabled<br>EALLOW-protected and normal mode Write:<br>Disable Automatic baudrate adjustment<br>1h (R/W) = EALLOW-protected and normal mode Read:<br>Automatic baudrate adjustment is enabled<br>EALLOW-protected and normal mode Write:<br>Enable Automatic baudrate adjustment |
| 8 | SLEEP | R/W | 0h | SCI sleep<br>SCI compatibility mode only. In a multiprocessor configuration, this bit controls the receive sleep function. Clearing this bit brings the SCI out of sleep mode.<br>The receiver still operates when the SLEEP bit is set<br>however, RXRDY is updated and SCIRD is loaded with new data only when an address frame is detected.<br>This field is writable in SCI mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Sleep mode is disabled.<br>1h (R/W) = Sleep mode is enabled. |
| 7 | SWnRST | R/W | 0h | Software reset (active low).<br>This bit is effective in LIN and SCI modes.<br>Reset type: SYSRSn<br>0h (R/W) = The SCI/BLIN is in its reset state<br>1h (R/W) = The SCI/BLIN is in its ready state |

**Table 28-10. SCIGCR1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | LINMODE | R/W | 0h | LIN mode<br>This bit controls the module mode of operation.<br>Reset type: SYSRSn<br>0h (R/W) = LIN mode is disabled<br>SCI compatibility mode is enabled.<br>1h (R/W) = LIN mode is enabled<br>SCI compatibility mode is disabled. |
| 5 | CLK_MASTER | R/W | 0h | SCI internal clock enable or LIN Master/Slave configuration.<br>In the SCI mode, this bit enables the clock to the SCI module. In LIN mode, this bit determines whether a LIN node is a slave or master<br>Reset type: SYSRSn<br>0h (R/W) = SCI-compatible mode<br>Reserved.<br>LIN mode<br>The node is in slave mode.<br>1h (R/W) = SCI-compatible mode<br>Enable clock to the SCI module.<br>LIN mode<br>The node is in master mode. |
| 4 | STOP | R/W | 0h | SCI number of stop bits.<br>This bit is effective in SCI-compatible mode only.<br>Note: The receiver checks for only one stop bit. However in idle-line mode, the receiver waits until the end of the second stop bit (if STOP = 1) to begin checking for an idle period.<br>This field is writable in SCI mode only.<br>Reset type: SYSRSn<br>0h (R/W) = One stop bit is used<br>1h (R/W) = Two stop bits are used. |
| 3 | PARITY | R/W | 0h | SCI parity odd/even selection.<br>SCI-Compatible mode only. If the PARITY ENA bit (SCIGCR1.2) is set, PARITY designates odd or even parity.<br>Note: The parity bit is calculated based on the data bits in each frame and the address bit (in address-bit mode). The start and stop fields in the frame are not included in the parity calculation.<br>Note: For odd parity, the SCI transmits and expects to receive a value in the parity bit that makes odd the total number of bits in the frame with the value of 1.<br>For even parity, the SCI transmits and expects to receive a value in the parity bit that makes even the total number of bits in the frame with the value of 1.<br>This field is writable in SCI mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Odd parity is used.<br>1h (R/W) = Even parity is used. |

**Table 28-10. SCIGCR1 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | PARITYENA | R/W | 0h | Parity enable.<br>Enables or disables the parity function. Compatible or buffered SCI mode:<br>Reset type: SYSRSn<br>0h (R/W) = SCI compatibility or buffered SCI mode<br>Parity disabled<br>no parity bit is generated during transmission or is expected during reception<br>LIN mode<br>ID-parity verification is disabled.<br>1h (R/W) = SCI compatibility or buffered SCI mode<br>Parity enabled. A parity bit is generated during transmission and is expected during reception<br>LIN mode<br>ID-parity verification is enabled. |
| 1 | TIMINGMODE | R/W | 0h | SCI timing mode bit.<br>This bit is effective in SCI-compatible mode only. It must be set to 1 when the SCI mode is used. It configures the SCI for asynchronous operation.<br>Reset type: SYSRSn<br>0h (R/W) = Reserved.<br>1h (R/W) = Must be set to 1 when module is configured for SCI operation |
| 0 | COMMMODE | R/W | 0h | SCI/BLIN communication mode bit.<br>In compatibility mode, it selects the SCI communication mode. In LIN mode it selects length control option for ID-field bits ID4 and ID5.<br>Reset type: SYSRSn<br>0h (R/W) = SCI-compatible mode:<br>Idle-line mode is used.<br>LIN mode<br>ID4 and ID5 are not used for length control.<br>1h (R/W) = SCI-compatible mode:<br>Address-bit mode is used.<br>LIN mode<br>ID4 and ID5 are used for length control. |

#### 28.5.1.1.3 SCIGCR2 Register (Offset = 4h) [reset = 0h]

SCIGCR2 is shown in Figure 28-24 and described in Table 28-11.

Return to Summary Table.

Global Control Register 2

**Figure 28-24. SCIGCR2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | CC | SC |
| R-0h | | | | | | R/W-0h | R/W-0h |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | GENWU |
| R-0h | | | | | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | POWERDOWN |
| R-0h | | | | | | | R/W-0h |

**Table 28-11. SCIGCR2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-18 | RESERVED | R | 0h | Reserved |
| 17 | CC | R/W | 0h | Compare Checksum.<br>LIN mode only. This bit is used by the receiver for extended frames to trigger a checksum compare. The user will initiate this transaction by writing a one to this bit.<br>In non multibuffer mode, once the CC bit is set, the checksum will be compared on the byte that is currently being received, expected to be the checkbyte.<br>During MultiBuffer mode, following are the scenarios associated with the "CC" bit :<br>- If "Compare Checksum" bit is set during the reception of the data, then the byte that is received after the reception of the programmed no. of data bytes indicated by SCIFORMAT[18:16], is treated as a checksum byte.<br>- If "Compare Checksum" bit is set during the IDLE period (i.e. during inter-frame space), then the next immediate byte will be treated as a checksum byte.<br>A CE will immediatly be flagged if there is a checksum error. This bit is automatically cleared once the checksum is successfully compared.<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Compare checksum on expected checkbyte |
| 16 | SC | R/W | 0h | Send Checksum<br>LIN mode only. This bit is used by the transmitter with extended frames to send a checkbyte. In non multibuffer mode the checkbyte will be sent after the current byte transmission. In multibuffer mode the checkbyte will be sent after the last byte count, indicated by the SCIFORMAT[18:16]).<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = No checkbyte will be sent.<br>1h (R/W) = A checkbyte will be sent |

**Table 28-11. SCIGCR2 Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-9 | RESERVED | R | 0h | Reserved |
| 8 | GENWU | R/W | 0h | Generate wakeup signal.<br>This bit controls the generation of a wakeup signal, by transmitting the TDO buffer value. This bit is cleared on reception of a valid synch break.<br>Reset type: SYSRSn<br>0h (R/W) = No effect<br>1h (R/W) = Transmit TDO for wakeup<br>The bit will be cleared on a SWnRST (SCIGCR1.7) |
| 7-1 | RESERVED | R | 0h | Reserved |
| 0 | POWERDOWN | R/W | 0h | Power down.<br>This bit is effective in LIN or SCI-compatible mode. When the powerdown bit is set, the SCI/BLIN module attempts to enter local low-power mode. If the POWERDOWN bit is set while the receiver is actively receiving data and the wakeup interrupt is disabled, then the SCI/BLIN will delay low-power<br>mode from being entered until completion of reception. In LIN mode the user may set the POWERDOWN bit on Sleep Command reception or on idle bus detection (more than 4 seconds, i.e. 80,000 cycles at 20kHz)<br>Reset type: SYSRSn<br>0h (R/W) = Normal operation<br>1h (R/W) = Request local low-power mode |

#### 28.5.1.1.4 SCISETINT Register (Offset = 6h) [reset = 0h]

SCISETINT is shown in Figure 28-25 and described in Table 28-12.

Return to Summary Table.

Interrupt Enable Register

**Figure 28-25. SCISETINT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| SETBEINT | SETPBEINT | SETCEINT | SETISFEINT | SETNREINT | SETFEINT | SETOEINT | SETPEINT |
| 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 | |
| R-0h | | | | | R-0h | R-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | SETIDINT | RESERVED | | | SETRXINT | SETTXINT |
| R-0h | | R/W-0h | R-0h | | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SETTOA3WUSINT | SETTOAWUSINT | RESERVED | SETTIMEOUTINT | RESERVED | | SETWAKEUPINT | SETBRKDTINT |
| R/W-0h | R/W-0h | R-0h | R/W-0h | R-0h | | R/W-0h | R/W-0h |

**Table 28-12. SCISETINT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | SETBEINT | R/W | 0h | Set bit error interrupt.<br>This bit is effective in LIN mode only. Setting this bit enables the SCI/BLIN module to generate an interrupt when there is a bit error.<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |
| 30 | SETPBEINT | R/W | 0h | Set physical bus error interrupt.<br>This bit is effective in LIN mode only. Setting this bit enables the SCI/BLIN module to generate an interrupt when a physical bus error occurs.<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |

**Table 28-12. SCISETINT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 29 | SETCEINT | R/W | 0h | Set checksum-error Interrupt.<br>This bit is effective in LIN mode only. Setting this bit enables the SCI/BLIN module to generate an interrupt when there is a checksum error.<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |
| 28 | SETISFEINT | R/W | 0h | Set inconsistent-synch-field-error interrupt.<br>This bit is effective in LIN mode only. Setting this bit enables the SCI/BLIN module to generate an interrupt when there is an inconsistent synch field error.<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |
| 27 | SETNREINT | R/W | 0h | Set no-response-error interrupt.<br>This bit is effective in LIN mode only. Setting this bit enables the SCI/BLIN module to generate an interrupt when a no-response error occurs.<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |

**Table 28-12. SCISETINT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 26 | SETFEINT | R/W | 0h | Set framing-error interrupt.<br>This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/BLIN module to generate an interrupt when a framing error occurs.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |
| 25 | SETOEINT | R/W | 0h | Set overrun-error interrupt.<br>This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/BLIN module to generate an interrupt when an overrun error occurs.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |
| 24 | SETPEINT | R/W | 0h | Set parity interrupt.<br>This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/BLIN module to generate an interrupt when a parity error occurs.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |
| 23-19 | RESERVED | R | 0h | Reserved |
| 18 | RESERVED | R | 0h | Reserved |
| 17-16 | RESERVED | R | 0h | Reserved |
| 15-14 | RESERVED | R | 0h | Reserved |

**Table 28-12. SCISETINT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 13 | SETIDINT | R/W | 0h | LIN mode only.<br>This bit is set to enable interrupt once a valid matching identifier<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |
| 12-10 | RESERVED | R | 0h | Reserved |
| 9 | SETRXINT | R/W | 0h | Receiver interrupt enable.<br>Setting this bit enables the SCI to generate a receive interrupt after a frame has been completely received and the data is being transferred from SCIRXSHF to SCIRD.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |
| 8 | SETTXINT | R/W | 0h | Set Transmitter interrupt.<br>Setting this bit enables the SCI/ BLIN to generate a transmit interrupt as data is being transferred from SCITD to SCITXSHF and the TXRDY bit is being set.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |

**Table 28-12. SCISETINT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7 | SETTOA3WUSINT | R/W | 0h | Set Timeout After 3 Wakeup Signals interrupt.<br>LIN mode only. Setting this bit enables the BLIN to generate an interrupt when there is a timeout after 3 wakeup signals have been sent.<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |
| 6 | SETTOAWUSINT | R/W | 0h | Set Timeout After Wakeup Signal interrupt.<br>LIN mode only. Setting this bit enables the BLIN to generate an interrupt when there is a timeout after one wakeup signal has been sent.<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | SETTIMEOUTINT | R/W | 0h | Set timeout interrupt.<br>This bit is effective in LIN mode only. Setting this bit enables the SCI/BLIN to generate an interrupt when no LIN bus activity occurs for at least four seconds.<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |
| 3-2 | RESERVED | R | 0h | Reserved |

**Table 28-12. SCISETINT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | SETWAKEUPINT | R/W | 0h | Set wakeup interrupt.<br>This bit is effective in LIN or SCI-compatible mode. Setting this bit enables the SCI/BLIN to generate a wake-up interrupt and thereby exit low-power mode. The wakeup interrupt is asserted on falling edge of the wakeup pulse. If enabled, the wake-up interrupt is asserted when local low-power mode is requested while the receiver is busy or if a low level is detected on the SCIRX pin during low-power mode. Wakeup interrupt is not asserted upon a wakeup pulse if the module is not in power down mode.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |
| 0 | SETBRKDTINT | R/W | 0h | Set break-detect interrupt.<br>This bit is effective in SCI-compatible<br>mode only. Setting this bit enables the SCI/BLIN to generate an error interrupt if a break condition is detected on the SCIRX pin.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |

#### 28.5.1.1.5 SCICLEARINT Register (Offset = 8h) [reset = 0h]

SCICLEARINT is shown in Figure 28-26 and described in Table 28-13.

Return to Summary Table.

Interrupt Disable Register

**Figure 28-26. SCICLEARINT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CLRBEINT | CLRPBEINT | CLRCEINT | CLRISFEINT | CLRNREINT | CLRFEINT | CLROEINT | CLRPEINT |
| 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 | |
| R-0h | | | | | R-0h | R-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | CLRIDINT | RESERVED | | | CLRRXINT | CLRTXINT |
| R-0h | | R/W-0h | R-0h | | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CLRTOA3WUSINT | CLRTOAWUSINT | RESERVED | CLRTIMEOUTINT | RESERVED | | CLRWAKEUPINT | CLRBRKDTINT |
| R/W-0h | R/W-0h | R-0h | R/W-0h | R-0h | | R/W-0h | R/W-0h |

**Table 28-13. SCICLEARINT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | CLRBEINT | R/W | 0h | Clear Bit Error Interrupt.<br>LIN mode only. This bit disables the bit error interrupt when set.<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |
| 30 | CLRPBEINT | R/W | 0h | Clear Physical Bus Error Interrupt.<br>LIN mode only. This bit disables the physical-bus error interrupt when set.<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |

**Table 28-13. SCICLEARINT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 29 | CLRCEINT | R/W | 0h | Clear checksum-error Interrupt.<br>LIN mode only. This bit disables the checksum interrupt when set.<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |
| 28 | CLRISFEINT | R/W | 0h | Clear Inconsistent-Synch-Field-Error Interrupt.<br>LIN mode only. This bit disables the ISFE interrupt when set.<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |
| 27 | CLRNREINT | R/W | 0h | Clear No-Reponse-Error Interrupt.<br>LIN mode only. Setting this bit disables the BLIN module to generate an interrupt when there is a No-Response error.<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |

**Table 28-13. SCICLEARINT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 26 | CLRFEINT | R/W | 0h | Clear Framing-Error Interrupt.<br>Setting this bit disables the SCI/BLIN module to generate an interrupt when there is a Framing error<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |
| 25 | CLROEINT | R/W | 0h | Clear Overrun-Error Interrupt.<br>This bit disables the SCI/ BLIN overrun interrupt when set.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |
| 24 | CLRPEINT | R/W | 0h | Clear Parity Interrupt.<br>Setting this bit disables the BLIN /SCI Parity error interrupt.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |
| 23-19 | RESERVED | R | 0h | Reserved |
| 18 | RESERVED | R | 0h | Reserved |
| 17-16 | RESERVED | R | 0h | Reserved |
| 15-14 | RESERVED | R | 0h | Reserved |

**Table 28-13. SCICLEARINT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 13 | CLRIDINT | R/W | 0h | LIN mode only.<br>This bit disables the ID interrupt when set.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |
| 12-10 | RESERVED | R | 0h | Reserved |
| 9 | CLRRXINT | R/W | 0h | Clear Receiver interrupt.<br>This bit disables the receiver interrupt when set.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |
| 8 | CLRTXINT | R/W | 0h | Clear Transmitter interrupt.<br>This bit disables the transmitter interrupt when set.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |
| 7 | CLRTOA3WUSINT | R/W | 0h | Clear Timeout After 3 Wakeup Signals interrupt.<br>LIN mode only. Setting this bit disables the timeout after 3 wakeup signals interrupt.<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is disabled<br>Normal and EALLOW mode (write):<br>leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt is enabled<br>Normal and EALLOW mode (write):<br>enable interrupt |

**Table 28-13. SCICLEARINT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | CLRTOAWUSINT | R/W | 0h | Clear Timeout After Wakeup Signal interrupt.<br>LIN mode only. Setting this bit disables the timeout after one wakeup signal interrupt.<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = SCI-compatible mode<br>Reserved.<br>LIN mode<br>The node is in slave mode.<br>1h (R/W) = SCI-compatible mode<br>Enable clock to the SCI module.<br>LIN mode<br>The node is in master mode. |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | CLRTIMEOUTINT | R/W | 0h | Clear Timeout interrupt.<br>LIN mode only. Setting this bit disables the timeout (LIN bus idle) interrupt.<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = SCI-compatible mode<br>Reserved.<br>LIN mode<br>The node is in slave mode.<br>1h (R/W) = SCI-compatible mode<br>Enable clock to the SCI module.<br>LIN mode<br>The node is in master mode. |
| 3-2 | RESERVED | R | 0h | Reserved |
| 1 | CLRWAKEUPINT | R/W | 0h | Clear Wake-up interrupt.<br>This bit disables the wakeup interrupt when set.<br>Reset type: SYSRSn<br>0h (R/W) = SCI-compatible mode<br>Reserved.<br>LIN mode<br>The node is in slave mode.<br>1h (R/W) = SCI-compatible mode<br>Enable clock to the SCI module.<br>LIN mode<br>The node is in master mode. |

**Table 28-13. SCICLEARINT Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | CLRBRKDTINT | R/W | 0h | Clear Break-detect interrupt.<br>Compatibility mode. This bit disables the Break-detect interrupt when set.<br>This field is writable in SCI mode only.<br>Reset type: SYSRSn<br>0h (R/W) = SCI-compatible mode<br>Reserved.<br>LIN mode<br>The node is in slave mode.<br>1h (R/W) = SCI-compatible mode<br>Enable clock to the SCI module.<br>LIN mode<br>The node is in master mode. |

#### 28.5.1.1.6 SCISETINTLVL Register (Offset = Ah) [reset = 0h]

SCISETINTLVL is shown in Figure 28-27 and described in Table 28-14.

Return to Summary Table.

Set Interrupt Level Register

**Figure 28-27. SCISETINTLVL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| SETBEINTLVL | SETPBEINTLVL | SETCEINTLVL | SETISFEINTLVL | SETNREINTLVL | SETFEINTLVL | SETOEINTLVL | SETPEINTLVL |
| 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 | |
| R-0h | | | | | R-0h | R-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | SETIDINTLVL | RESERVED | | | SETRXINTOVO | SETTXINTLVL |
| R-0h | | R/W-0h | R-0h | | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SETTOA3WUSINTLVL | SETTOAWUSINTLVL | RESERVED | SETTIMEOUTINTLVL | RESERVED | | SETWAKEUPINTLVL | SETBRKDTINTLVL |
| R/W-0h | R/W-0h | R-0h | R/W-0h | R-0h | | R/W-0h | R/W-0h |

**Table 28-14. SCISETINTLVL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | SETBEINTLVL | R/W | 0h | Set Bit Error Interrupt Level.<br>LIN mode only.<br>Reset type: SYSRSn |
| 30 | SETPBEINTLVL | R/W | 0h | Set Physical Bus Error Interrupt Level.<br>LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |
| 29 | SETCEINTLVL | R/W | 0h | Set checksum-error Interrupt Level.<br>LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |

**Table 28-14. SCISETINTLVL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 28 | SETISFEINTLVL | R/W | 0h | Set Inconsistent-Synch-Field-Error Interrupt Level.<br>LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |
| 27 | SETNREINTLVL | R/W | 0h | Set No-Reponse-Error Interrupt Level.<br>LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |
| 26 | SETFEINTLVL | R/W | 0h | Set Framing-Error Interrupt Level.<br>Normal and EALLOW mode (read):<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |
| 25 | SETOEINTLVL | R/W | 0h | Set Overrun-Error Interrupt Level.<br>Normal and EALLOW mode (read):<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |

**Table 28-14. SCISETINTLVL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24 | SETPEINTLVL | R/W | 0h | Set Parity Error Interrupt Level.<br>Normal and EALLOW mode (read):<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |
| 23-19 | RESERVED | R | 0h | Reserved |
| 18 | RESERVED | R | 0h | Reserved |
| 17-16 | RESERVED | R | 0h | Reserved |
| 15-14 | RESERVED | R | 0h | Reserved |
| 13 | SETIDINTLVL | R/W | 0h | Set ID interrupt level.<br>LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |
| 12-10 | RESERVED | R | 0h | Reserved |
| 9 | SETRXINTOVO | R/W | 0h | Set Receiver interrupt Level.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |
| 8 | SETTXINTLVL | R/W | 0h | Set Transmitter interrupt Level.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |

**Table 28-14. SCISETINTLVL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7 | SETTOA3WUSINTLVL | R/W | 0h | Set Timeout After 3 Wakeup Signals interrupt Level.<br>LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |
| 6 | SETTOAWUSINTLVL | R/W | 0h | Set Timeout After Wakeup Signal interrupt Level.<br>LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | SETTIMEOUTINTLVL | R/W | 0h | Set Timeout interrupt Level.<br>LIN mode only<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |
| 3-2 | RESERVED | R | 0h | Reserved |
| 1 | SETWAKEUPINTLVL | R/W | 0h | Set Wake-up interrupt Level<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |

**Table 28-14. SCISETINTLVL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | SETBRKDTINTLVL | R/W | 0h | Set Break-detect interrupt Level.<br>Compatible mode only.<br>This field is writable in SCI mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |

#### 28.5.1.1.7 SCICLEARINTLVL Register (Offset = Ch) [reset = 0h]

SCICLEARINTLVL is shown in Figure 28-28 and described in Table 28-15.

Return to Summary Table.

Clear Interrupt Level Register

**Figure 28-28. SCICLEARINTLVL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CLRBEINTLVL | CLRPBEINTLVL | CLRCEINTLVL | CLRISFEINTLVL | CLRNREINTLVL | CLRFEINTLVL | CLROEINTLVL | CLRPEINTLVL |
| 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 | |
| R-0h | | | | | R-0h | R-0h | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | CLRIDINTLVL | RESERVED | | | CLRRXINTLVL | CLRTXINTLVL |
| R-0h | | R/W-0h | R-0h | | | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CLRTOA3WUSINTLVL | CLRTOAWUSINTLVL | RESERVED | CLRTIMEOUTINTLVL | RESERVED | | CLRWAKEUPINTLVL | CLRBRKDTINTLVL |
| R/W-0h | R/W-0h | R-0h | R/W-0h | R-0h | | R/W-0h | R/W-0h |

**Table 28-15. SCICLEARINTLVL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | CLRBEINTLVL | R/W | 0h | Clear Bit Error Interrupt Level.<br>LIN mode only.<br>Reset type: SYSRSn |
| 30 | CLRPBEINTLVL | R/W | 0h | Clear Physical Bus Error Interrupt Level<br>LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |
| 29 | CLRCEINTLVL | R/W | 0h | Clear checksum-error Interrupt Level. LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |

**Table 28-15. SCICLEARINTLVL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 28 | CLRISFEINTLVL | R/W | 0h | Clear Inconsistent-Synch-Field-Error Interrupt Level.<br>LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |
| 27 | CLRNREINTLVL | R/W | 0h | Clear No-Reponse-Error Interrupt Level.<br>LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |
| 26 | CLRFEINTLVL | R/W | 0h | Clear Framing-Error Interrupt Level.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |
| 25 | CLROEINTLVL | R/W | 0h | Clear Overrun-Error Interrupt Level.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |

**Table 28-15. SCICLEARINTLVL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24 | CLRPEINTLVL | R/W | 0h | Clear Parity Error Interrupt Level.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |
| 23-19 | RESERVED | R | 0h | Reserved |
| 18 | RESERVED | R | 0h | Reserved |
| 17-16 | RESERVED | R | 0h | Reserved |
| 15-14 | RESERVED | R | 0h | Reserved |
| 13 | CLRIDINTLVL | R/W | 0h | Clear ID interrupt level.<br>LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |
| 12-10 | RESERVED | R | 0h | Reserved |
| 9 | CLRRXINTLVL | R/W | 0h | Clear Receiver interrupt Level.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |
| 8 | CLRTXINTLVL | R/W | 0h | Clear Transmitter interrupt Level.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |

**Table 28-15. SCICLEARINTLVL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7 | CLRTOA3WUSINTLVL | R/W | 0h | Clear Timeout After 3 Wakeup Signals interrupt Level.<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |
| 6 | CLRTOAWUSINTLVL | R/W | 0h | Clear Timeout After Wakeup Signal interrupt Level.<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | CLRTIMEOUTINTLVL | R/W | 0h | Clear Timeout interrupt Level.<br>LIN mode only<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |
| 3-2 | RESERVED | R | 0h | Reserved |
| 1 | CLRWAKEUPINTLVL | R/W | 0h | Clear Wake-up interrupt Level<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |

**Table 28-15. SCICLEARINTLVL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | CLRBRKDTINTLVL | R/W | 0h | Clear Break-detect interrupt Level.<br>Compatible mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT0 line<br>Normal and EALLOW mode (write):<br>Leaves the corresponding bit unchanged<br>1h (R/W) = Normal and EALLOW mode (read):<br>Interrupt level mapped to INT1 line<br>Normal and EALLOW mode (write):<br>Set interrupt level to line INT1 |

#### 28.5.1.1.8 SCIFLR Register (Offset = Eh) [reset = 100h]

SCIFLR is shown in Figure 28-29 and described in Table 28-16.

Return to Summary Table.

Flag Register

**Figure 28-29. SCIFLR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| BE | PBE | CE | ISFE | NRE | FE | OE | PE |
| 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 | | | | | | | |
| R/W-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | IDRXFLAG | IDTXFLAG | RXWAKE | TXEMPTY | TXWAKE | RXRDY | TXRDY |
| R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-1h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TOA3WUS | TOAWUS | RESERVED | TIMEOUT | BUSY | IDLE | WAKEUP | BRKDT |
| R/W-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 28-16. SCIFLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | BE | R/W | 0h | Bit Error Flag.<br>This bit is set when there has been a bit error. This is detected by the bit monitor in TED. The Bit Error flag is cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register, by the RESET bit, by the SW nRESET bit, by a system reset or by writing a 1 to this bit.<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Read:<br>No bit error detected<br>Write:<br>No effect<br>1h (R/W) = Read:<br>Bit error detected<br>Write:<br>Clears this bit to 0 |
| 30 | PBE | R/W | 0h | Physical Bus Error Flag.<br>LIN mode only. This bit is set when there has been a physical bus error. This is detected by the bit monitor in TED. The Physical Bus Error flag is cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register, by the RESET bit, by the SW nRESET bit, by a system reset or by writing a 1 to this bit.<br>Reset type: SYSRSn<br>0h (R/W) = Read:<br>No physiscal bus error detected<br>Write:<br>No effect<br>1h (R/W) = Read:<br>physiscal bus error detected<br>Write:<br>Clears this bit to 0 |

**Table 28-16. SCIFLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 29 | CE | R/W | 0h | Checksum Error Flag.<br>LIN mode only. This bit is set when there is checksum error detected by a receiving node. The type of checksum to be used depends on the SCIGCR1.CTYPE bit. The Checksum Error flag is cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register, by the RESET bit, by the SW nRESET bit, by a system reset or by writing a 1 to this bit.<br>Reset type: SYSRSn<br>0h (R/W) = Read:<br>No Checksum error detected<br>Write:<br>No effect<br>1h (R/W) = Read:<br>Checksum error detected<br>Write:<br>Clears this bit to 0 |
| 28 | ISFE | R/W | 0h | Inconsistent Synch Field Error Flag.<br>LIN mode only. This bit is set when there has been an inconsistent Synch Field error detected by the Synchronizer during Header reception. See the "Header Reception and Adaptive Baudrate" section for more information. The Inconsistent Synch Field Error flag is cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register, by the RESET bit, by the SW nRESET bit, by a system reset or by writing a 1 to this bit.<br>Reset type: SYSRSn<br>0h (R/W) = Read:<br>No Inconsistent Synch Field error detected<br>Write:<br>No effect<br>1h (R/W) = Read:<br>Inconsistent Synch Field error detected<br>Write:<br>Clears this bit to 0 |
| 27 | NRE | R/W | 0h | No-Response Error Flag.<br>LIN mode only. This bit is set when there is no response to a master's Header completed within TFRAME_MAX. The No-Response Error flag is cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register, by the RESET bit, by the SW nRESET bit, by a system reset or by writing a 1 to this bit.<br>Reset type: SYSRSn<br>0h (R/W) = Read:<br>No No-Response error detected<br>Write:<br>No effect<br>1h (R/W) = Read:<br>No-Response error detected<br>Write:<br>Clears this bit to 0 |

**Table 28-16. SCIFLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 26 | FE | R/W | 0h | Framing error flag.<br>This bit is effective in LIN or SCI-compatible mode. This bit is set when an expected stop bit is not found. In SCI compatible mode, only the first stop bit is checked. The missing stop bit indicates that synchronization with the start bit has been lost and that the character is incorrectly framed. Detection of a framing error causes the SCI to generate an error interrupt if the RXERR INT ENA bit is set. The framing error flag is cleared by reading the corresponding interrupt offset in the<br>SCIINTVECT0/1 register, by the RESET bit, by the SW nRESET bit, by a system reset or by writing a 1 to this bit. In multibuffer mode the frame is defined in the SCIFORMAT register.<br>Reset type: SYSRSn<br>0h (R/W) = Read:<br>No framing error detected<br>Write:<br>No effect<br>1h (R/W) = Read:<br>framing error detected<br>Write:<br>Clears this bit to 0 |
| 25 | OE | R/W | 0h | Overrun error flag.<br>This bit is effective in LIN or SCI-compatible mode. This bit is set when the transfer of data from SCIRXSHF to SCIRD overwrites unread data already in SCIRD or the RDy buffers. Detection of an overrun error causes the LIN to generate an error interrupt if the SET OE INT bit is one. The OE flag is reset by reading the corresponding interrupt offset in the SCIINTVECT0/1 register, by the RESET bit, by an active SW nRESET, a system reset, or by writing a 1 to this bit.<br>Reset type: SYSRSn<br>0h (R/W) = Read:<br>No overrun error detected<br>Write:<br>No effect<br>1h (R/W) = Read:<br>Overrun error detected<br>Write:<br>Clears this bit to 0 |

**Table 28-16. SCIFLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24 | PE | R/W | 0h | Parity error flag.<br>This bit is effective in LIN or SCI-compatible mode. This bit is set when a parity error is detected in the received data. In address-bit mode, the parity is calculated on the data and address bit fields of the received frame. In idle-line mode, only the data is used to calculate parity. An error is generated<br>when a character is received with a mismatch between the number of 1s and its parity bit. For more information on parity checking, see the "SCI Global Control Register (SCIGCR1)" description. If the parity function is disabled (that is, SCIGCR1.2 = 0), the PE flag is disabled and read as 0.<br>Detection of a parity error causes the LIN to generate an error interrupt if the SET PE INT bit =1. The PE bit is reset by reading the corresponding interrupt offset in the SCIINTVECT0/1 register, by the RESET bit, by an active SW nRESET, a system reset or by writing a 1 to this bit.<br>Reset type: SYSRSn<br>0h (R/W) = Read:<br>No parity error or parity disabled<br>Write:<br>No effect<br>1h (R/W) = Read:<br>Parity error detected<br>Write:<br>Clears this bit to 0 |
| 23-16 | RESERVED | R/W | 0h | Reserved |
| 15 | RESERVED | R | 0h | Reserved |
| 14 | IDRXFLAG | R/W | 0h | Identifier On Receive Flag.<br>LIN mode only. This flag is set once an identifier is received with an RX match and no ID-parity error. See the "Message Filtering and Validation" section for more details. When this flag is set it indicates that a new valid identifier has been received on an RX match. This bit is cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register, by the RESET bit, by an active SW nRESET, a system reset, by reading the LINID register or by writing a 1 to this bit<br>Reset type: SYSRSn<br>0h (R/W) = Read<br>No valid ID received<br>Write<br>No effect<br>1h (R/W) = Read<br>Valid ID RX received in LINID[23:16] on RX match<br>Write<br>Clears this bit to 0 |

**Table 28-16. SCIFLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 13 | IDTXFLAG | R/W | 0h | Identifier On Transmit Flag.<br>LIN mode only. This flag is set once an identifier is received with a TX match and no ID-parity error. See the "Message Filtering and Validation" section for more details. When this flag is set it indicates that a new valid identifier has been received on a TX match. This bit is cleared by reading<br>the corresponding interrupt offset in the SCIINTVECT0/1 register, by the RESET bit, by an active SW nRESET, a system reset, by reading the LINID register or by writing a 1 to this bit.<br>Reset type: SYSRSn<br>0h (R/W) = Read<br>No valid ID received<br>Write<br>No effect<br>1h (R/W) = Read<br>Valid ID received in LINID[23:16] on TX match<br>Write<br>Clears this bit to 0 |
| 12 | RXWAKE | R/W | 0h | Receiver wakeup detect flag.<br>Compatible mode only. The SCI sets this bit to indicate that the data currently in SCIRD is an address. RXWAKE is cleared by the RESET bit, by an active SW nRESET, a system reset, or by the SCI upon receipt of a data frame.<br>Reset type: SYSRSn<br>0h (R/W) = The data in SCIRD is not an address.<br>1h (R/W) = The data in SCIRD is an address.<br>See [1] Section 3.4.4, Sleep Mode for Multiprocessor Communication, on page 16 for more information on using the RXWAKE bit with sleep mode. |
| 11 | TXEMPTY | R/W | 0h | Transmitter Empty flag.<br>The value of this flag indicates the contents of the transmitter's buffer register(s) (SCITD/TDy) and shift register (SCITXSHF). In multibuffer mode, this flag indicates the value of the TDx registers and shift register (SCITXSHF). In non multibuffer mode, this flag indicates the value of LINTDO (byte) and shift register (SCITXSHF).<br>The RESET bit, an active SW nRESET (SCIGCR1.7) or a system reset sets this bit. This bit does not cause an interrupt request.<br>Reset type: SYSRSn<br>0h (R/W) = Compatible mode or LIN with no multibuffer.<br>Transmitter buffer or shift register (or both) are loaded with data.<br>In LIN mode using multibuffer mode<br>Multibuffer or shift register (or all) are loaded with data<br>1h (R/W) = Compatible mode or LIN with no multibuffer.<br>Transmitter buffer and shift registers are both empty.<br>In LIN mode using multibuffer mode<br>Multibuffer and shift registers are all empty. |

**Table 28-16. SCIFLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 10 | TXWAKE | R/W | 0h | SCI transmitter wakeup method select.<br>Compatibility mode only. The TXWAKE bit controls whether the data in SCITD should be sent as an address or data frame using multiprocessor communication format. This bit is set to 1 or 0 by software before a byte is written to SCITD and is cleared by the SCI when data is transferred from<br>SCITD to SCITXSHF or by a system reset. TXWAKE is not cleared by the SW nRESET bit (SCIGCR1.7).<br>Reset type: SYSRSn<br>0h (R/W) = Address-bit mode:<br>Frame to be transmitted will be data (address bit = 0).<br>Idle-line mode:<br>Frame to be transmitted will be data+P338<br>1h (R/W) = Address-bit mode:<br>Frame to be transmitted will be an address (address bit=1).<br>Idle-line mode:<br>Following frame to be transmitted will be an address (writing a 1 to this bit followed by writing dummy data to the SCITD will result in a idle period of 11 bit periods before the next frame is transmitted). |
| 9 | RXRDY | R/W | 0h | Receiver ready flag.<br>In compatibility mode, the receiver sets this bit to indicate that the SCIRD contains new data and is ready to be read by the CPU. In LIN mode, RXRDY is set once a valid frame is received in multibuffer mode, a valid frame being a message frame received with no errors. In non multibuffer mode RXRDY is set for each received byte and will be set for the last byte of the frame if there are no errors. The SCI/BLIN generates a receive interrupt when RXRDY flag bit is set<br>if the interrupt-enable bit is set (SCISETINT.9). RXRDY is cleared by the RESET bit, by an active SW nRESET, a system reset, writing a 1 to this bit, by reading SCIRD in compatibility mode, or by reading last data byte RDy of the response in LIN mode.<br>Note: The RXRDY flag cannot be cleared by reading the corresponding interrupt offset in the<br>SCIINTVECT0/1 register.<br>Reset type: SYSRSn<br>0h (R/W) = Read:<br>No new data in SCIRD/Rdy<br>Write:<br>No effect<br>1h (R/W) = Read:<br>New data ready to be read from SCIRD<br>Write:<br>No effectClears this bit to 0 |

**Table 28-16. SCIFLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 8 | TXRDY | R/W | 1h | Transmitter buffer register ready flag.<br>When set, this bit indicates that the transmit buffer(s) register (SCITD in compatibility mode and LINTDO, LINTD1 in MBUF mode) is/are ready to get another character from a CPU write. In compatibility mode writing data to SCITD automatically clears this bit. In LIN mode, this bit is cleared once byte 0 (TD0) is written to LINTD0. This bit is set after the data of the TX buffer are shifted into the SCITXSHF register. This event can trigger a transmit DMA event if the DMA enable bit is set. TXRDY is also set to 1 by the RESET bit, by enabling SW nRESET (SCIGCR1.7) or by a system reset.<br>Note: The TXRDY flag cannot be cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register.<br>Note: The transmit interrupt request can be eliminated until the next series of data is written into the transmit buffers LINTD0 and LINTD1, by disabling the corresponding interrupt via the SCICLEARINT register or by disabling the transmitter via the TXENA bit (SCIGCR1.25=0).<br>Reset type: SYSRSn<br>0h (R/W) = Compatible mode:<br>SCITD is full.<br>LIN mode:<br>The multibuffers are full.<br>1h (R/W) = Compatible mode:<br>SCITD is ready to receive the next character<br>LIN mode:<br>The multibuffers are ready to receive the next character(s). |
| 7 | TOA3WUS | R/W | 0h | Timeout After 3 Wakeup Signals flag.<br>LIN mode only. This flag is set if there is no Synch Break received after 3 wakeup signals and a period of 1.5 seconds have passed. Such expiration time is used before issuing another round of wakeup signals.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>No timeout after 3 wakeup signals<br>Normal and EALLOW mode (write):<br>No effect<br>1h (R/W) = Normal and EALLOW mode (read):<br>Timeout after 3 wakeup signals and 1.5s time.<br>Normal and EALLOW mode (write):<br>Clears this bit to zero |

**Table 28-16. SCIFLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | TOAWUS | R/W | 0h | Timeout After Wakeup Signal flag.<br>LIN mode only. This bit is set if there is no Synch Break received after a wakeup signal has been sent. A minimum of 150ms expiration time is used before issuing another wakeup signal. This bit is cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register, by the RESET bit, by the SWnRESET bit, by a system reset, or by writing a 1 to this bit.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>No timeout after one wakeup signal (150 ms)<br>Normal and EALLOW mode (write):<br>No effect<br>1h (R/W) = Normal and EALLOW mode (read):<br>Timeout after one wakeup signal<br>Normal and EALLOW mode (write):<br>Clears this bit to zero |
| 5 | RESERVED | R | 0h | Reserved |
| 4 | TIMEOUT | R/W | 0h | LIN Bus IDLE timeout flag.<br>LIN mode only. This bit is set if there is no LIN bus activity for at least 4 seconds. LIN bus activity being a transition from recessive to dominant. This bit is cleared by reading the corresponding interrupt offset in the SCIINTVECT0/1 register, by the RESET bit, by the SWnRESET bit, by a system reset, or by writing a 1 to this bit.<br>Reset type: SYSRSn<br>0h (R/W) = Normal and EALLOW mode (read):<br>No bus idle detected<br>Normal and EALLOW mode (write):<br>No effect<br>1h (R/W) = Normal and EALLOW mode (read):<br>LIN bus idle detected<br>Normal and EALLOW mode (write):<br>Clears this bit to zero |
| 3 | BUSY | R/W | 0h | Bus BUSY flag.<br>This bit is effective in LIN mode and SCI-compatible mode. This bit indicates whether the receiver is in the process of receiving a frame. As soon as the receiver detects the beginning of a start bit, the BUSY bit is set to 1. When the reception of a frame is complete, the BUSY bit is cleared. If SET WAKEUP INT is set and power down is requested while this bit is set, the SCI/BLIN automatically prevents low-power mode from being entered and generates wakeup interrupt. The BUSY bit is controlled directly by the SCI receiver but is cleared by the RESET bit, by an active SWnRESET or by a system reset.<br>Reset type: SYSRSn<br>0h (R/W) = Receiver is not currently receiving a frame.<br>1h (R/W) = Receiver is currently receiving a frame |

**Table 28-16. SCIFLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | IDLE | R/W | 0h | SCI receiver in idle state.<br>Compatible mode only. While this bit is set, the SCI looks for an idle period to resynchronize itself with the bit stream. The receiver does not receive any data while the bit is set. The bus must be idle for 11 bit periods to clear this bit. The SCI enters this state:<br>- After a system reset<br>- After a SCI software reset<br>- After coming out of power down<br>Reset type: SYSRSn<br>0h (R/W) = Idle period detected, the SCI is ready to receive.<br>1h (R/W) = Idle period not detected, the SCI will not receive any data. |
| 1 | WAKEUP | R/W | 0h | Wake-up flag.<br>This bit is set by the SCI/BLIN when receiver or transmitter activity has taken the module out of power-down mode. An interrupt is generated if the SET WAKEUP INT bit (SCISETINT.1) is set. It is cleared by the following:<br>- Reading the corresponding interrupt offset in the SCIINTVECT0/1 register.<br>- Setting the SWnRESET bit (SCIGCR1.7<br>- RESET bit (SCIGCR0.0)<br>- System reset<br>- By writing a 1 to this bit.<br>This field is writable in LIN mode only.<br>Reset type: SYSRSn<br>0h (R/W) = Read:<br>Do not wake up from power-down mode<br>Write:<br>No effect<br>1h (R/W) = Read:<br>Wake up from power-down mode<br>Write:<br>Clears this bit to zero |

**Table 28-16. SCIFLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | BRKDT | R/W | 0h | SCI break-detect flag.<br>Compatible mode only This bit is set when the SCI detects a break condition on the LINRX pin. A break condition occurs when the LINRX pin remains continuously low for at least 10 bits after a missing first stop bit, that is, after a framing error. Detection of a break condition causes the SCI to generate an error interrupt if the BRKDT INT ENA bit is set. The BRKDT bit is cleared by the following:<br>- Reading the corresponding interrupt offset in the SCIINTVECT0/1 register.<br>- Setting the SWnRESET bit (SCIGCR1.7)<br>- RESET bit (SCIGCR0.0)<br>- System reset<br>- By writing a 1 to this bit.<br>Reset type: SYSRSn<br>0h (R/W) = Read:<br>No break condition detected<br>Write:<br>No effect<br>1h (R/W) = Read:<br>Break condition detected<br>Write:<br>Clears this bit to 0 |

#### 28.5.1.1.9 SCIINTVECT0 Register (Offset = 10h) [reset = 0h]

SCIINTVECT0 is shown in Figure 28-30 and described in Table 28-17.

Return to Summary Table.

Interrupt Vector Offset Register 0

**Figure 28-30. SCIINTVECT0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | | | | | INTVECT0 | | | | |
| R-0h | | | | | | | | | | | R-0h | | | | |

**Table 28-17. SCIINTVECT0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-5 | RESERVED | R | 0h | Reserved |
| 4-0 | INTVECT0 | R | 0h | Interrupt vector offset for INT0.<br>This register indicates the offset for interrupt line INT0. A read to this register updates its value to the next highest priority pending interrupt in SCIFLR and clears the flag corresponding to the offset that was read.<br>Note: The flags for the receive (SCIFLR.9) and the transmit (SCIFLR.8) interrupts cannot be cleared by reading the corresponding offset vector in this register (see detailed description in SCIFLR register).<br>Reset type: SYSRSn |

#### 28.5.1.1.10 SCIINTVECT1 Register (Offset = 12h) [reset = 0h]

SCIINTVECT1 is shown in Figure 28-31 and described in Table 28-18.

Return to Summary Table.

Interrupt Vector Offset Register 1

**Figure 28-31. SCIINTVECT1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | | | | | INTVECT1 | | | | |
| R-0h | | | | | | | | | | | R-0h | | | | |

**Table 28-18. SCIINTVECT1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-5 | RESERVED | R | 0h | Reserved |
| 4-0 | INTVECT1 | R | 0h | Interrupt vector offset for INT10.<br>This register indicates the offset for interrupt line INT0. A read to this register updates its value to the next highest priority pending interrupt in SCIFLR and clears the flag corresponding to the offset that was read.<br>Note: The flags for the receive (SCIFLR.9) and the transmit (SCIFLR.8) interrupts cannot be cleared by reading the corresponding offset vector in this register (see detailed description in SCIFLR register).<br>Reset type: SYSRSn |

#### 28.5.1.1.11 SCIFORMAT Register (Offset = 14h) [reset = 0h]

SCIFORMAT is shown in Figure 28-32 and described in Table 28-19.

Return to Summary Table.

Length Control Register

**Figure 28-32. SCIFORMAT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | LENGTH | | |
| R-0h | | | | | | | | | | | | | R/W-0h | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | | | | | CHAR | | | | |
| R-0h | | | | | | | | | | | R/W-0h | | | | |

**Table 28-19. SCIFORMAT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-19 | RESERVED | R | 0h | Reserved |
| 18-16 | LENGTH | R/W | 0h | Frame length control bits.<br>In LIN mode, these bits indicate the number of bytes in the response field from 1 to 8 bytes. In buffered SCI mode, these bits indicate the number of characters. When these bits are used to indicate LIN response length (SCIGCR1[0] = 1), then when there is an ID RX match, this value should be updated with the expected length of the response. In buffered SCI mode, these bits indicate the number of characters with SCIFORMAT[2:0] bits per character. i.e. these bits indicate the transmitter/receiver format for the number of characters: 1 to 8. There can be up to eight<br>characters with eight bits each<br>Reset type: SYSRSn<br>0h (R/W) = The response field has 1 bytes/characters.<br>1h (R/W) = The response field has 2 bytes/characters.<br>2h (R/W) = The response field has 3 bytes/characters.<br>3h (R/W) = The response field has 4 bytes/characters.<br>4h (R/W) = The response field has 5 bytes/characters.<br>5h (R/W) = The response field has 6 bytes/characters.<br>6h (R/W) = The response field has 7 bytes/characters.<br>7h (R/W) = The response field has 8 bytes/characters. |
| 15-5 | RESERVED | R | 0h | Reserved |
| 4-0 | CHAR | R/W | 0h | Character length control bits.<br>These bits are effective in SCI compatible or buffered SCI modes only. These bits set the SCI character length from 1 to 8 bits.<br>Note: In compatibility mode or buffered SCI mode, when data of fewer than eight bits in length is received, it is left justified in SCIRD/RDy and padded with trailing zeros. Data read from the SCIRD should be shifted by software to make the received data right justified.<br>Note: Data written to the SCITD should be right justified but does not need to be padded with leading zero<br>Reset type: SYSRSn<br>0h (R/W) = The character is 1 bits long.<br>1h (R/W) = The character is 2 bits long.<br>2h (R/W) = The character is 3 bits long.<br>3h (R/W) = The character is 4 bits long.<br>4h (R/W) = The character is 5 bits long.<br>5h (R/W) = The character is 6 bits long.<br>6h (R/W) = The character is 7 bits long.<br>7h (R/W) = The character is 8 bits long. |

#### 28.5.1.1.12 BRSR Register (Offset = 16h) [reset = 0h]

BRSR is shown in Figure 28-33 and described in Table 28-20.

Return to Summary Table.

Baud Rate Selection Register

**Figure 28-33. BRSR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | M | | | | SCI_LIN_PSH | | | | | | | |
| R-0h | | | | R/W-0h | | | | R/W-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SCI_LIN_PSL | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 28-20. BRSR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-28 | RESERVED | R | 0h | Reserved |
| 27-24 | M | R/W | 0h | SCI/BLIN 4-bit Fractional Divider Selection.<br>These bits are effective in LIN or SCI asynchronous mode. These bits are used to select a baud rate for the SCI/BLIN module, and they are a fractional part for the baud rate specification. The M divider allows fine-tuning of the baud rate over the P prescaler with 15 additional intermediate values for each of the P integer values.<br>Reset type: SYSRSn |
| 23-16 | SCI_LIN_PSH | R/W | 0h | PRESCALER . (High Bits)<br>SCI/BLIN 24-bit Integer Prescaler Selection.<br>These bits are used to select a baudrate for the SCI/BLIN module. These bits are effective in LIN mode and SCI compatibility mode. The SCI/BLIN has an internally generated serial clock determined by the LIN module input clock and the prescalers P and M in this register. The SCI/LIN uses the 24-bit integer prescaler P value to select 1 of over 16,700,000<br>available baud rates. The additional 4-bit fractional prescaler M refines the baudate selection.<br>Reset type: SYSRSn |
| 15-0 | SCI_LIN_PSL | R/W | 0h | PRESCALER . (Low Bits)<br>SCI/BLIN 24-bit Integer Prescaler Selection.<br>These bits are used to select a baudrate for the SCI/BLIN module. These bits are effective in LIN mode and SCI compatibility mode. The SCI/BLIN has an internally generated serial clock determined by the LIN module input clock and the prescalers P and M in this register. The SCI/LIN uses the 24-bit integer prescaler P value to select 1 of over 16,700,000<br>available baud rates. The additional 4-bit fractional prescaler M refines the baudate selection.<br>Reset type: SYSRSn |

#### 28.5.1.1.13 SCIED Register (Offset = 18h) [reset = 0h]

SCIED is shown in Figure 28-34 and described in Table 28-21.

Return to Summary Table.

Emulation buffer Register

**Figure 28-34. SCIED Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | | | | | | | | | ED | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | R-0h | | | | | | | |

**Table 28-21. SCIED Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-8 | RESERVED | R | 0h | Reserved |
| 7-0 | ED | R | 0h | Receiver Emulation Data.<br>This bit is effective in SCI-compatible mode only. Reading SCIED(7-0) does not clear the RXRDY flag. This register should be used only by an emulator that must continually read the data buffer without affecting the RXRDY flag.<br>Reset type: SYSRSn |

#### 28.5.1.1.14 SCIRD Register (Offset = 1Ah) [reset = 0h]

SCIRD is shown in Figure 28-35 and described in Table 28-22.

Return to Summary Table.

Receiver data buffer Register

**Figure 28-35. SCIRD Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | | | | | | | | | RD | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | R/W-0h | | | | | | | |

**Table 28-22. SCIRD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-8 | RESERVED | R | 0h | Reserved |
| 7-0 | RD | R/W | 0h | Received Data.<br>This bit is effective in SCI-compatible mode only. When a frame has been completely received, the data in the frame is transferred from the receiver shift register SCIRXSHF to this register. As this transfer occurs, the RXRDY flag is set and a receive interrupt is generated if RX INT ENA<br>(SCISETINT0.9) is set. When the data is read from SCIRD, the RXRDY flag is automatically cleared.<br>Reset type: SYSRSn |

#### 28.5.1.1.15 SCITD Register (Offset = 1Ch) [reset = 0h]

SCITD is shown in Figure 28-36 and described in Table 28-23.

Return to Summary Table.

Transmit data buffer Register

**Figure 28-36. SCITD Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | | | | | | | | | TD | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | R/W-0h | | | | | | | |

**Table 28-23. SCITD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-8 | RESERVED | R | 0h | Reserved |
| 7-0 | TD | R/W | 0h | Transmit data<br>This bit is effective in SCI-compatible mode only. The transfer of data from this register to the transmit shift register SCITXSHF sets the TXRDY flag (SCIFLR.23), which indicates that SCITD is ready to be loaded with another byte of data. Note: If TX INT ENA (SCISETINT.8) is set, this data transfer also causes an interrupt.<br>Reset type: SYSRSn |

#### 28.5.1.1.16 SCIPIO0 Register (Offset = 1Eh) [reset = 0h]

SCIPIO0 is shown in Figure 28-37 and described in Table 28-24.

Return to Summary Table.

Pin control Register 0

**Figure 28-37. SCIPIO0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | TXFUNC | RXFUNC | RESERVED |
| R-0h | | | | | R-0h | R-0h | R-0h |

**Table 28-24. SCIPIO0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-3 | RESERVED | R | 0h | Reserved |
| 2 | TXFUNC | R | 0h | Transfer function.<br>This bit is effective in LIN or SCI mode. This bit defines the function of LINTX pin.<br>Reset type: SYSRSn<br>0h (R/W) = LINTX pin is disabled.<br>1h (R/W) = LINTX pin is enabled. |
| 1 | RXFUNC | R | 0h | Transfer function.<br>This bit is effective in LIN or SCI mode. This bit defines the function of the LINRX pin.<br>Reset type: SYSRSn<br>0h (R/W) = LINRX pin is disabled.<br>1h (R/W) = LINRX pin is enabled. |
| 0 | RESERVED | R | 0h | Reserved |

#### 28.5.1.1.17 SCIPIO2 Register (Offset = 22h) [reset = 0h]

SCIPIO2 is shown in Figure 28-38 and described in Table 28-25.

Return to Summary Table.

Pin control Register 2

**Figure 28-38. SCIPIO2 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | TXIN | RXIN | RESERVED |
| R-0h | | | | | R-0h | R-0h | R-0h |

**Table 28-25. SCIPIO2 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-3 | RESERVED | R | 0h | Reserved |
| 2 | TXIN | R | 0h | Transmit data in.<br>This bit is effective in LIN or SCI-compatible mode. This bit contains the current value on the LINTX pin.<br>Reset type: SYSRSn |
| 1 | RXIN | R | 0h | Receive data in.<br>This bit is effective in LIN or SCI-compatible mode. This bit contains the current value on the LINRX pin<br>Reset type: SYSRSn |
| 0 | RESERVED | R | 0h | Reserved |

#### 28.5.1.1.18 LINCOMP Register (Offset = 30h) [reset = 0h]

LINCOMP is shown in Figure 28-39 and described in Table 28-26.

Return to Summary Table.

Compare register

**Figure 28-39. LINCOMP Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | SDEL | | RESERVED | | | | | SBREAK | | |
| R-0h | | | | | | R/W-0h | | R-0h | | | | | R/W-0h | | |

**Table 28-26. LINCOMP Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | RESERVED | R | 0h | Reserved |
| 15-10 | RESERVED | R | 0h | Reserved |
| 9-8 | SDEL | R/W | 0h | 2-bit synch delimiter compare.<br>These bits are effective in LIN mode only. These bits are used to configure the number of Tbit for the synch delimiter in the synch field.<br>The time delay value for the synchronization delimiter is<br>TSDEL = (SDEL + 1)Tbit<br>Reset type: SYSRSn<br>0h (R/W) = The synch delimiter has 1 Tbit.<br>1h (R/W) = The synch delimiter has 2 Tbit.<br>2h (R/W) = The synch delimiter has 3 Tbit.<br>3h (R/W) = The synch delimiter has 4 Tbit. |
| 7-3 | RESERVED | R | 0h | Reserved |
| 2-0 | SBREAK | R/W | 0h | 3-bit Synch Break Extend.<br>LIN mode only. These bits are used to configure the number of Tbits for the synch break to extend the minimum 13 Tbit in the Synch Field to a maximum of 20 Tbit.<br>Note: The default value is 0x0, which adds nothing to the automatically generated SYNCH BREAK.<br>The time delay for the SYNCH BREAK is<br>TSYNBRK = 13Tbit + SBREAK x Tbit<br>Reset type: SYSRSn<br>0h (R/W) = The synch break has no additional Tbit.<br>1h (R/W) = The synch break has 1 additional Tbit.<br>2h (R/W) = The synch break has 2 additional Tbit.<br>3h (R/W) = The synch break has 3 additional Tbit.<br>4h (R/W) = The synch break has 4 additional Tbit.<br>5h (R/W) = The synch break has 5 additional Tbit.<br>6h (R/W) = The synch break has 6 additional Tbit.<br>7h (R/W) = The synch break has 7 additional Tbit. |

#### 28.5.1.1.19 LINRD0 Register (Offset = 32h) [reset = 0h]

LINRD0 is shown in Figure 28-40 and described in Table 28-27.

Return to Summary Table.

Receive data register 0

**Figure 28-40. LINRD0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RD0 | | | | | | | | RD1 | | | | | | | | RD2 | | | | | | | | RD3 | | | | | | | |
| R-0h | | | | | | | | R-0h | | | | | | | | R-0h | | | | | | | | R-0h | | | | | | | |

**Table 28-27. LINRD0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | RD0 | R | 0h | 8-bit Receive Buffer 0<br>Each response data-byte that is received in the SCIRXSHFT register is transferred to the<br>corresponding RDy register according to the number of bytes received.<br>Reset type: SYSRSn |
| 23-16 | RD1 | R | 0h | 8-bit Receive Buffer 1.<br>Each response data-byte that is received in the SCIRXSHFT register is transferred to the<br>corresponding RDy register according to the number of bytes received.<br>Reset type: SYSRSn |
| 15-8 | RD2 | R | 0h | 8-bit Receive Buffer 2.<br>Each response data-byte that is received in the SCIRXSHFT register is transferred to the<br>corresponding RDy register according to the number of bytes received.<br>Reset type: SYSRSn |
| 7-0 | RD3 | R | 0h | 8-bit Receive Buffer 3.<br>Each response data-byte that is received in the SCIRXSHFT register is transferred to the<br>corresponding RDy register according to the number of bytes received.<br>A read of this byte clears the RXDY byte.<br>Note: RD<x-1> is equivalent to Data byte <x> of the LIN frame.<br>Reset type: SYSRSn |

#### 28.5.1.1.20 LINRD1 Register (Offset = 34h) [reset = 0h]

LINRD1 is shown in Figure 28-41 and described in Table 28-28.

Return to Summary Table.

Receive data register 1

**Figure 28-41. LINRD1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RD4 | | | | | | | | RD5 | | | | | | | | RD6 | | | | | | | | RD7 | | | | | | | |
| R-0h | | | | | | | | R-0h | | | | | | | | R-0h | | | | | | | | R-0h | | | | | | | |

**Table 28-28. LINRD1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | RD4 | R | 0h | 8-bit Receive Buffer 4<br>Each response data-byte that is received in the SCIRXSHFT register is transferred to the<br>corresponding RDy register according to the number of bytes received.<br>Reset type: SYSRSn |
| 23-16 | RD5 | R | 0h | 8-bit Receive Buffer 5.<br>Each response data-byte that is received in the SCIRXSHFT register is transferred to the<br>corresponding RDy register according to the number of bytes received.<br>Reset type: SYSRSn |
| 15-8 | RD6 | R | 0h | 8-bit Receive Buffer 6.<br>Each response data-byte that is received in the SCIRXSHFT register is transferred to the<br>corresponding RDy register according to the number of bytes received.<br>Reset type: SYSRSn |
| 7-0 | RD7 | R | 0h | 8-bit Receive Buffer 7.<br>Each response data-byte that is received in the SCIRXSHFT register is transferred to the<br>corresponding RDy register according to the number of bytes received.<br>A read of this byte clears the RXDY byte.<br>Note: RD<x-1> is equivalent to Data byte <x> of the LIN frame.<br>Reset type: SYSRSn |

#### 28.5.1.1.21 LINMASK Register (Offset = 36h) [reset = 0h]

LINMASK is shown in Figure 28-42 and described in Table 28-29.

Return to Summary Table.

Acceptance mask register

**Figure 28-42. LINMASK Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | RXIDMASK | | | | | | | | RESERVED | | | | | | | | TXIDMASK | | | | | | | |
| R-0h | | | | | | | | R-0h | | | | | | | | R-0h | | | | | | | | R-0h | | | | | | | |

**Table 28-29. LINMASK Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | RESERVED | R | 0h | Reserved |
| 23-16 | RXIDMASK | R | 0h | Receive ID mask.<br>LIN mode only. This 8-bit mask is used for filtering an incoming ID message and compare it to the ID-byte. A compare match of the received ID witht the RX ID mask will set the ID RX flag and trigger and ID interrupt if enabled. A 0 bit in the mask indicates that bit is compared to the ID-byte. A 1 bit in the mask indicates that that bit is filtered and therefore not used in the compare.<br>Reset type: SYSRSn |
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-0 | TXIDMASK | R | 0h | Transmit ID mask.<br>LIN mode only. This 8-bit mask is used for filtering an incoming ID message and compare it to the ID-byte. A compare match of the received ID with the TX ID Mask will set the ID TX flag and trigger an ID interrupt if enabled. A 0 bit in the mask indicates that bit is compared to the ID-byte. A 1 bit in the mask indicates that bit is filtered and therefore not used for the compare.<br>Reset type: SYSRSn |

#### 28.5.1.1.22 LINID Register (Offset = 38h) [reset = 0h]

LINID is shown in Figure 28-43 and described in Table 28-30.

Return to Summary Table.

LIN ID Register

**Figure 28-43. LINID Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | RECEIVEDID | | | | | | | |
| R-0h | | | | | | | | R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IDSLAVETASKBYTE | | | | | | | | IDBYTE | | | | | | | |
| R/W-0h | | | | | | | | R/W-0h | | | | | | | |

**Table 28-30. LINID Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | RESERVED | R | 0h | Reserved |
| 23-16 | RECEIVEDID | R | 0h | LIN mode only.<br>This byte contains the current message identifier. During header reception the received ID is copied from the SCIRXSHF register to this byte if there is no ID-parity error and there has been an RX/TX match.<br>Reset type: SYSRSn |
| 15-8 | IDSLAVETASKBYTE | R/W | 0h | LIN mode only.<br>This byte contains the identifier to which the received ID of an incoming Header will be compared in order to decide whether a RX response, a TX response or no action needs to be done by the LIN+L484 node.<br>Reset type: SYSRSn |
| 7-0 | IDBYTE | R/W | 0h | LIN mode only.<br>This byte is the LIN mode message ID. On a master node, a write to this register by the CPU initiates a header transmission. For a slave task, this byte is used for message filtering when HGENCTRL (SCIGCR1.12) is '0'.<br>Reset type: SYSRSn |

#### 28.5.1.1.23 LINTD0 Register (Offset = 3Ah) [reset = 0h]

LINTD0 is shown in Figure 28-44 and described in Table 28-31.

Return to Summary Table.

Transmit Data Register 0

**Figure 28-44. LINTD0 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TD0 | | | | | | | | TD1 | | | | | | | | TD2 | | | | | | | | TD3 | | | | | | | |
| R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | |

**Table 28-31. LINTD0 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | TD0 | R/W | 0h | 8-bit Transmit Buffer 0.<br>Byte 0 to be transmitted is written into this register and then copied to SCITXSHF for transmission. Once byte 0 is written in TDO buffer, transmission will be initiated.<br>Reset type: SYSRSn |
| 23-16 | TD1 | R/W | 0h | 8-bit Transmit Buffer 3.<br>Byte 1 to be transmitted is written into this register and then copied to SCITXSHF for transmission. Once byte 0 is written in TDO buffer, transmission will be initiated.<br>Reset type: SYSRSn |
| 15-8 | TD2 | R/W | 0h | 8-bit Transmit Buffer 2.<br>Byte 2 to be transmitted is written into this register and then copied to SCITXSHF for transmission. Once byte 0 is written in TDO buffer, transmission will be initiated.<br>Reset type: SYSRSn |
| 7-0 | TD3 | R/W | 0h | 8-bit Transmit Buffer 3.<br>Byte 3 to be transmitted is written into this register and then copied to SCITXSHF for transmission. Once byte 0 is written in TDO buffer, transmission will be initiated.<br>Reset type: SYSRSn |

#### 28.5.1.1.24 LINTD1 Register (Offset = 3Ch) [reset = 0h]

LINTD1 is shown in Figure 28-45 and described in Table 28-32.

Return to Summary Table.

Transmit Data Register 1

**Figure 28-45. LINTD1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TD4 | | | | | | | | TD5 | | | | | | | | TD6 | | | | | | | | TD7 | | | | | | | |
| R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | | R/W-0h | | | | | | | |

**Table 28-32. LINTD1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-24 | TD4 | R/W | 0h | 8-bit Transmit Buffer 4.<br>Byte4 to be transmitted is written into this register and then copied to SCITXSHF for transmission. Once byte 0 is written in TDO buffer, transmission will be initiated.<br>Reset type: SYSRSn |
| 23-16 | TD5 | R/W | 0h | 8-bit Transmit Buffer 5.<br>Byte 5 to be transmitted is written into this register and then copied to SCITXSHF for transmission. Once byte 0 is written in TDO buffer, transmission will be initiated.<br>Reset type: SYSRSn |
| 15-8 | TD6 | R/W | 0h | 8-bit Transmit Buffer 6.<br>Byte 6 to be transmitted is written into this register and then copied to SCITXSHF for transmission. Once byte 0 is written in TDO buffer, transmission will be initiated.<br>Reset type: SYSRSn |
| 7-0 | TD7 | R/W | 0h | 8-bit Transmit Buffer 7.<br>Byte 7 to be transmitted is written into this register and then copied to SCITXSHF for transmission. Once byte 0 is written in TDO buffer, transmission will be initiated.<br>Reset type: SYSRSn |

#### 28.5.1.1.25 MBRSR Register (Offset = 3Eh) [reset = DACh]

MBRSR is shown in Figure 28-46 and described in Table 28-33.

Return to Summary Table.

Baud Rate Selection Register

**Figure 28-46. MBRSR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | | | | | MBR | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | R/W-DACh | | | | | | | | | | | |

**Table 28-33. MBRSR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-13 | RESERVED | R | 0h | Reserved |
| 12-0 | MBR | R/W | DACh | Maximum Baud Rate Prescaler.<br>LIN mode only. This 13-bit prescaler is used during the synchronization phase (see the "Header Reception and Adaptive Baudrate" section) of a slave module if the ADAPT bit is set. In this way, a BLIN slave using an automatic or select bit rate modes detects any LIN bus legal rate automatically.<br>The MBR value should be programmed to allow a maximum baud rate that is not more than 10% above the expected operating baud rate in the LIN network. Otherwise a s 0x00 data byte could misleadingly be detected as synchbreak.<br>The default value is for a 30MHz LINCLK (0xDAC).<br>This MBR prescaler is used by the wake-up and idle time counters for a constant expiration<br>time relative to a 20kHz rate<br>Reset type: SYSRSn |

### 28.5.1.1.26 IODFTCTRL Register (Offset = 48h) [reset = 0h]

IODFTCTRL is shown in Figure 28-47 and described in Table 28-34.

Return to Summary Table.

IODFT for LIN

**Figure 28-47. IODFTCTRL Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| BERRENA | PBERRENA | CERRENA | ISFERRENA | RESERVED | FERRENA | PERRENA | BRKDTERRENA |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R-0h | R/W-0h | R/W-0h | R/W-0h |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | PINSAMPLEMASK | | TXSHIFT | | |
| R/W-0h | | | R/W-0h | | R/W-0h | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | IODFTENA | | | |
| R-0h | | | | R/W-0h | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | LPBENA | RXPENA |
| R-0h | | | | | | R/W-0h | R/W-0h |

**Table 28-34. IODFTCTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31 | BERRENA | R/W | 0h | LIN Mode Only<br>This bit is used to create a Bit error. When this bit is set, the bit received is ORed with 1 and passed to the Bit monitor circuitry.<br>Reset type: SYSRSn |
| 30 | PBERRENA | R/W | 0h | LIN Mode only<br>This bit is used to create a Physical Bus Error. When this bit is set, the bit received during Synch Break field transmission is ORed with 1 and passed to the Bit monitor circuitry<br>Reset type: SYSRSn |
| 29 | CERRENA | R/W | 0h | LIN Mode only<br>This bit is used to create a checksum error. When this bit is set, the polarity of the CTYPE (checksum type) in the receive checksum calculator is changed so that a checksum error is occurred<br>Reset type: SYSRSn |
| 28 | ISFERRENA | R/W | 0h | LIN Mode only<br>This bit is used to create an ISF error. When this bit is set, the bit widths in the synch field are varied so that the ISF check fails and the error flag is set.<br>Reset type: SYSRSn |
| 27 | RESERVED | R | 0h | Reserved |
| 26 | FERRENA | R/W | 0h | This bit is used to create a Frame Error.<br>When this bit is set, the stop bit received is ANDed with '0' and passed to the stop bit check circuitry.<br>Reset type: SYSRSn |
| 25 | PERRENA | R/W | 0h | Compatible Mode only<br>This bit is used to create a Parity Error. When this bit is set, in compatible mode, the parity bit received is toggled so that a parity error occurs.<br>Reset type: SYSRSn |

**Table 28-34. IODFTCTRL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 24 | BRKDTERRENA | R/W | 0h | Compatible Mode only<br>This bit is used to create BRKDT error (SCI mode only). When this bit is set, the stop bit of the frame is ANDed with '0' and passed to the RSM so that a frame error occurs. Then the RX Pin is forced to continuous low for 10 Tbits so that a BRKDT error occurs.<br>Reset type: SYSRSn |
| 23-21 | RESERVED | R/W | 0h | Reserved |
| 20-19 | PINSAMPLEMASK | R/W | 0h | These bits define the sample number at which the TX Pin value that is being transmitted will be inverted to verify the receive pin samples correctly with the majority detection circuitry.<br>Reset type: SYSRSn<br>0h (R/W) = No Mask<br>1h (R/W) = Invert the TX Pin value at TBIT_CENTER<br>2h (R/W) = Invert the TX Pin value at TBIT_CENTER + SCLK<br>3h (R/W) = Invert the TX Pin value at TBIT_CENTER + 2 SCLK |
| 18-16 | TXSHIFT | R/W | 0h | Transmit shift.<br>These bits define the delay by which the value on TX pin is delayed so that the value on RX Pin is asynchronous. (Not applicable to Start Bit)<br>Reset type: SYSRSn<br>0h (R/W) = No Delay<br>1h (R/W) = Delay by 1 SCLK<br>2h (R/W) = Delay by 2 SCLK<br>3h (R/W) = Delay by 3 SCLK<br>4h (R/W) = Delay by 4 SCLK<br>5h (R/W) = Delay by 5 SCLK<br>6h (R/W) = Delay by 6 SCLK<br>7h (R/W) = Delay by 7 SCLK |
| 15-12 | RESERVED | R | 0h | Reserved |
| 11-8 | IODFTENA | R/W | 0h | IO DFT Enable Key<br>Normal and EALLOW mode reads.<br>Write only in EALLOW mode<br>Reset type: SYSRSn<br>0h (R/W) = IODFT is disabled<br>1h (R/W) = IODFT is disabled<br>2h (R/W) = IODFT is disabled<br>3h (R/W) = IODFT is disabled<br>4h (R/W) = IODFT is disabled<br>5h (R/W) = IODFT is disabled<br>6h (R/W) = IODFT is disabled<br>7h (R/W) = IODFT is disabled<br>8h (R/W) = IODFT is disabled<br>9h (R/W) = IODFT is disabled<br>Ah (R/W) = IODFT is enabled<br>Bh (R/W) = IODFT is disabled<br>Ch (R/W) = IODFT is disabled<br>Dh (R/W) = IODFT is disabled<br>Eh (R/W) = IODFT is disabled<br>Fh (R/W) = IODFT is disabled |
| 7-2 | RESERVED | R | 0h | Reserved |

**Table 28-34. IODFTCTRL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | LPBENA | R/W | 0h | Module loopback enable.<br>Note: In analog loopback mode the complete communication path through the I/Os can be tested, whereas in digital loopback mode the I/O buffers are excluded from this path.<br>Normal and mode reads:<br>Write only in EALLOW mode:<br>Reset type: SYSRSn<br>0h (R/W) = Digital loopback is enabled.<br>1h (R/W) = Analog loopback is enabled in module I/O DFT mode (when IODFTENA = 1010) |
| 0 | RXPENA | R/W | 0h | Module Analog loopback through receive pin enable.<br>This bit defines whether the I/O buffers for the transmit or the receive pin are included in the communication path (in analog loopback mode).<br>Normal and EALLOW mode reads:<br>Write only in EALLOW mode:<br>Reset type: SYSRSn<br>0h (R/W) = Analog loopback through the transmit pin is enabled.<br>1h (R/W) = Analog loopback through the receive pin is enabled. |

#### 28.5.1.1.27 LIN_GLB_INT_EN Register (Offset = 70h) [reset = 0h]

LIN_GLB_INT_EN is shown in Figure 28-48 and described in Table 28-35.

Return to Summary Table.

LIN Global Interrupt Enable Register

**Figure 28-48. LIN_GLB_INT_EN Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | GLBINT1_EN | GLBINT0_EN |
| R-0h | | | | | | R/W-0h | R/W-0h |

**Table 28-35. LIN_GLB_INT_EN Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-2 | RESERVED | R | 0h | Reserved |
| 1 | GLBINT1_EN | R/W | 0h | Global Interrupt Enable for LIN INT1<br>0 = LIN INT1 does not generate interrupt to PIE<br>1= LIN INT1 generates interrupt to PIE if interrupt condition occurs<br>Reset type: SYSRSn |
| 0 | GLBINT0_EN | R/W | 0h | Global Interrupt Enable for LIN INT0<br>0 = LIN INT0does not generate interrupt to PIE<br>1= LIN INT0 generates interrupt to PIE if interrupt condition occurs<br>Reset type: SYSRSn |

#### 28.5.1.1.28 LIN_GLB_INT_FLG Register (Offset = 72h) [reset = 0h]

LIN_GLB_INT_FLG is shown in Figure 28-49 and described in Table 28-36.

Return to Summary Table.

LIN Global Interrupt Flag Register

**Figure 28-49. LIN_GLB_INT_FLG Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | INT1_FLG | INT0_FLG |
| R-0h | | | | | | R-0h | R-0h |

**Table 28-36. LIN_GLB_INT_FLG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-2 | RESERVED | R | 0h | Reserved |
| 1 | INT1_FLG | R | 0h | Global Interrupt Flag for LIN INT1<br>0 = No interrupt generated<br>1= Interrupt is generated due to LIN INT1 (refer to LIN Interrupt Status Register for the condition)<br>Reset type: SYSRSn |
| 0 | INT0_FLG | R | 0h | Global Interrupt Flag for LIN INT0<br>0 = No interrupt generated<br>1= Interrupt is generated due to LIN INT0 (refer to LIN Interrupt Status Register for the condition)<br>Reset type: SYSRSn |

#### 28.5.1.1.29 LIN_GLB_INT_CLR Register (Offset = 74h) [reset = 0h]

LIN_GLB_INT_CLR is shown in Figure 28-50 and described in Table 28-37.

Return to Summary Table.

LIN Global Interrupt Clear Register

**Figure 28-50. LIN_GLB_INT_CLR Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | INT1_FLG_CLR | INT0_FLG_CLR |
| R-0h | | | | | | W-0h | W-0h |

**Table 28-37. LIN_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 LIN INT1<br>0 = No effect<br>1= Write '1' to clear the corresponding bit of the Global Interrupt Flag Register<br>Reset type: SYSRSn |
| 0 | INT0_FLG_CLR | W | 0h | Global Interrupt flag clear for LIN INT0<br>0 = No effect<br>1= Write '1' to clear the corresponding bit of the Global Interrupt Flag Register<br>Reset type: SYSRSn |

# Fast Serial Interface (FSI)

This chapter contains a general description of the Fast Serial Interface (FSI) module. The FSI is a serial peripheral capable of reliable high-speed communication across isolation barriers.

**Topic** **Page**

## 29.1 Introduction to the FSI Module

The Fast Serial Interface (FSI) module is a serial communication peripheral capable of reliable high-speed communication across isolation devices. Galvanic isolation devices are used in situations where two different electronic circuits, which do not have common power and ground connections, must exchange information. Though isolation devices facilitate these signal communications, they can also introduce a large delay on the signal lines and add skew between the signals. The FSI is designed specifically to ensure reliable high-speed communication for system scenarios that involve communication across isolation barriers without adding components.

The FSI consists of independent transmitter (FSITX) and receiver (FSIRX) cores. The FSITX and FSIRX cores are configured and operated independently.

### *29.1.1 Features*

The FSI module includes the following features:

- Independent transmitter and receiver cores
- Source-synchronous transmission
- Dual data rate (DDR)
- One or two data lines
- Programmable data length
- Skew adjustment block to compensate for board and system delay mismatches
- Frame error detection
- Programmable frame tagging for message filtering
- Hardware ping to detect line breaks during communication (ping watchdog)
- Two interrupts per FSI core
- Externally triggered frame generation
- Hardware- or software-calculated CRC
- Embedded ECC computation module
- Register write protection
- DMA support
- CLA task triggering
- SPI compatibility mode (limited features available)

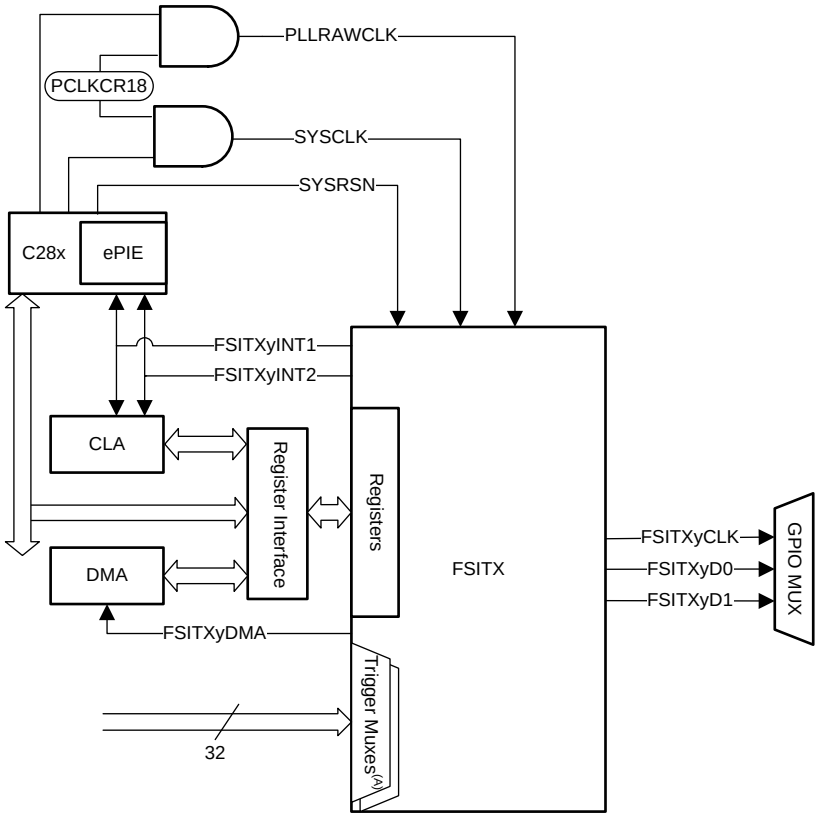
## 29.2 System-level Integration

This section describes the device-level integration of the FSI module. Some of the features may require additional configuration of modules that are not within the scope of this chapter; the details of which can be found elsewhere in this document.

### *29.2.1 CPU Interface*

The following diagrams show the CPU interface of each FSI module.

**Figure 29-1. FSI Transmitter (FSITX) CPU Interface**




A The signals connected to the trigger muxes are described in Section 29.2.6.

**Figure 29-2. FSI Receiver (FSIRX) CPU Interface**

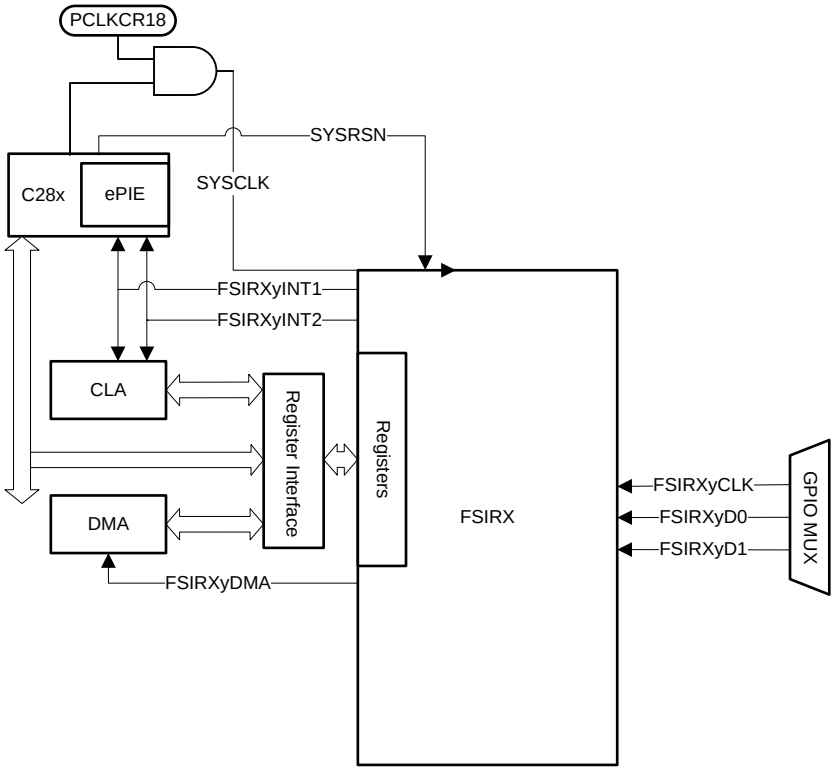
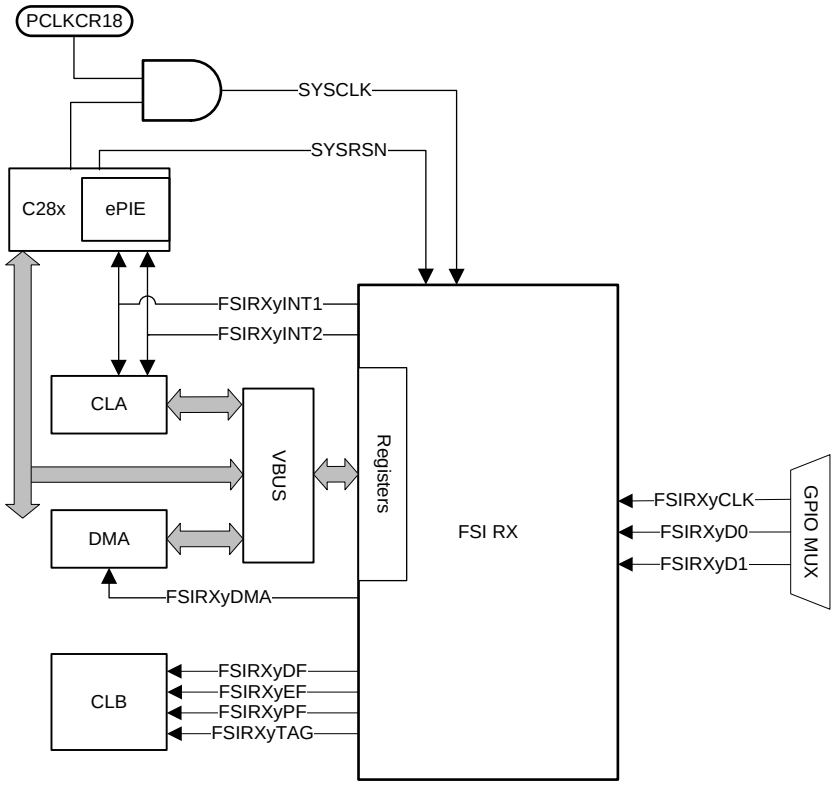


**Figure 29-3. FSI Receiver (FSIRX) CPU Interface with CLB**




### 29.2.2 Signal Description

FSI is a point-to-point communication protocol. Hence, an FSI transmitter core will communicate directly to a single FSI receiver core. Similarly, an FSI receiver core will receive data from a single FSI transmitter core.

Each FSI core has three signals associated with it: one clock and two data signals. Data is always transmitted or received with the most significant bit of each frame field being first. If multi-lane transmissions are not used, the TXD1 and RXD1 signals can be left unconnected and their GPIOs repurposed for other application needs. Table 29-1 and Table 29-2 describe the various signals that can be selected by the GPIO mux to be brought out to device pins.

**Table 29-1. FSI Transmitter Core Signals**

| Signal Name | Direction | Description | Inactive Level[(1)] |
|---|---|---|---|
| TXCLK | Output | This is the transmit clock. It is driven by the FSI transmit module.<br>During a transmission, four clock edges will be transmitted before the start of frame phase (premable) and four clock edges will follow the last bit of the frame (postamble). Data is transmitted on both edges of the clock.<br>In SPI compatibility mode, the preamble and the post frame clock edges will not be transmitted. Data is transmitted only on one edge of the clock. The data is launched on the falling edge of the clock with the expectation that the receiving SPI will capture the data on the rising edge. | Logic High |
| TXD0 | Output | This is the primary data output line for transmission. This signal is driven by the FSI transmit module. | Logic High |

(1) Inactive level refers to the state of the pin while the module is not actively transmitting, or held in reset.

**Table 29-1. FSI Transmitter Core Signals (continued)**

| Signal Name | Direction | Description | Inactive Level[1] |
|---|---|---|---|
| | | When the FSI is configured for multi-lane transmission, TXD0 will contain all the even numbered bits of the data and CRC bytes. Other frame fields such as frame type, start-of-frame, tag, and end-of-frame will be transmitted in full. | |
| TXD1 | Output | This is an additional data output line for transmission if the FSI is configured for multi-lane transmission. This signal is driven by the FSI transmit module.<br>During transmission, the data bits are split between TXD0 and TXD1. TXD1 will contain all the odd numbered bits of the data and CRC bytes. This applies only to the data words and the CRC bytes. Other data frame related information like Frame Type, Start-of-Frame, Tag and End-of-frame, the state of this line will be identical to TXD0. | Logic High |

**Table 29-2. FSI Receiver Core Signals**

| Signal Name | Direction | Description | Inactive Level[1] |
|---|---|---|---|
| RXCLK | Input | This is the receive clock input signal for the FSI receive module.<br>This must should be connected to TXCLK of the transmitting FSI module. | Logic High |
| RXD0 | Input | This is the primary data input line for reception. This should be connected to the TXD0 of the transmitting FSI module. | Logic High |
| RXD1 | Input | This is an additional data input line for reception. This signal should be connected to the TXD1 of the transmitting FSI module if multi-lane transmission is used. | Logic High |

(1) Inactive level refers to the state of the pin while the module is not actively receiving data.

#### 29.2.2.1 Configuring Device Pins

The GPIO mux registers must be configured to connect this peripheral to the device pins. To avoid glitches on the pins, the GPyGMUX bits must be configured first (while keeping the corresponding GPyMUX bits at the default of zero), followed by writing the GPyMUX register to the desired value.

Some IO functionality is defined by GPIO register settings independent of this peripheral. For input signals, the GPIO input qualification should be set to asynchronous mode by setting the appropriate GPxQSELn register bits to 0x3. The internal pullups can be configured in the GPyPUD register. See *General Purpose Input-Output*, Chapter 8, for more details on the GPIO mux and settings.

### *29.2.3 Interrupts*

Each FSI module contains multiple interrupt sources which can be assigned to two different interrupt vectors: INT1 and INT2. Each interrupt source has an associated status flag, force, and clear bits in the EVT_STS, EVT_FRC, and the EVT_CLR registers, respectively.

**NOTE:** Because the transmitter and receiver cores have their own distinct register sets, the preceding 'TX_' and 'RX_' will be left off of the register names unless otherwise noted.

Each interrupt can be assigned to either interrupt vector, INT1 and INT2, to allow for two priority levels. Alternately, the interrupt source can be prevented from generating any interrupt, though the status flag can still be set and monitored by software. The transmitter events are assigned to either interrupt vector in the TX_INT_CTRL register. The receiver events are assigned an interrupt vector using RX_INT1_CTRL and RX_INT2_CTRL registers. If an interrupt is not required, ensure the bit is not set in the respective INT_CTRL register.

#### 29.2.3.1 Transmitter Interrupts

The transmitter can generate the following interrupts:

- **Frame Done (FRAME_DONE)**
  This event indicates that has completed transmitting a frame.
- **Buffer Underrun (BUF_UNDERRUN)**
  This event indicates that the transmit buffer has experienced underrun. Buffer underrun occurs when the transmitter tries to read data from a location which has not yet be written to by the CLA, CPU, or DMA.
- **Buffer Overrun (BUF_OVERRUN)**
  The buffer overrun interrupt is generated when the buffer has experienced overrun. Buffer overrun may occur if a piece of data is overwritten before it has been transmitted.
- **Ping Frame Triggered (PING_TRIGGERED)**
  The ping frame triggered interrupt is generated when the ping frame has been triggered. This bit will be set when the ping counter has timed out or an external ping trigger event has occurred.

#### 29.2.3.2 Receiver Interrupts

The receiver core is capable of generating interrupts from many different events. These events are described below.

- **Ping Watchdog Timeout (PING_WD_TO)**
  This event indicates that the ping watchdog timer has timed out. The receiver has not received a valid frame within the time period specified in the RX_PING_WD_REF register.
- **Frame Watchdog Timeout (FRAME_WD_TO)**
  This event indicates that the frame watchdog timer has timed out. The conditions of this timeout are set using the RX_FRAME_WD_CTRL register. As soon as the start of frame phase is detected, the frame watchdog counter will start counting from 0. The end of frame phase must complete by the time the watchdog counter reaches the reference value. If this does not happen, the watchdog will time out and this event will be generated. If this event occurs, the receiver must undergo a soft reset and subsequent resynchronization in order to guarantee proper operation.
- **CRC Error (CRC_ERR)**
  This error indicates that a CRC error has occurred. A CRC error will be generated when the received CRC and the computed CRC do not match.
- **Frame Type Error (TYPE_ERR)**
  This error indicates that an invalid frame type has been received. If this error occurs, the receiver must undergo a soft reset and subsequent resynchronization in order to guarantee proper operation.
- **End-of-Frame Error (EOF_ERR)**
  This error indicates that an invalid end-of-frame bit pattern has been received. If this error occurs, the receiver must undergo a soft reset and subsequent resynchronization in order to guarantee proper operation.
- **Receive Buffer Overrun (BUF_OVERRUN)**
  This event indicates that an overrun condition has occurred in the receive buffer.
- **Receive Buffer Underrun (BUF_UNDERRUN)**
  This event indicates that an underrun condition has occurred in the receive buffer. This condition occurs when software reads the buffer while it is empty.
- **Frame Done (FRAME_DONE)**
  This event indicates that a valid frame has been received without error.
- **Error Frame Received (ERR_FRAME)**
  This event indicates that an error frame has been received.
- **Ping Frame Received (PING_FRAME)**
  This event indicates that a ping frame has been received.
- **Frame Overrun (FRAME_OVERRUN)**
  This event indicates that a new frame has been received while the FRAME_DONE flag was still set.
- **Data Frame Received (DATA_FRAME)**
  This event indicates that a data frame has been received.

#### 29.2.3.3 Configuring Interrupts

To configure interrupts on the FSI the application should select the interrupt vector for each desired event using the TX_INT_CTRL register for the transmitter, and RX_INT1_CTRL and RX_INT2_CTRL registers for the receiver. There is no module-level interrupt enable bit to configure.

**NOTE:** If an event is registered for both interrupt vectors, both interrupts will fire. There are no hardware checks for overlapping interrupt vector assignments.

#### 29.2.3.4 Handling Interrupts

Inside the interrupt service routine (ISR), the user should clear the event flag using the EVT_CLR register and then acknowledge the CPU interrupt.

If the one event occurs multiple times before the corresponding bit is cleared by software, no new interrupt will be generated.

If multiple events occur simultaneously, or very close in time, it is possible to handle multiple conditions within a single interrupt. Each flag is independently set by hardware and must be cleared by application software. If multiple different events occur, the ISR can handle each in whatever order is deemed necessary by the application. It is not advisable to clear the full interrupt status register in every ISR. This may cause the application to miss events that may be detrimental to the application. A sample sequence for handling interrupts on the receiver follows, the transmitter routine will be similar..

- On receiving an interrupt, copy the current state of the receive event and error status flag register (RX_EVT_STS) into a local snapshot variable.
- Read all of the bits from the snapshot to determine the events which require action.
- Perform the necessary actions for each fo the events seen in the snapshot.
- Write to the receive event and error clear register (RX_EVT_CLR) with the snapshot to clear only those interrupts that were set at the beginning of the ISR.
- Repeat this sequence for every generated ISR.

There is a chance that another event occurred during the just-handled ISR since only the snapshot of events was handled and then cleared; an event flag may still be set at the end of the ISR. As soon as the ISR completes, a new interrupt will be generated and this flag will still be set and can be handled accordingly.

Software accesses tied to multiple events and handled within the same ISR may cause race conditions which will cause the software to not function as desired. For example, it is recommended to use different interrupt lines if the user wants to enable events for both ping and data frames. If both are handled within the same interrupt line, the software may only respond to one of the events if they both occur close in time.

### *29.2.4 CLA Task Triggering*

In addition to generating interrupt vectors to the PIE, both interrupts lines for each module TX_INT1, TX_INT2, RX_INT1, and RX_INT2 can be assigned to trigger CLA tasks. Refer to Table 5-1 for the list of all sources capable of CLA task triggering. The configuration and use of CLA tasks are described in the Section 5.2.4. The CLA has access to the entire FSI register map. This allows the CLA to manage the FSI independently from the CPU, freeing it up for other tasks.

### *29.2.5 DMA Interface*

Both the transmitter and receiver are capable of using the DMA for automatic data transfers. The DMA trigger is independent from the interrupt signals. DMA events are only triggered on the completion of a data frame.

The transmitter DMA trigger is enabled by setting TX_DMA_CTRL.DMA_EVT_EN to 1. The transmitter must also set TX_OPER_CTRL_LO.START_MODE to 0x2 to allow either a write to the TX_FRAME_CTRL.START bit or to the TX_FRAME_TAG_UDATA register to start the transmission.

The receiver DMA trigger is enabled by setting RX_DMA_CTRL.DMA_EVT_EN to 1.

Refer to Section 29.3.2 and Section 29.3.3 for more DMA information specific to each FSI Module.

### 29.2.6 External Frame Trigger Mux

The FSI has two muxes connected to the transmitter module. These muxes are used to select triggers to start ping frames, and/or generic frames. These muxes are independently configured for each type of frame. The application may select one trigger source per frame type. Use of these triggers are optional.

The external ping frame trigger is configured by setting TX_PING_CTRL.EXT_TRIG_SEL to the index of the desired trigger. TX_PING_CTRL.EXT_TRIG_EN must also be set to allow the trigger to generate a ping frame.

The generic frame trigger is configured by setting TX_OPER_CTRL_HI.EXT_TRIG_SEL to the index of the desired trigger. TX_OPER_CTRL_LO.START_MODE must be set to 0x1 in order for a frame to be transmitted by an external trigger.

**Table 29-3. External Trigger Sources and Their Index**

| Index | External Trigger Source |
|---|---|
| 0:7 | Reserved |
| 8 | EPWM1-SOCA |
| 9 | EPWM1-SOCB |
| 10 | EPWM2-SOCA |
| 11 | EPWM2-SOCB |
| 12 | EPWM3-SOCA |
| 13 | EPWM3-SOCB |
| 14 | EPWM4-SOCA |
| 15 | EPWM4-SOCB |
| 16 | EPWM5-SOCA |
| 17 | EPWM5-SOCB |
| 18 | EPWM6-SOCA |
| 19 | EPWM6-SOCB |
| 20 | EPWM7-SOCA |
| 21 | EPWM7-SOCB |
| 22 | EPWM8-SOCA |
| 23 | EPWM8-SOCB |
| 31:24 | Reserved |

## 29.3 FSI Operation

### 29.3.1 Introduction to Operation

The transmitter and receiver modules are two completely independent modules on the device. Each module has an independent set of control registers, clocking, and interrupts. The following sections describe the frame format and the various initialization and configuration procedures for both the transmitter and receiver.

### 29.3.2 FSI Transmitter Module

The FSI transmitter module handles the framing of data, CRC generation, and signal generation of TXCLK, TXD0, and TXD1, as well as interrupt generation. The operation of the transmitter core is controlled and configured through programmable control registers. The transmitter control registers allow the CPU (or the CLA) to program, control, and monitor the operation of the FSI receiver. The transmit data buffer is accessible by the CPU, CLA, and the DMA.

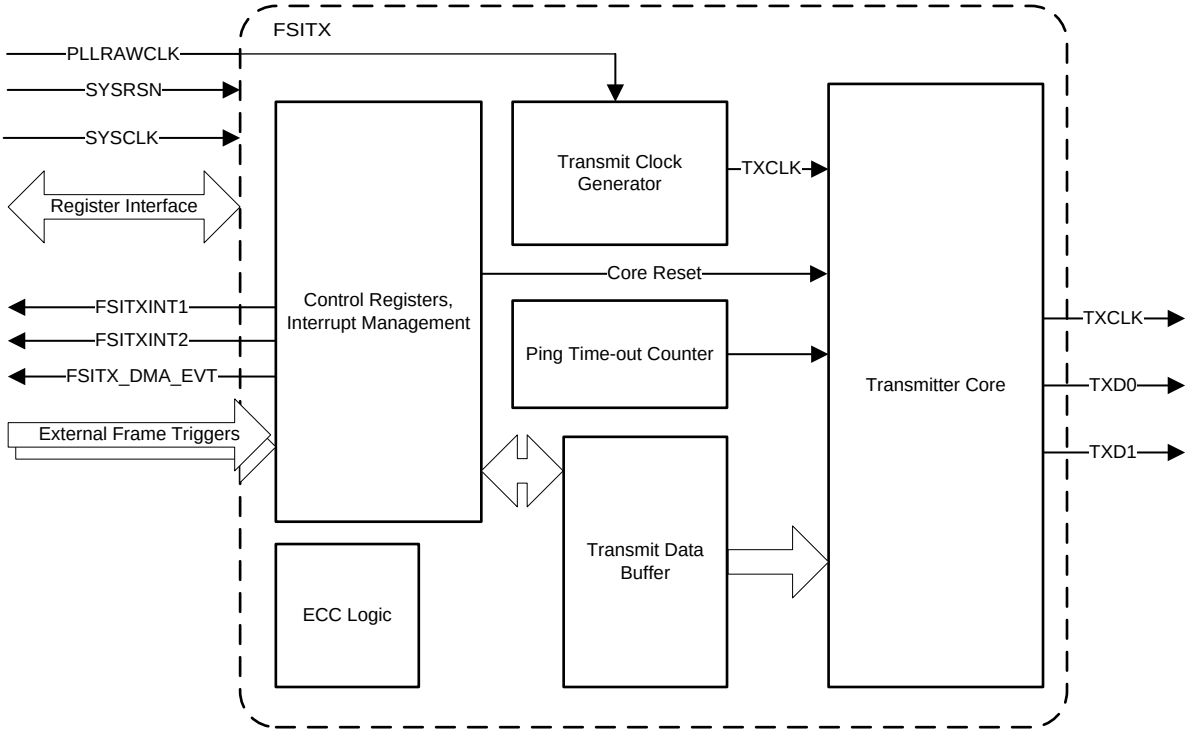
The transmitter has the following features:

- Automated ping frame generation

- Externally triggered ping frames
- Externally triggered data frames
- Software-configurable frame lengths
- 16-word data buffer
- Data buffer underrun and overrun detection
- Hardware-generated CRC on data bits
- Software ECC calculation on select data
- DMA support
- CLA task triggering

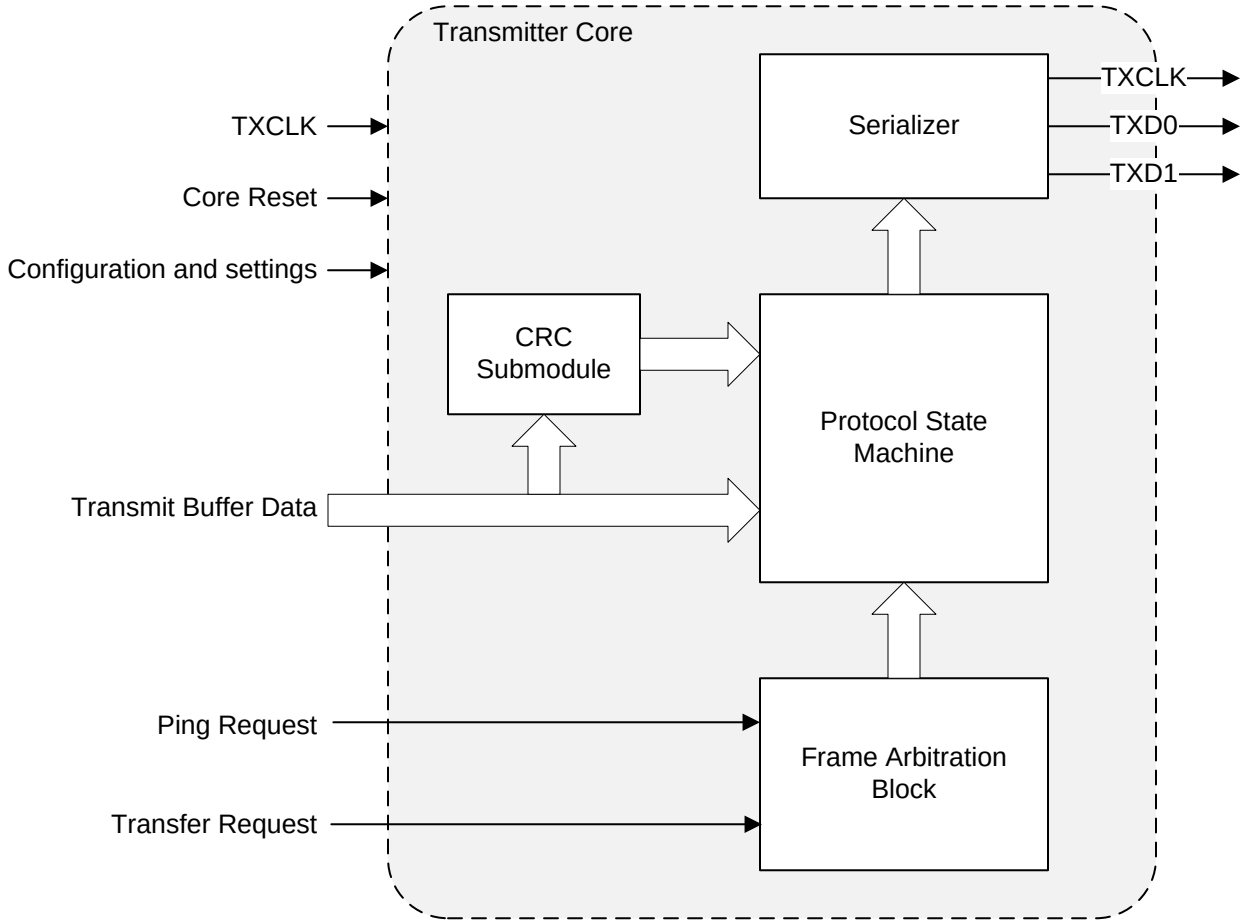
Figure 29-4 shows the high-level block diagram of the FSI transmitter. Figure 29-5 shows the block diagram of the transmitter core submodule.

**Figure 29-4. FSI Transmitter Block Diagram**




The block diagram of the transmitter core submodule is shown in Figure 29-5.

**Figure 29-5. FSI Transmitter Core Block Diagram**




The following sections describe the various aspects of the FSI transmitter in detail.

#### 29.3.2.1 Initialization

On the first initialization or after a module reset due to an underrun condition, the transmitter module should execute the following initialization sequence in order to start or resume transmit operations.

1. Initialize the transmitter clock by setting TX_CLK_CTRL.CLK_RST to 1 and subsequently clearing it.
2. Set the clock to the transmitter core to PLLRAWCLK by setting TX_OPER_CTRL_LO.SEL_PLLCLK to 1.
3. Set the clock prescaler value to the desired rate by writing to TX_CLK_CTRL.PRESCALE_VAL.
4. Enable the transmitter clock divider by setting TX_CLK_CTRL.CLK_EN to 1.
5. Assert the transmitter module soft reset by writing 0xA501 to TX_MASTER_CTRL.
6. Wait four TXCLK cycles.
7. Release the transmitter core from reset by writing 0xA500 to TX_MASTER_CTRL

After initialization and configuration, the transmitter module should synchronize with receiver module before transmitting. The synchronization sequence is described in Section 29.4.1.

**CAUTION**

Do not change TX_CLK_CTRL.PRESCALE_VAL while the clock is enabled (TX_CLK_CTRL.CLK_EN = 1). Doing so may cause undefined behavior.

#### 29.3.2.2 Clocking

The transmitter core registers and control logic run off of the device system clock (SYSCLK).

The FSI Transmit Clock (TXCLK) is derived from PLLRAWCLK. PLLRAWCLK is divided down by configuring the clock prescaler value (TX_CLK_CTRL.PRESCALE_VAL) then setting the clock divider enable bit (TX_CLK_CTRL.CLK_EN). The clock prescaler value can be set to divide PLLRAWCLK by 1 (TX_CLK_CTRL.PRESCALE_VAL = 0x0 or 0x1) through 255(TX_CLK_CTRL.PRESCAL_VAL = 0xFF). Though TXCLK and SYSCLK are both derived from PLLRAWCLK, TXCLK is asynchronous with respect to SYSCLK.

**CAUTION**

TXCLK should never be configured to be faster than SYSCLK.

#### 29.3.2.3 Transmitting Frames

On the transmitter, the ping frame is the only frame which can be set up and transmitted without any further software or DMA intervention. Ping frames can be transmitted by any (or all) of the three sources: automatic ping timer, software, or external triggers.

Each available frame type can be sent multiple ways. Generically, the following steps must be executed before the frame is sent. These steps may be executed in any order before the start condition is set.

1. Configure the frame type
2. Set the frame tag
3. If the frame to be sent is a data frame:
   1. Set the user data
   2. Write to the data buffer
   3. Set the word length if the frame is a software defined frame length
4. Set the start condition

**NOTE:** Note: There is no hardware check implemented to check whether the type field written by software is valid or not. If an invalid type is used and a frame transmission is initiated, the behavior will be as follows:

- The transmitted frame structure will be exactly like an NWORD data frame. The size of the data frame will be determined by the value in the TX_FRAME_CTRL.N_WORDS register.
- The frame type field of the transmitted data frame will be transmitted as programmed. If this is received by an FSI receiver, it will generate a Type error.

This mechanism can be used for force a Type error in a received frame for testing purposes.

The following sections describe the specific configuration for each frame type and start condition.

##### *29.3.2.3.1 Software Triggered Frames*

The most basic way to transmit a data frame is through software. Each step must be handled by the application. To send a data frame using software, the following steps should be executed. Steps 1-6 may be executed in any order before setting TX_FRAME_CTRL.START. Some fields may not need to be reconfigured for every transmission. The frame tag, user data, and frame type are sticky and will be re-transmitted in the subsequent frame unless they are modified by software.

1. Write the data to be transmitted to the next location of the transmit data buffer.
2. Set TX_FRAME_CTRL.FRAME_TYPE to the appropriate value for the type of frame to be transmitted.
3. Set TX_FRAME_CTRL.N_WORDS to 1 less than the number of words to be transmitted if TX_FRAME_CTRL.FRAME_TYPE is set to 0011, the frame type of the software-defined length data frame. That is, if 16 words will be transmitted, N = 16, set TX_FRAME_CTRL.N_WORDS to 15.
4. When the frame is assembled before transmitting, the FSITX hardware will calculate the CRC to be

transmitted. If TX_OPER_CTRL_LO.SW_CRC is 1, the application may calculate a custom CRC value and then set TX_USER_CRC to the result.

5. Set TX_FRAME_TAG_UDATA.FRAME_TAG to the desired tag.
6. Set TX_FRAME_TAG_UDATA.USER_DATA to the desired user data.
7. Set TX_FRAME_CTRL.START to 1 to initiate the transmission of the data frame.

Once the frame transmission has started, the TX_FRAME_CTRL.START will be cleared by hardware. To monitor if the frame has completed, the software can poll TX_EVT_STS.FRAME_DONE.

#### *29.3.2.3.2 Externally Triggered Frames*

The transmitter can transmit frames when triggered by an external source. See Section 29.2.6 for more information on the available external triggers.

To transmit frames using an external trigger, the application must follow the same procedure as described in Section 29.3.2.3.1. The only difference is that in Step 7, the start condition will be automatically set when the external trigger condition is met rather than by software.

It is important to note that by externally triggering frames, the frame information to be sent will be pulled from the same registers described in the previous section. Because of this, it is possible to send any type of frame from an external trigger including ping, error, and data frames. Also, there is no hardware mechanism by which the FSI can determine if multiple triggers occur. The FSITX will take the data as is, and the application software should ensure that this data has been updated as necessary.

Using TX_EVT_STS fields either by polling or by interrupts, the application can populate or update the frame information to be sent in the next frame

#### *29.3.2.3.3 Ping Frame Generation*

Assuming the FSI transmitter has already been properly initialized, the following sequences can be used to configure and send ping frames.

##### *29.3.2.3.3.1 Automatic Ping Frames*

To generate periodic ping frames, the following steps must be followed:

1. Initialize the ping counter by writing 1 to TX_PING_CTRL.CNT_RST.
2. Set the desired ping tag to TX_PING_TAG.TAG.
3. Set the ping timer reference value to TX_PING_TO_REF.TO_REF.
4. Enable the ping timer by writing 1 to TX_PING_CTRL.TIMER_EN.

The ping timer is a free-running counter which will count up from 0. The current value of the ping timer counter is found in TX_PING_TO_CNT. When the current value of TX_PING_TO_CNT matches the reference value TX_PING_TO_REF.TO_REF, TX_EVT_STS.PING_TRIGGERED will be set. TX_PING_TO_CNT will reset to 0 and resume counting until the next match has occurred or the ping timer is halted by software (TX_PING_CTRL.TIMER_EN is set to 0).

##### *29.3.2.3.3.2 Software Triggered Ping Frame*

Software can also manually generate a ping frame. The process for sending a ping frame with software is very similar to sending the other types of frames. The following steps must be followed:

1. Set TX_FRAME_CTRL.FRAME_TYPE to 0000’b to denote that the frame being sent will be a Ping Frame.
2. Set TX_FRAME_TAG_UDATA.FRAME_TAG to the desired value.
3. Write 1 to TX_FRAME_CTRL.START. This will start the transmission.

Once the frame transmission has started, the TX_FRAME_CTRL.START will be cleared by hardware. To monitor if the frame has completed, the software can poll TX_EVT_STS.FRAME_DONE.

#### 29.3.2.3.3.3 Externally Triggered Ping Frame

The last source for generating ping frames is an external trigger. One of up to 32 different triggers may be selected. See Section 29.2.6 for the list of input sources.

**CAUTION**

Ping frames can be triggered by both an external trigger source and the internal ping timer. If TX_PING_CTRL.EXT_TRIG_EN is set to 1, it will take precedence and the ping timer will be ignored.

#### 29.3.2.3.4 Transmitting Frames with DMA

The FSI transmitter can send data which is continuously fed with the DMA. A DMA trigger will be generated every time a data frame transmission is completed. This is concurrent with the FRAME_DONE signal that sets the TX_EVT_STS.FRAME_DONE flag.

In order to transmit continuous data with the DMA, some configurations need to be made on the transmitter:

First, set TX_DMA_CTRL.DMA_EVT_EN to 1. This will allow the DMA trigger to propagate to the DMA module. Next, TX_OPER_CTRL_LO.START_MODE must be set to 0x2. The transmitter is now able to start a transmission using a software write to TX_FRAME_CTRL.START or TX_FRAME_TAG_UDATA..

The DMA must also be configured properly for the FSI to send the data. One possible solution of using the DMA to continuously feed the transmit buffer is shown below:

- Set up two DMA channels to be triggered by the same FSI transmitter and DMA trigger.
- Configure one channel to fill the transmit buffer.
- Configure the other channel to set the frame tag and user data fields
- Since the FSI transmit buffer is a 16-word circular buffer, ensure the DMA channel servicing the data buffer wraps the after 16 words are copied.

**NOTE:** Because the frame tag and user data must written in to in order to initiate the transmission of the frame, use two consecutive DMA channels. This ensures that the DMA channels are always executed in sequence. The DMA channel servicing the data buffer should be the lower numbered channel and the tag/user data channel should be the next. For example, configure DMA channel 3 to service the data buffer, and configure DMA channel 4 to service the tag and user data.

### 29.3.2.4 Transmit Buffer Management

The FSI transmitter has a 16-word buffer from which it pulls data to transmit. This buffer is implemented as a circular buffer, not a FIFO, so some care must be taken to properly interpret buffer overrun and underrun, as well as the TX_BUF_PTR_STS register. These flags and pointers work under the assumption that the software or DMA is using the buffer as a circular buffer. This mode of operation is the only way in which the overrun, underrun, and pointer status are meaningful. If data is being sourced by the DMA and there is some other periodic trigger mechanism trying to initiate transfers, underrun becomes a critical error. If an underrun happens, it means that the buffer went out of sync. This would not only affect the current transfer, but all future transfers also could not be guaranteed due to the ring buffer. Under such conditions, the underrun would need a soft reset to cleanly recover. Alternately, the software could manually stop the transmitting, reset the buffer pointers, clear the remaining error conditions, and then restart transmission. The software method involves a few steps, while the soft reset is a single action and will guarantee a full reset of the control registers.

Due to the flexibility of the transmit buffer, it is possible for software to implement a simple ping-pong buffer, or to randomly load and send from any location of the buffer. If the buffer is used in this manner, error flags and status fields may be ignored without adversely affecting the transmitter capability. Additionally, the CURR_WORD_CNT will also be invalid if used in this way. The application can set the buffer pointer manually by writing the 4-bit index to TX_BUF_PTR_LOAD. This will force the transmitter to start picking the data from the indicated location in the buffer.

#### 29.3.2.5 CRC Submodule

The FSI transmitter can supply the CRC to the frame being transmitted through the embedded hardware CRC submodule or by supplying a user-defined value. This is controlled by setting TX_OPER_CTRL_LO.SW_CRC appropriately.

If hardware CRC generation is selected (TX_OPER_CTRL_LO.SW_CRC = 0, the default), the CRC is computed by hardware on the data and user data fields using the CRC polynomial 0x7 ($x^8 + x^2 + x + 1$). The transmitter module will automatically compute the CRC on the data fields without user intervention when the frame is transmitted. For more information on how the CRC is generated by the CRC Submodule, refer to Section 29.3.7.

If software CRC generation is selected (TX_OPER_CTRL_LO.SW_CRC = 1), the CRC must be computed by software and placed in the TX_USER_CRC register. The next frame to be transmitted will use the value placed in the TX_USER_CRC register in place of the CRC value generated by the hardware.

As the TX_USER_CRC register is software-programmable, the application may use this field as an extra data field for application specific purposes. If TX_USER_CRC is used in this manner, the CRC detection on the receiver will not be valid and should be ignored.

#### 29.3.2.6 Conditions in Which the Transmitter Must Undergo a Soft Reset

Unlike the receiver, there are no detectable errors that require a soft reset. A buffer overrun or underrun interrupt may or may not require a soft reset in order to resume proper operation. This determination is up to the application software. Refer to Section 29.3.2.4 for more information on the transmit buffer.

#### 29.3.2.7 Reset

The entire transmitter module and all transmitter registers are reset by SYSRSn. The transmitter core is reset by SYSRSn or by writing a 1 to TX_MASTER_CTRL.CORE_RST.

A module reset will cause the registers to be reset to their default state.

### *29.3.3 FSI Receiver Module*

The receiver module interfaces to the FSI clock (RXCLK), and data lines (RXD0 and RXD1) after they pass through an optional programmable delay line. The receiver core handles the data framing, CRC computation, and frame-related error checking. The receiver bit clock and state machine are run by the RXCLK input, which is asynchronous to the device system clock.

The receiver control registers allow the CPU (or the CLA) to program, control, and monitor the operation of the FSI receiver. The receive data buffer is accessible by the CPU, CLA, and the DMA.

The receiver core has the following features:

- 16-word data buffer
- Multiple supported frame types
- Ping frame watchdog
- Frame watchdog
- CRC calculation and comparison in hardware
- ECC detection
- Programmable delay line control on incoming signals
- DMA support
- CLA task triggering

- SPI compatibility mode

Figure 29-6 provides a high-level overview of the internal modules present in the FSI receiver. Figure 29-7 shows a view of the FSI receiver core submodule. Not all data paths and internal connections are shown.

**Figure 29-6. FSI Receiver Block Diagram**

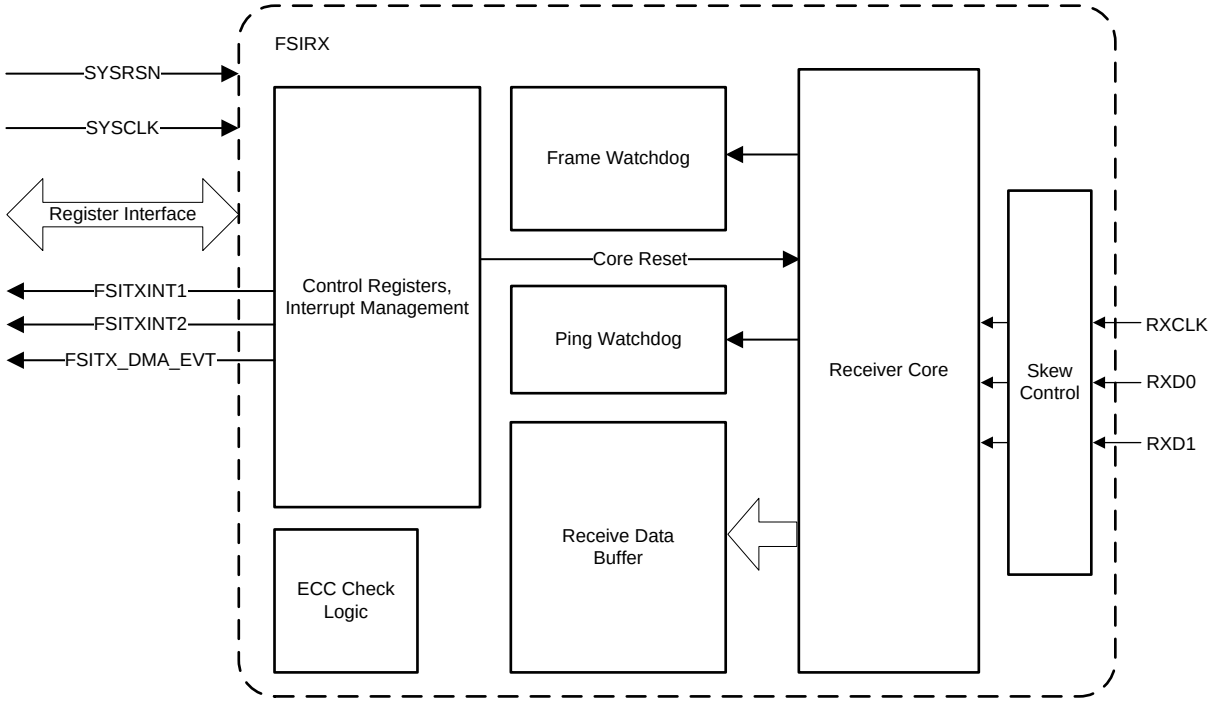
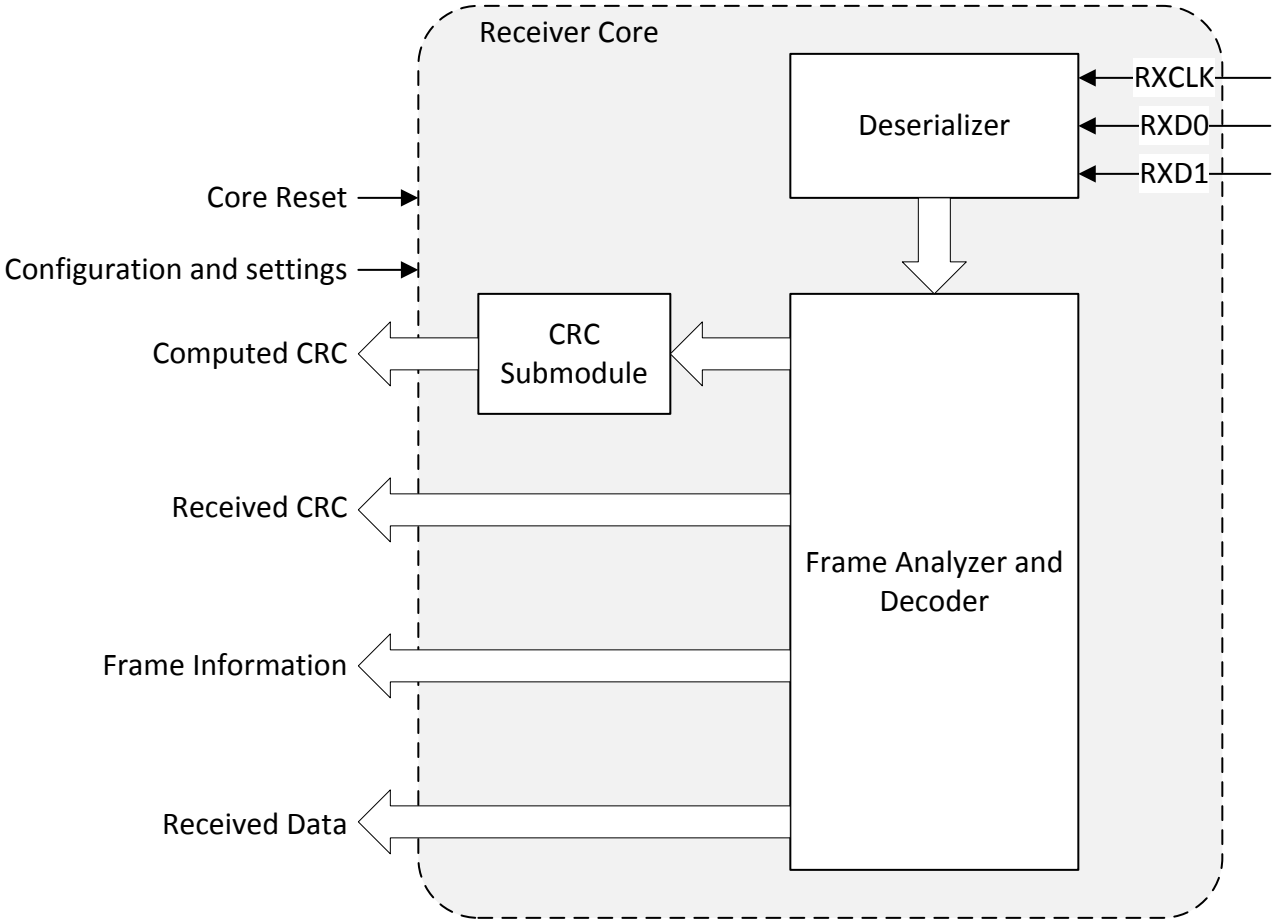


**Figure 29-7. FSI Receiver Core Block Diagram**




The following sections describe the various aspects of the FSI receiver module

#### 29.3.3.1 Initialization

On the first initialization or after a module reset following any frame error, the receiver module should assert and release the receiver core reset bit (RX_MASTER_CTRL.CORE_RST) prior to any other initialization. Once the receiver module is initialized, the following steps may be executed.

1. If required, assign interrupt sources to the necessary interrupt line.
2. If required, configure the ping watchdog to periodically check for an active link to the transmitter. See Section 29.3.3.4 for configuration details.
3. If required, configure the frame watchdog to ensure that each frame is received within a predetermined window. See Section 29.3.3.5 for configuration details.
4. Initialize the receive buffer pointer by writing to the RX_BUF_PTR_CTRL register. Received data will be placed into the buffer starting with the address loaded in this register.
5. Ensure all errors and flags have been cleared from the RX_EVT_STS register.

At this point the receiver is ready to receive any incoming frames. Software can now either poll on the RX_EVT_STS register for various conditions. For example, when the RX_EVT_STS.FRAME_DONE and no other flags are set, the receiver has successfully received a frame without error.

Next, the application should configure the various features such as the ping and frame watchdogs, DMA, external triggering, and so on. These features are described in subsequent sections. The receiver module is now ready to synchronize with the transmitter then begin reception. The synchronization sequence is described in Section 29.4.1.

#### 29.3.3.2 Clocking

The receiver module registers and control logic are clocked by the device system clock (SYSCLK). The receiver state machine is clocked by the receiver input clock pin (RXCLK).

**CAUTION**

RXCLK should never be faster than SYSCLK.

#### 29.3.3.3 Receiving Frames

Once the receiver has been properly configured and synchronized, incoming messages are handled as described below. It is important to note that there is no equivalent to a chip select signal to gate incoming data. Every valid clock edge will latch data into the receiver.

The header information of the received frame will be placed in their respective register fields.

- RX_FRAME_INFO.FRAME_TYPE will contain the received frame type.
- RX_FRAME_TAG_UDATA.FRAME_TAG will contain the received frame tag.
- RX_FRAME_TAG_UDATA.USER_DATA will contain the received user data.

If any error conditions occur during reception such as a CRC mismatch, frame error, frame timeout, buffer overrun, or ping watchdog timeout, the corresponding flag will be set in the RX_EVT_STS register.

**NOTE:** If at any point during operation a frame error occurs, the receiver module must be reset and re-synchronized with the transmitter before the next frame can be successfully received. The follow errors are classified as frame errors:

- Type error
- CRC error
- End of frame error

##### *29.3.3.3.1 Receiving Frames with DMA*

The FSI receiver can continuously receive data and move it from the receiver buffer with the DMA. A DMA trigger will be generated every time a data frame has been received. This is concurrent with the FRAME_DONE signal that sets the RX_EVT_STS.FRAME_DONE flag. In order to receive continuous data with the DMA, some configurations need to be made on the receiver.

First, set RX_DMA_CTRL.DMA_EVT_EN to 1. This will allow the DMA trigger to propagate to the DMA module. The receiver is now able to trigger a DMA event upon the reception of a data frame.

The DMA must also be configured properly for the FSI to receive the data. One possible solution for using the receiver to continuously feed the DMA is show below:

- Set up two DMA channels to be triggered by the FSI Receiver DMA Trigger.
- One DMA channel should be configured to copy data from the receive buffer to a larger data buffer.
- The other DMA channel should be configured to copy the received frame tag and user data to another data buffer.
- Since the FSI receive buffer is a 16-word circular buffer, ensure the DMA channel servicing the data buffer wraps after 16 words are copied.

Unlike the transmitter, there is no requirement to have the DMA channel which is handling the data buffer execute before the DMA channel handling the received tag and user data.

#### 29.3.3.4 Ping Frame Watchdog

The ping frame watchdog is a hardware-enabled automatic error detection of the connection status to the transmitter. This watchdog monitors the time elapsed between ping frames. If the transmitter has been set up to periodically send out a ping frame, the receiver can be set up to monitor whether this frame has been received within a specified amount of time. If the time between ping frames has exceeded the programmed number of clock cycles, an event will be triggered which can generate an interrupt or be monitored by software.

This watchdog has a dedicated counter which is reset and restarted upon the successful reception of a ping frame. The watchdog counter will be incremented at the rate of SYSCLK. Optionally, the watchdog can be configured to be reset upon the successful reception of any frame. This option allows the receiver to monitor for any successful frame to indicate that the connection is still alive and the transmitter is still functioning as expected.

To configure the ping frame watchdog for operation:

1. Reset the ping watchdog counter by setting RX_PING_WD_CTRL.PING_WD_RST to 1 and then subsequently clearing it to 0.
2. Set RX_OPER_CTRL.PING_WD_RST_MODE to the desired watchdog reset event, either 0 for ping frames only, or 1 for any frame.
3. Set RX_PING_WD_REF to the maximum time between frames. Add 10 additional SYSCLK cycles to account for clock synchronization.
4. Enable the ping watchdog by setting RX_PING_WD_CTRL.PING_EN to 1.

The ping watchdog is now enabled and can now monitor for ping frames.

If the RX_PING_WD_CNT value reaches the value programmed in RX_PING_WD_REF, the RX_EVT_STS.PING_WD_TO flag will be set. If configured, and interrupt can be generated on this event.

#### 29.3.3.5 Frame Watchdog

The frame watchdog is an additional feature the receiver can use to monitor for any error conditions. This dedicated watchdog monitors the duration that it takes for a single frame to be received. The watchdog starts incrementing at the time the receiver detects a proper start of frame condition. If the end of frame condition is not detected within the expected number of SYSCLK cycles, the frame watchdog will be triggered which can generate an interrupt or be monitored by software.

This watchdog is automatically started and stopped at the start-of-frame and end-of-frame conditions respectively. The frame watchdog is connected to SYSCLK.

To configure the frame watchdog for operation:

1. Reset the frame watchdog counter by setting RX_FRAME_WD_CTRL. FRAME_WD_CNT_RST to 1 and then subsequently clearing it to 0.
2. Set RX_FRAME_WD_REF.FRAME_WD_REF to the maximum number of SYSCLK cycles expected to be in the longest frame that may be received. Add an additional 10 SYSCLK cycles to account for clock synchronization.
3. Enable the frame watchdog by setting RX_FRAME_WD_CTRL.FRAME_WD_CNT_EN to 1.

The frame watchdog is now enabled and can detect a failed frame.

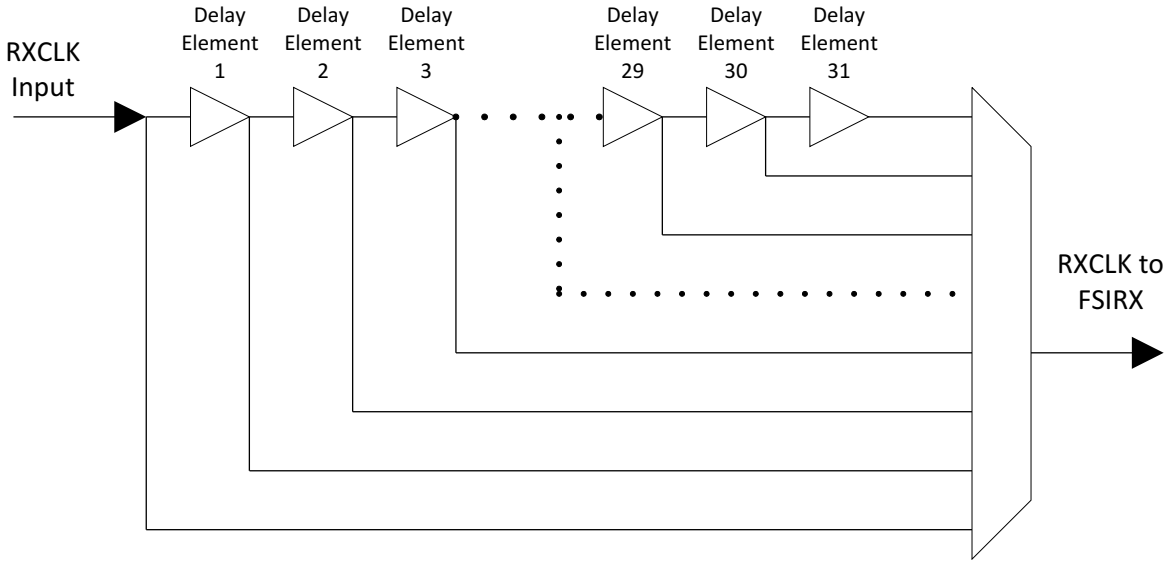
If the RX_FRAME_WD_CNT reaches the value programmed in RX_FRAME_WD_REF, the RX_EVT_STS.FRAME_WD_TO flag will be set. If enabled, an interrupt can be generated on this event.

If the frame watchdog interrupt ever occurs, the receiver core is in an invalid state to receive a new transmission. The only way to recover from a frame watchdog time out is to undergo a soft reset, and subsequently resynchronizing with the transmitter.

### 29.3.3.6 Delay Line Control

The receiver module has a programmable delay line on each of the external signal inputs: RXCLK, RXD0, and RXD1. The delay elements introduce delays on the respective lines. This is to facilitate adjustment for signal delays introduced by system level components such as signal buffers, ferrite beads, isolators, and so on, or board delays such as uneven trace lengths, long cable length, and so on. The length of the delay is controlled by setting the RX_DLY_LINE_CTRL register values for each line. By default, no delay is introduced by the delay line elements. The delay values should only be adjusted while the FSIRX is held in soft reset, ensuring that there are no active transmissions during this process. Figure 29-8 shows a representation of the delay line circuitry for the input signals. The implementation for RXCLK, RXD0, and RXD1 are replicas of this diagram. All circuits will behave similarly.

**Figure 29-8. Delay Line Control Circuit**




### 29.3.3.7 Buffer Management

The FSI receiver has a 16-word buffer into which the data is copied to when it has been received. This buffer is implemented as a circular buffer, not a FIFO, so some care must be taken to properly interpret buffer overrun and underrun as well as the RX_BUF_PTR_STS register. These flags and pointers work under the assumption that the software or DMA is using the buffer as a circular buffer. If the receiver state machine enters into an erroneous state, there is no way for software to cleanly handle this because there is no guaranteed receive clock. For the receiver to detect a clean resynchronization, the state machine needs to be operational and not in the error state. The only way to recover from the error state is to reset the entire receiver module. For overrun and underrun, the receiver can no longer guarantee which values in the buffer are valid. As such, the best way to recover is to reset the FSI and resynchronize with the transmitter.

Due to the flexibility of the receive buffer, it is possible for software to implement a simple ping-pong buffer, or to randomly receive and read from any location of the buffer. If the buffer is used in this manner, these flags and status fields may be ignored without adversely affecting the receiver capability. Additionally, the CURR_WORD_CNT will also be invalid if used in this way. The application can set the buffer pointer manually by writing the 4-bit index to RX_BUF_PTR_LOAD. This will force the receiver to start storing the received data starting at the indicated location in the buffer.

### 29.3.3.8 CRC Submodule

The receive module will automatically calculate the CRC on the incoming data. The received CRC value is placed into RX_CRC_INFO.RX_CRC. The CRC value calculated by hardware on the received data is placed into RX_CRC_INFO.CALC_CRC. These values are compared by hardware and will set RX_EVT_STS.CRC_ERROR if there is a mismatch. The receiver can generate an interrupt based on RX_EVT_STS.CRC_ERROR if enabled.

Since the CRC is only used in data frames, the values found in RX_CRC_INFO.RX_CRC and RX_CRC_INFO.CALC_CRC are undefined during ping and error frames.

For more information on how the CRC is calculated, refer to Section 29.3.7.

If the transmitting module is sending a software-defined CRC value (FSITX.TX_OPER_CTRL_LO.SW_CRC = 1) the receiver module will trigger a CRC Error event if the received value does not match the hardware-calculated value. As this is an application level decision, the FSIRX can safely disregard the CRC error event. Application software will need to calculate and verify the incoming CRC using the same custom algorithm used on the transmitter and act appropriately.

The CRC field may have also been used as an application-specific value, not a CRC, the application can use the RX_CRC_INFO.RX_CRC as required. All CRC errors and flags can be ignored in this situation.

#### 29.3.3.9 Using the Zero Bits of the Receiver Tag Registers

The receiver tag registers (receiver frame tag and user data (RX_FRAME_TAG_UDATA) register and receiver ping tag (RX_PING_TAG) register) have the Least Significant Bit is set to 0. The actual received tag is in the bit positions 4:1. The reason for this is to facilitate user software to create a table of functions which can be called depending on the tag value. A function pointer needs a 32-bit storage space and hence each successive pointer will be offset by 2. If the first pointer is at address x, then the second pointer will be at address x+2, the third at address x+4 and so on. By keeping the LSB 0, the five bits of the tag register (bits 4:0) can now be directly used as an index into a table of function pointers.

#### 29.3.3.10 Conditions in Which the Receiver Must Undergo a Soft Reset

The receiver will receive data on every clock edge. While there are specific patterns which determine the a start of a frame, and denote the end of a frame, these patterns are able to occur at any point during normal operation inside of the frame. If there ever is a point at which the receiver fails to detect a successful frame, the module must be reset in order to ensure that subsequent frames are received properly.

When any of the following errors occur in a received frame, the receiver may be required to be reset and resynchronized with the transmitter:

- Frame type error
- End of frame error
- Ping frame watchdog timeout
- Frame watchdog timeout
- Receiver in an invalid state due to noisy clock

The receiver core status (RX_VIS_1.RX_CORE_STS) can be monitored to determine if it has entered into an error state requiring a soft reset in order to resume communication. Incorrect frame type and end of frame errors will always cause this bit to become set. A soft reset is required in these cases. A frame watchdog timeout will always require a reset due to the fact that the receiver state machine is still expecting more information when the watchdog timed out. RX_CORE_STS can be used to determine if a noise event was the cause of the failed frame. The ping frame watchdog also will not cause RX_CORE_STS to be set. Similar to the frame watchdog, a corrupt receiver may not be the reason for the ping frame to have timed out. The transmitter may have gone offline and never sent a ping frame. Alternately, during idle time, a noise event may have occurred, thereby putting the receiver into a corrupt state. As the receiver will be able to detect this during the ping frame watchdog timeout interrupt handler, this type of event will not be lost and the application can act appropriately.

As the receiver is clocked by RXCLK, not SYSCLK, a noisy clock or data line may cause some internal design constraints to be violated, putting the receiver core logic into undefined states. Ensure that the clock and data lines satisfy the Electrical Characteristics and timing requirements of the FSI module found in the data manual for this device. Failure to do so may cause the receiver state machine to go into an unrecoverable error state. The receiver can only be recovered by undergoing a soft reset. To determine the state of the receiver core after an unexpected frame error, the application should check the receiver core status bit.

In addition to the above errors, buffer overrun or underrun may warrant a soft reset in order to resynchronize with the local application software. Refer to Section 29.3.3.8 for more information on the receive buffers. The requirement of resetting the receiver due to overrun or underrun is up to the application.

After the receiver has been placed into soft reset, the application must notify the other device's transmitter to begin a new synchronization phase. The simplest way to achieve this is through a ping or error frame sent with a designated tag. If the application is not using the FSITX on the device with the detected error, some other method must be established. The other device should stop transmitting and begin a new synchronization phase.

#### 29.3.3.11 Reset

The receiver module and its registers are reset by SYSRSn. The receiver core is reset by SYSRSn or by writing a 1 to RX_MASTER_CTRL.CORE_RST.

A module reset will cause the registers to be reset to their default state. After a module reset, the receiver module must be re-initialized, and the data link re-established.

### *29.3.4 Frame Format*

The FSI module transmits and receives information in frames. Each frame will contain multiple phases where different information can be found. The number of phases as well as the total length of the frame will vary depending on the frame type being transmitted. Frames may be as short as 16 bits long for a ping or error frame or 288 bits long for a 16-word data frame.

In normal transmission mode, there are four preamble clock edges before the start of the frame and four post-frame clock edges (postamble). Data is transmitted on both edges of the clock (double data rate). The basic frame structure is shown below. Each phase of the frame (such as start-of-frame, frame type, and so on) will be transmitted with the most significant bit first. Table 29-4 describes the basic frame structure used by the FSI and adapted according to which frame type is transmitted.

**Table 29-4. Basic Frame Structure**

| Idle state | Preamble | Start of Frame | Frame Type | User Data | Data Words | CRC byte | Frame Tag | End of Frame | Postamble | Idle state |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1111 | 1001 | 4 bits | 8 bits | 1-16 words | 8 bits | 4 bits | 0110 | 1111 | |

The FSI also supports a SPI compatibility mode. The SPI compatible frame structure is similar to a standard FSI frame, but there are differences. Refer to Section 29.3.9 for more information on how to configure and use the SPI compatibility mode.

**NOTE:** One word of the FSI refers to 16 bits.

The terms “frame” and “packet” can be used interchangeably to describe the signaling format of the FSI .

#### 29.3.4.1 FSI Frame Phases

The different phases of the frame structure are described in detail below:

- **Idle State**
  During the idle state, the clock and data lines will be driven high, the inactive state.
- **Preamble**
  The preamble phase contains four clock edges (or two complete clock pulses) with the data signals held in the high state. These clock edges serve to flush the receiver logic and prepare it for receiving a new frame. This phase is not present in SPI compatibility mode.
- **Start of Frame**
  The start of frame phase contains two clock pulses with four bits, 1001, transmitted on the data lines.
- **Frame Type**

The frame type phase contains two clock pulses with the 4-bit frame type code being transmitted on the data lines. The different frame types are described in detail in Section 29.3.4.2. The transmitter must set the TX_FRAME_CTRL.FRAME_TYPE field before transmitting a frame. The received frame type is stored in the RX_FRAME_INFO.FRAME_TYPE.

- **User Data**
  The user data phase contains a fully user-configurable data field. There are no restrictions on how this field is used. This phase is only available in data frames. The user data to be transmitted is set by writing to TX_FRAME_TAG_UDATA.USER_DATA. The received user data is stored in RX_FRAME_TAG_UDATA.USER_DATA.
- **Data**
  The data phase contains the data that is being transmitted. The data is pulled from the transmit buffer of the transmitter and will be placed in the receive buffer of the receiver. Word 0 is transmitted first. This phase is only present in data frames. Depending on the type of frame transmitted, this can contain anywhere between 1 and 16 words depending on the frame type selected. More information on data frames can be found in Section 29.3.4.2.3.
- **CRC Byte**
  The CRC byte contains the CRC of the transmitted data. The value present in this phase can be sourced from either hardware or software based on the TX_OPER_CTRL_LO.SW_CRC bit. Refer to the module-specific section of the CRC Submodule for more information on the CRC is generated or used, for the transmitter and receiver modules respectively. The CRC byte is only present in data frames.
- **Frame Tag**
  The frame tag contains the 4-bit user-defined frame tag. There are no restrictions on how this field is used in an application. The transmitter supplies this tag into the TX_FRAME_TAG_UDATA.FRAME_TAG bits for data frames. Ping frames use the tag defined in TX_PING_TAG.TAG. The receiver can access the received frame tag in RX_FRAME_TAG_UDATA.FRAME_TAG.
- **End of Frame**
  The end of frame contains four clock edges with four bits, 0110, transmitted on the data lines.
- **Postamble**
  The postamble contains four additional clock edges with the data lines held in the high state. After the postamble, the clock and data lines will be driven high, their inactive state. This phase is not present in SPI compatibility mode.

### 29.3.4.2 Frame Types

The FSI hardware can generate and handle many predefined frame types. The different frame types can be used by the application to signal different types of events or convey different information to the receiver. The different frame types influence which phases and data fields to include in the transmitted frames.

Table 29-5 provides a short overview of the different frame types used by the FSI. Each frame type is described in more detail in the following subsections.

**Table 29-5. Frame Types and their 4-bit codes**

| Frame Type | 4-bit Frame Code | Description |
|---|---|---|
| PING | 0000 | This is the ping frame which can be sent either by software or automatically by hardware. |
| ERROR | 1111 | This should be used typically during error conditions or any condition where one side wants to signal the other side for attention. However, the user software is at liberty to use this for any purpose. |
| DATA_1_WORD | 0100 | 1 word data packet (16 bits of data) |
| DATA_2_WORD | 0101 | 2 word data packet (32 bits of data) |
| DATA_4_WORD | 0110 | 4 word data packet (64 bits of data) |
| DATA_6_WORD | 0111 | 6 word data packet (96 bits of data) |

**Table 29-5. Frame Types and their 4-bit codes (continued)**

| Frame Type | 4-bit Frame Code | Description |
|---|---|---|
| DATA_N_WORD | 0011 | N(1-16) word data packet where software has programmed the number of the data words in a designated register. Both transmitter and receiver modules should have the same value programmed. |
| Reserved | 0001, 0010, and 1000-1110 | Reserved |

#### *29.3.4.2.1 Ping Frames*

Ping frames are one of the most basic frames that can be generated by the FSI. Table 29-6 shows the structure of the ping frames.

**Table 29-6. Ping Frame**

| Idle state | Preamble | SOF | Frame Type | Frame Tag | EOF | Postamble | Idle state |
|---|---|---|---|---|---|---|---|
| | 1111 | 1001 | 0000 | xxxx | 0110 | 1111 | |

The ping frame type is always 0000. The frame tag is defined by the application. Separate frame tags exist for timer and software initiated ping frames. No data or CRC is transmitted in a ping frame.

The main purpose of the ping frame is to periodically send a notification to the receiver to ensure an active connection between the transmitter and receiver. The transmitter and receiver cores implement different features to allow the ping frame to operate as a line break detect feature.

On the transmitter, the ping frame is the only frame which can be set up and transmitted without any further software or DMA intervention. Ping frames can be transmitted by any (or all) of the three sources: automatic ping timer, software, or external triggers. See Section 29.3.2.3.3 for information on how the transmitter configures and sends the ping frames.

The receiver has a ping watchdog that can detect if a ping frame has not been received in a predetermined window. This allows the receiver to know if the connection between it and the transmitter has been broken. See Section 29.3.3.4 for information on how the receiver handles ping frames.

#### *29.3.4.2.2 Error Frames*

Error frames are similar to ping frames in that there are no data fields transmitted. Despite the naming of this frame as an “error frame,” the usage of it is up to the application, as no restrictions are placed on how and when this type of frame is transmitted. Table 29-7 shows the structure of an error frame.

**Table 29-7. Error Frame**

| Idle state | Preamble | SOF | Frame Type | Frame Tag | EOF | Postamble | Idle state |
|---|---|---|---|---|---|---|---|
| | 1111 | 1001 | 1111 | xxxx | 0110 | 1111 | |

The structure of the error frame is the same as a ping frame. No data or CRC values are transmitted. The frame type is 1111 for all error frames, and the frame tag is defined by software in the TX_FRAME_TAG_UDATA register.

The receiver can detect if an error frame has been received based on the frame type field. Because of this, the receiver can read the incoming frame tag from the RX_FRAME_TAG_UDATA register and act on up to 16 different conditions.

#### *29.3.4.2.3 Data Frames*

Data frames are the most complex frames. As the name indicates, these frames are used to transfer data. Table 29-8 shows the general structure of data frames.

**Table 29-8. Data Frame**

| Idle state | Preamble | SOF | Frame Type | User Data | Data Words | CRC Byte | Frame Tag | EOF | Postamble | Idle state |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1111 | 1001 | 0xxx | xxxx xxxx | 1-16 words | xxxx xxxx | xxxx | 0110 | 1111 | |

The frame type field will reflect the 4-bit code of the frame type. A list of frame types can be seen in Table 29-5. The number of the data words transmitted will be determined by the frame type chosen.

There are four fixed-length data frames supported by the frame type: 1 word, 2 words, 4 words, and 6 words.

Additionally, there is a user-defined data length frame type where the number of data words is fixed by software. Anywhere from 1 to 16 words can be transmitted in this frame type. This length must be configured in the N_WORDS field of the transmitter's TX_FRAME_CTRL register and receiver's RX_OPER_CTRL register.

#### 29.3.4.3 Multi-Lane Transmission

The FSI is capable of transmitting and receiving data on two parallel data lines. When enabled, data bits will be split between the data lines while the start of frame, frame type, frame tag, and end of frame fields will be identical and complete on each line. The user data, data, and CRC fields will be split between the data lines. Starting with the most significant bit, the odd-numbered bits appear on D0 and even-numbered bits appear on D1.

In the example below, assume the following:

8-bit user data: u7u6u5u4u3u2u1u0

16-bit data: d15d14d13d12…d1d0

8-bit CRC: c7c6c5c4c3c2c1c0

**Table 29-9. Multi-Lane Frame Format**

| Idle state | Preamble | SOF | Frame Type | User Data | Data Words | CRC Byte | Frame Tag | EOF | Postamble | Idle state |
|---|---|---|---|---|---|---|---|---|---|---|
| TXD0 | 1111 | 1001 | 0011 | $u^7u^5u^3u^1$ | $d^{15}d^{13}…d^1$ | $c^7c^5c^3c^1$ | xxxx | 0110 | 1111 | |
| TXD1 | 1111 | 1001 | 0011 | $u^6u^4u^2u^0$ | $d^{14}d^{12}…d^0$ | $c^6c^4c^2c^0$ | xxxx | 0110 | 1111 | |

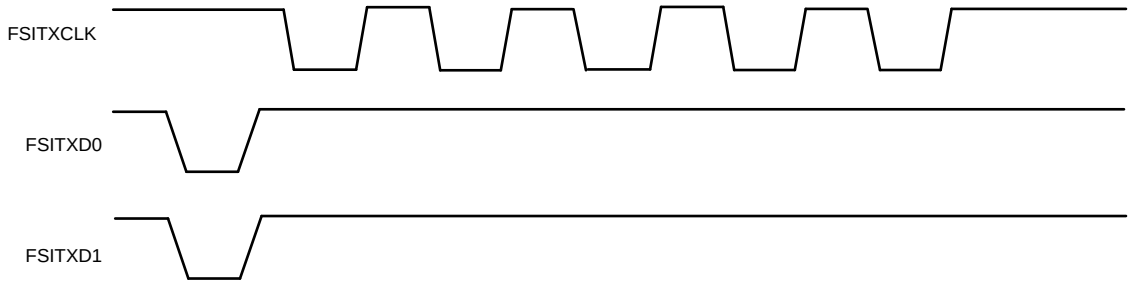
### *29.3.5 The Flush Sequence*

Every time there is a soft reset of the receiver, the receiver requires a flush sequence from the transmitter before it can receive and decode frames. The receiver core has an asynchronous reset mechanism which will allow the receive module to be reset even in the absence of the receive clocks. However, due to the design, this reset will be released synchronous to the receive clock (RXCLK). Thus, the receiver will require five full clock pulses to be able to come out of reset. Sending the flush pattern will ensure that these clock edges are received and any subsequent frames sent to the receiver will be correctly interpreted.

The flush sequence consists of a single toggle on both of the data lines as well as five consecutive pulses on the clock line.

Figure 29-9 shows a sample plot of the flush sequence.

**Figure 29-9. Flush Sequence Signals**



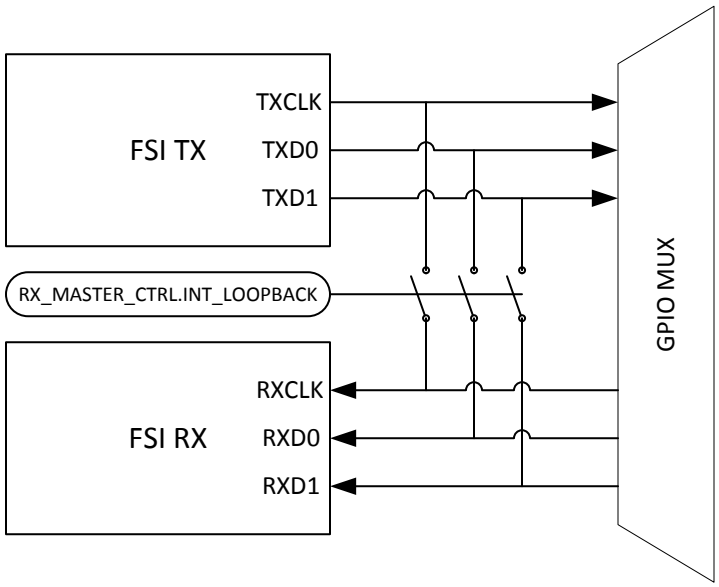

If the FSI receiver is receiving data from a standard SPI, a data word of 0xFFFF from the SPI will have the same effect as a flush sequence.

### 29.3.6 Internal Loopback

The transmitter and receiver cores can be connected together internally to allow for development and debug. This is achieved by setting RX_MASTER_CTRL.INT_LOOPBACK to 1. Internal loopback will route the signals from the corresponding transmitter to the appropriate receiver pin. No configuration needs to be done in the transmitter.

Figure 29-10 shows the signal connections with internal loopback.

**Figure 29-10. FSI with Internal Loopback**




### 29.3.7 CRC Generation

The FSI uses CRC-8 with the polynomial 0x07 for the internal hardware CRC generation. This polynomial is also represented as $x^8+x^2+x+1$.

For example, for a 2-word data packet the following calculation would occur:

Data-1 = 0x4433

Data-0 = 0x2211

User Data = 0xAA

The CRC would be computed with the bytes being taken in the following order (first to last)

0xAA – Byte 0, User Data

0x11 – Byte 1, Data-0, Least significant byte

0x22 – Byte 2, Data-0, Most significant byte

0x33 – Byte 3, Data-1, Least significant byte

0x44 – Byte 4, Data-1, Most significant byte

### 29.3.8 ECC Module

The FSI module comes with a 16-bit or 32-bit ECC computation module in both the transmitter and receiver. Use of this module is optional.

Note that the ECC is independent and unrelated to the hardware CRC computation module present in both the transmitter and receiver cores.

The following example shows a scenario in which the application requires ECC be calculated and transmitted on a 2-word data frame.

In the FSITX module:

1. Configure the ECC module for 32-bit data by setting TX_OPER_CTRL_HI.ECC_SEL to 1.
2. Write the data to the TX_ECC_DATA register as well as the transmit buffer.
3. Read TX_ECC_VAL Register. This register contains the 8-bit ECC value calculated on the data.
4. Copy the 8-bit data from TX_ECC_VAL to TX_FRAME_TAG_UDATA.USER_DATA.
5. Set the Start Condition to begin the transmission.

The reverse process is followed on the FSIRX module. Once the data frame is received, user software can do the following:

1. Copy the data from the receive buffer to the RX_ECC_DATA register.
2. Copy the received user data that contains the transmitted ECC value from RX_FRAME_TAG_UDATA.USER_DATA to the RX_ECC_VAL register.
3. Read the RX_ECC_LOG register. This contains the result of the ECC computation using the RX_ECC_DATA and RX_ECC_VAL registers.
    1. If no ECC errors were detected, RX_ECC_LOG will be 0. The correct data will be available in RX_ECC_SEC_DATA.
    2. If a single bit error was detected, RX_ECC_LOG.SBE will be 1. The autocorrected data will be available in RX_ECC_SEC_DATA.
    3. If multiple bit errors occurred, RX_ECC_LOG.MBE will be set to 1. The data in RX_ECC_SEC_DATA is invalid and should not be used.

Using a 2-word data frame plus using the user data for the ECC is one possible implementation for ECC detection. Another option is to use a larger data frame a allocate one of the data words to be the ECC value.

### 29.3.9 SPI Compatibility Mode

The FSI supports a SPI compatibility mode. While the FSI can communicate with a standard SPI module, the FSI supports a limited configuration. The features of this compatibility mode follow:

- Data will transmit on rising edge and receive on falling edge of the clock.
- Only 16-bit word size is supported.
- TXD1 will be driven like an active-low, chip-select signal. The signal will be low for the duration for the full frame transmission.
- No receiver chip-select input is required. RXD1 is not used. Data is shifted into the receiver on every active clock edge.
- No preamble or postamble clocks will be transmitted. All signals return to the IDLE state after the frame phase is finished.
- It is not possible to transmit in the SPI slave configuration because the FSI TXCLK cannot take an external clock source.

Table 29-10 lists the frame structure of the SPI compatibility mode. Each frame phase is present in this mode. If the FSI is transmitting to a standard SPI module, the SPI must decode the frame structure. Similarly, if the FSI is configured as a SPI slave, the standard SPI must encode the transmission to be sent.

**Table 29-10. SPI Compatibility Frame Structure**

| Idle state | Start of Frame | Frame Type | User Data | Data Words | CRC byte[1] | Frame Tag | End of Frame | Idle state |
|---|---|---|---|---|---|---|---|---|
| | 1001 | 4 bits | 8 bits | 1-16 words | 8 bits | 4 bits | 0110 | |

(1) The CRC byte is present only in data frames.

Because of the requirement that the standard SPI module encodes the various frame data, this limits the type of modules that can be connected to the FSI in SPI mode. The paired SPI module must have enough functionality to encode and decode the frames.

If the FSI is transmitted to a standard 16-bit SPI, the data would be arranged in the following manner. The example provided in Table 29-11 assumes a DATA_2_WORD frame has been sent.

**Table 29-11. Contents of Data Received by a Standard SPI**

| SPI Data | Data Contents |
|---|---|
| SPI word 0 | 1001, 0100, 8-bit User Data |
| SPI word 1 | Data word 1 |
| SPI word 2 | Data word 2 |
| SPI word 3 | 8-bit CRC, 4-bit Frame Tag, 0110 |

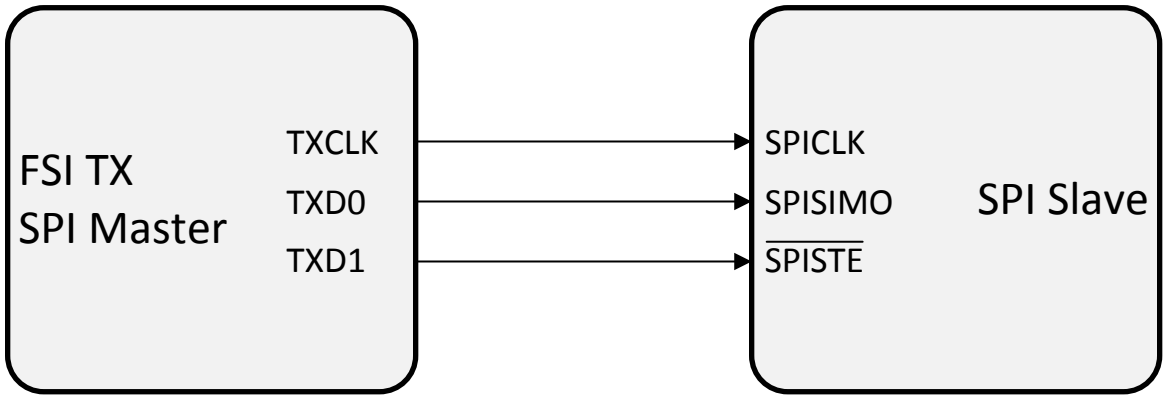
### 29.3.9.1 Available SPI Modes

There are a few wiring schemes available for the FSI to use when communicating with a SPI module.

#### *29.3.9.1.1 FSITX as SPI Master, Transmit Only*

The FSITX can operate as an independent SPI Master module. In this condition, TXCLK will be connected to SPICLK, TXD0 will be connected to SPISIMO, and TXD1 will be connected to $\overline{\text{SPISTE}}$, the chip select.

**Figure 29-11. FSITX as SPI Master, Transmit Only**




When the FSI is a SPI transmitter, the application has the ability to check for frame errors, line breaks, CRC errors, and ECC checks on data. These are all encoded by hardware in every FSI frame. The SPI receiver will require some software to act upon this information.

**Table 29-12. FSI as Master Transmitter, SPI as Slave Receiver**

| Capability | Availability | Comment |
|---|---|---|
| Framing checks on the data frames | Yes | Can be implemented in software on the SPI receiver. |
| Ability to detect line breaks | Yes | Can be implemented in software on the SPI receiver but will require additional software overhead such as a timer, or watchdog. |
| CRC check | Yes | Can be implemented in software on the SPI receiver. For devices which have VCU, this will be more efficient. |
| ECC on data | Yes | Can be implemented in software in the SPI receiver |
| Detection of abruptly terminated frames | No | |
| Double edge data rate | No | |
| Recovery from glitches on signal lines between frames | No | |
| Skew adjustment on signal lines | No | |

#### *29.3.9.1.1.1 Initialization*

To configure the FSITX module to be a SPI master for transmit only, proceed through the standard FSITX initialization procedure. Before releasing the FSITX from reset, set TX_OPER_CTRL_LO.SPI_MODE to 1. This will enable the SPI clocking scheme and signaling structure.

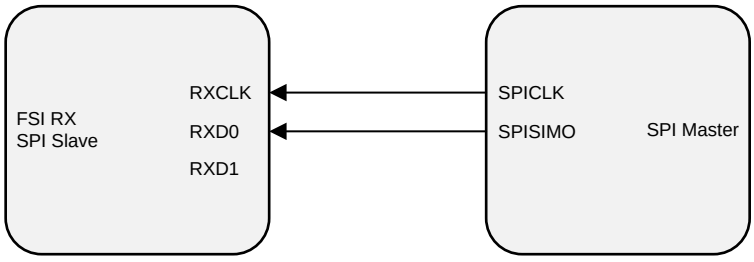
#### *29.3.9.1.1.2 Operation*

The operation of the FSITX module in SPI Compatibility mode is the same as if the module is in standard FSI mode. The application can utilize the frame timer, ping frames, external frame triggers, and so on. Refer to Section 29.3.2 for more information on each of these features.

### *29.3.9.1.2 FSIRX as SPI Slave, Receive Only*

The FSIRX can operate as an independent SPI slave module. In the usage, RXCLK will be connected to SPICLK, and RXD0 will be connected to SPISIMO. RXD1 is unused. There is no requirement for a chip select signal to be used when connected to the FSIRX. This is because the FSIRX will respond to any incoming clock edge. If there is any noise or unwanted clock transitions, a flush sequence will be required to resynchronize the FSIRX module with the Master.

**Figure 29-12. FSIRX as SPI Slave, Receive Only**




When the FSI is a SPI receiver communicating with a SPI transmitter, the application has the ability to detect frame errors, line breaks, CRC errors, ECC checks on data, as well as abruptly terminated frames. Note that the FSI can handle all of this in hardware, but the SPI transmitter must encode the information into the data to be transmitted.

**Table 29-13. SPI as master transmitter, FSI as slave receiver**

| Capability | Availability | Comment |
|---|---|---|
| Framing checks on the data frames | Yes | Standard on FSI |
| Ability to detect line breaks | Yes | Can be implemented in software in the SPI transmitter. But will have to use some timer or watchdog in the transmitting SPI device |
| CRC check | Yes | Can be implemented in software in the SPI transmitter. |
| ECC on data | Yes | Can be implemented in software in the SPI transmitter. |
| Detection of abruptly terminated frames | Yes | This is accomplished with the FSI setting up the frame watchdog counter. |
| Double edge data rate | No | |
| Recovery from glitches on signal lines between frames | Yes | Whenever glitches occur on either the clock or data lines in between transmissions, the initial flush pattern of a frame will discard the effects of these glitches and will cause the receiver to resynchronize when the real “start-of-frame” pattern is seen. So, the ability to reject glitches in between frames is hence very high. |
| Skew adjustment on signal lines | Yes | The FSI receiver has the ability to add delays to the incoming signal lines. |

#### *29.3.9.1.2.1 Initialization*

To configure the FSIRX module to be a SPI slave for receiving only, proceed through the standard FSIRX initialization procedure. Before releasing the FSIRX from reset, set RX_OPER_CTRL.SPI_MODE to 1. This will enable the SPI clocking scheme and signaling structure.

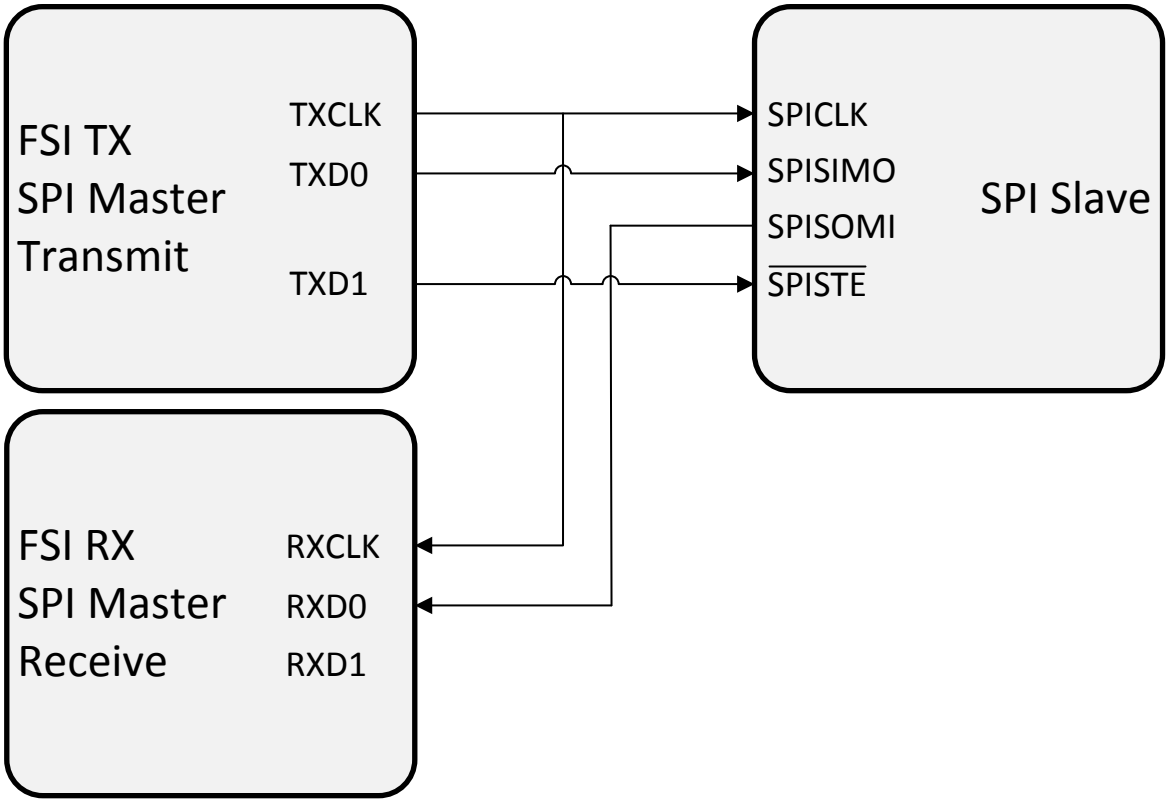
#### *29.3.9.1.2.2 Operation*

The operation of the FSIRX module in SPI compatibility mode is the same as if the module is in standard FSI mode. The application can utilize the Frame and Ping Watchdogs, CRC and ECC checks, and so on. Refer to Section 29.3.3 for more information on each of these features.

### *29.3.9.1.3 FSITX and FSIRX Emulating a Full Duplex SPI Master*

In this configuration, the FSITX is the clock master. The FSITX module will drive TXCLK (SPICLK), TXD0 (SPISIMO), and TXD1 (SPISTE/chip select), to the SPI slave. The SPISOMI signal will be connected back to the RXD0 signal. RXCLK can be fed either internally using the internal SPI pairing feature or wired externally depending on the application requirements. Since the FSITX and RX modules are independent, the FSIRX could also be thought of as an additional SPI slave. Some software logic will be required in order for the FSI to emulate a SPI master fully.

**Figure 29-13. FSITX and FSIRX as SPI Master, Full Duplex**




#### *29.3.9.1.3.1 Initialization*

To configure both FSITX and RX modules for full duplex SPI master operation, follow the initialization instructions for each module described in the preceding sections. Both FSITX and RX modules must set their respective SPI_MODE bits. This will enabled the SPI clocking scheme and signaling structures.

If internal clock loopback is desired, the FSIRX module must also set RX_MASTER_CTRL.SPI_PAIRING to 1. This will internally connect TXCLK to RXCLK. If using internal clock loopback, the GPIO used for RXCLK may be reallocated to other application requirements.

If the application requires an external clock loopback ensure that TXCLK is connected to RXCLK. This may be required if the SPI Slave is across an isolation barrier and there is latency between TXCLK being launched and SPISOMI data being received on RXD0.

#### *29.3.9.1.3.2 Operation*

In this mode of operation, some higher level software must be written to emulate a full SPI master module. There is no path for the transmit module to know what the receive module received. Both the TX and RX modules are still able to utilize the various other features available such as the ping frame timer, ping frame and frame watchdogs, CRC and ECC error checkers, and so on. The procedure for configuring these features is described elsewhere in this document.

## 29.4 Programmer's Model

This section describes various operational sequences and features for the FSI.

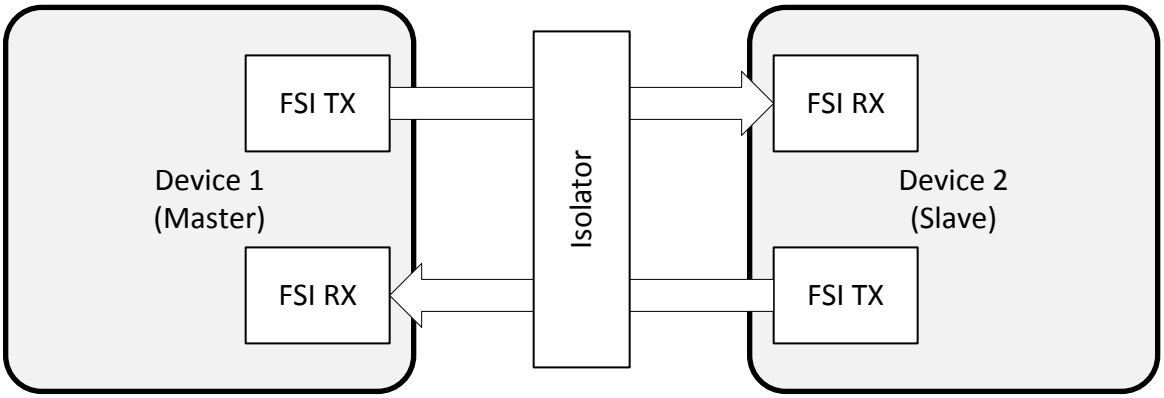
### 29.4.1 Establishing the Communication Link

Once the transmitter and receiver modules have been configured, some synchronization must occur before the modules may exchange data. Since the receiver will accept data on any clock transition, the receiver core logic must be flushed to properly interpret the start of a new, valid frame. This is especially true when the FSI modules reside on separate devices and are possibly isolated.

The following example walks through a suggested approach for establishing a clean communication link on two separate devices that may power up in an arbitrary order. Note that this is only a sample synchronization. Depending on application requirements, a different approach can be followed. The single most important aspect of synchronization is to ensure that the receiver is properly flushed and ready to receive a complete frame without error. How to achieve this is up to the application.

Figure 29-14 shows the connection of the devices in this example. While there is no true concept of a master or a slave node in the FSI protocol, the example uses this nomenclature as a simple way to describe the data flow.

**Figure 29-14. Point to Point Connection**




Device 1 is the master node; it will be the driver of the initialization sequence. Device 2 is the slave node; it will respond to the master's commands. In this example as well as in a real world use-case, neither the master nor the slave may know precisely when the other is ready to receive communication.

Sample sequences for both the master device and slave device is provided below:

#### 29.4.1.1 Establishing the Communication Link From the Master

The following sequence is an example of how the master node can establish the communication link with the slave without external signals outside of the standard communication link.

1. Assert the core reset to both the FSITX and FSIRX modules, and then deassert the resets.
2. Configure the transmitter and receiver for desired operation.
3. Set up the receiver interrupts to detect an incoming transmission.
4. Begin the ping loop:
   - Send the flush sequence.
   - Send a ping frame with the frame tag 0000.
   - Wait for some time. (determined by application)
   - If the FSIRX has received a valid ping frame, continue; else iterate the loop again.
   - If the received ping frame tag was 0001, continue; else iterate the loop again.
5. Send a ping frame with the frame tag 0001.

At this point both the master transmit and receive channels have successfully received a frame from their slave counterparts. The link has been established and standard application communication may begin.

#### 29.4.1.2 Establishing the Communication Link from the Slave

The following sequence is an example of how the slave node can establish the communication link with the master without external signals outside of the standard communication link.

1. Apply the core reset to both the FSITX and FSIRX modules, and then release the reset.
2. Configure the transmitter and receiver for desired operation.
3. Set up the receiver interrupts to detect an incoming transmission.
4. Wait for a receiver interrupt.
5. If the FSIRX has received a valid ping frame, continue; else return to step 4.
6. If the received frame tag was 0000, continue; else discard the transmission and return to step 4.
7. Send the flush sequence.
8. Send a ping frame with the frame tag 0001.
9. Wait for a receiver interrupt.
10. If the FSIRX has received a valid ping frame, continue; else return to step 4.
11. If the received ping frame tag was 0001, continue; else if the received frame tag was 0000, return to step 9. This can happen if a second ping frame was already in transit before receiving the slave's response in step 8.

At this point, both the transmit and receive modules have successfully received ping frames from their master counterparts. The link has been established and regular communication may now proceed. The application may configure periodic ping frames from the transmitter, initialize the receiver's ping and frame watchdogs and begin the communication required by the application.

### *29.4.2 Register Protection*

Both the FSITX and FSIRX modules contain control registers that have embedded write protection. This is accomplished through EALLOW, register keys, and a master register lock. These protections ensure that no spurious writes or unintentional modifications to these registers are accepted. Refer to the Section 29.5 for the register and bit descriptions to find the list of registers with write protections available.

**EALLOW Protection**

EALLOW is a device-level register protection; refer to Section 3.1 for more information on EALLOW. For those registers with EALLOW protection, the EALLOW bit should be set before modifying the register. The application should then clear the EALLOW bit to re-enable the write protection when access to EALLOW-protected registers are complete.

**Register Key Protection**

In addition to EALLOW, some bits in the FSI registers are protected by a key. In order to write to these bits, the key must be written at the same time. For example, to put the transmitter core into reset, TX_MASTER_CTRL.CORE_RST must be set. To do this, write 0xA501 to TX_MASTER_CTRL, where 0xA500 is the KEY value, and 0x0001 is the CORE_RESET bit. Refer to the register descriptions for more information on which registers have write keys added.

**Control Register Lock Protection**

There also exists a master lock to prevent any modifications to the control registers. There is an independent lock for each FSI module. For the list of registers that are protected by this control register lock, refer to the register descriptions section. The control register lock will prevent any writes to the control registers until the lock is released. To set the control register lock, write 0xA501 to RX_LOCK_CTRL and TX_LOCK_CTRL for the receiver and transmitter respectively.

The control register lock cannot be disabled by the application until a SYSRSn has been asserted. This can occur at the device level, or by writing to the appropriate peripheral soft reset register (DEV_CFG_REGS.SOFTPRESx) for the FSI module. Refer to Section 29.3.2.7 for more information on SYSRSn.

### 29.4.3 Emulation Mode

There is no specific emulation mode or configuration supported. The FSI cores will always be in free running mode. CPU halts will not have any effect on the operation of the FSI. However, reads of registers and data buffers by the debugger will not affect any flags, or status of the data buffers.

If the user wishes to stop the operation of either FSI module when the debugger halts, the following steps are required:

1. Set the debugger to real-time emulation mode.
2. Mark the FSI interrupt group as a time-critical interrupt. That is, enable the corresponding bit in the DBGIER register.
3. The ISR can check the DSTAT register and to determine if the ISR was called when the debugger was halted.
4. FSI operations can be disabled and the ISR can branch to a debug-specific halt location.

## 29.5 Registers

### 29.5.1 Fast Serial Interface Base Addresses

The FSI module contains two distinct sets of registers. One for the FSI receiver, and another for the FSI transmitter.

**Table 29-14. FSI Base Address Table**

| Device Registers | Register Name | Start Address | End Address |
|---|---|---|---|
| FsiTxaRegs | FSI_TX_REGS | 0x0000_6600 | 0x0000_667F |
| FsiRxaRegs | FSI_RX_REGS | 0x0000_6680 | 0x0000_66FF |

### 29.5.1.1 FSI_RX_REGS Registers

Table 29-15 lists the memory-mapped registers for the FSI_RX_REGS. All register offset addresses not listed in Table 29-15 should be considered as reserved locations and the register contents should not be modified.

**Table 29-15. FSI_RX_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | RX_MASTER_CTRL | Receive master control register | EALLOW | Go |
| 4h | RX_OPER_CTRL | Receive operation control register | EALLOW and LOCK | Go |
| 6h | RX_FRAME_INFO | Receive frame control register | | Go |
| 7h | RX_FRAME_TAG_UDATA | Receive frame tag and user data register | | Go |
| 8h | RX_DMA_CTRL | Receive DMA event control register | EALLOW and LOCK | Go |
| Ah | RX_EVT_STS | Receive event and error status flag register | | Go |
| Bh | RX_CRC_INFO | Receive CRC info of received and computed CRC | | Go |
| Ch | RX_EVT_CLR | Receive event and error clear register | EALLOW | Go |
| Dh | RX_EVT_FRC | Receive event and error flag force register | EALLOW | Go |
| Eh | RX_BUF_PTR_LOAD | Receive buffer pointer load register | EALLOW | Go |
| Fh | RX_BUF_PTR_STS | Receive buffer pointer status register | | Go |
| 10h | RX_FRAME_WD_CTRL | Receive frame watchdog control register | EALLOW and LOCK | Go |
| 12h | RX_FRAME_WD_REF | Receive frame watchdog counter reference | EALLOW and LOCK | Go |
| 14h | RX_FRAME_WD_CNT | Receive frame watchdog current count | | Go |
| 16h | RX_PING_WD_CTRL | Receive ping watchdog control register | EALLOW and LOCK | Go |
| 17h | RX_PING_TAG | Receive ping tag register | | Go |
| 18h | RX_PING_WD_REF | Receive ping watchdog counter reference | EALLOW and LOCK | Go |
| 1Ah | RX_PING_WD_CNT | Receive pingwatchdog current count | | Go |
| 1Ch | RX_INT1_CTRL | Receive interrupt control register for RX_INT1 | EALLOW and LOCK | Go |
| 1Dh | RX_INT2_CTRL | Receive interrupt control register for RX_INT2 | EALLOW and LOCK | Go |
| 1Eh | RX_LOCK_CTRL | Receive lock control register | | Go |
| 20h | RX_ECC_DATA | Receive ECC data register | | Go |
| 22h | RX_ECC_VAL | Receive ECC value register | | Go |
| 24h | RX_ECC_SEC_DATA | Receive ECC corrected data register | | Go |
| 26h | RX_ECC_LOG | Receive ECC log and status register | | Go |
| 30h | RX_DLYLINE_CTRL | Receive delay line control register | EALLOW and LOCK | Go |
| 38h | RX_VIS_1 | Receive debug visibility register 1 | | Go |
| 40h | RX_BUF_BASE | Base address for receive data buffer | EALLOW and LOCK | Go |

Complex bit access types are encoded to fit into small table cells. Table 29-16 shows the codes that are used for access types in this section.

**Table 29-16. FSI_RX_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |

**Table 29-16. FSI_RX_REGS Access Type Codes (continued)**

| Access Type | Code | Description |
|---|---|---|
| **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. |

#### 29.5.1.1.1 RX_MASTER_CTRL Register (Offset = 0h) [reset = 0h]

RX_MASTER_CTRL is shown in Figure 29-15 and described in Table 29-17.

Return to Summary Table.

Receive master control register

**Figure 29-15. RX_MASTER_CTRL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| KEY | | | | | | | |
| W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | SPI_PAIRING | INT_LOOPBACK | CORE_RST |
| R-0h | | | | | R/W-0h | R/W-0h | R/W-0h |

**Table 29-17. RX_MASTER_CTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | KEY | W | 0h | Write Key.<br>In order to write to this register, 0xA5 must be written to this field at the same time. Otherwise, writes are ignored. The key is cleared immediately after writing, so it must be written again for every change to this register.<br>Reset type: SYSRSn |
| 7-3 | RESERVED | R | 0h | Reserved |
| 2 | SPI_PAIRING | R/W | 0h | Clock Pairing for SPI-like Behavior Enable bit<br>This bit enables the internal clock pairing with the FSI TX module. This feature internally connects the TXCLK to RXCLK allowing the FSI TX module, acting as a SPI master, to clock data into the receiver and out of the transmitter like a standard SPI module. This configuration is valid when the Module is in SPI mode only (RX_OPER_CTRL.SPI_MODE = 1)<br>0h (R/W) = SPI clock pairing is not enabled.<br>1h (R/W) = SPI clock pairing is enabled. The RXCLK will be internally connected to the TXCLK of the corresponding FSI module.<br>Note: The KEY field must contatin 0xA5 for any write to this bit to take effect.<br>Reset type: SYSRSn |
| 1 | INT_LOOPBACK | R/W | 0h | Internal Loopback Enable bit<br>This bit enables the internal loopback functionality of the FSI receiver. By enabling this bit, a mux will select the signals coming directly from the corresponding FSI transmitter module rather than from the pins.<br>0h (R/W) = Internal loopback is disabled. The FSI RX module will receive signals coming from the pins.<br>1h (R/W) = Internal loopback is enabled. The FSI RX module will receive signals from the directly from FSI TX module rather than the pins.<br>Note: The KEY field must contatin 0xA5 for any write to this bit to take effect.<br>Reset type: SYSRSn |

**Table 29-17. RX_MASTER_CTRL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | CORE_RST | R/W | 0h | Receiver Master Core Reset bit<br>This bit controls the receiver master core reset. In order to receive any frame, this bit must be cleared.<br>0h (R/W) = Receiver core is not in reset and can receive frames.<br>1h (R/W) = Receiver core is held in reset.<br>Note: The KEY field must contatin 0xA5 for any write to this bit to take effect.<br>Reset type: SYSRSn |

#### 29.5.1.1.2 RX_OPER_CTRL Register (Offset = 4h) [reset = 0h]

RX_OPER_CTRL is shown in Figure 29-16 and described in Table 29-18.

Return to Summary Table.

Receive operation control register

**Figure 29-16. RX_OPER_CTRL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | PING_WD_RST_MODE |
| R-0h | | | | | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ECC_SEL | N_WORDS | | | | SPI_MODE | DATA_WIDTH | |
| R/W-0h | R/W-0h | | | | R/W-0h | R/W-0h | |

**Table 29-18. RX_OPER_CTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-9 | RESERVED | R | 0h | Reserved |
| 8 | PING_WD_RST_MODE | R/W | 0h | Ping Watchdog Timeout Mode Select bit<br>This bit selects the mode by which the ping watchdog counter is reset. The watchdog counter can be reset and restarted only by ping frames or by any received frame.<br>0h (R/W) = The ping watchdog counter will reset and restart only by ping frames.<br>1h (R/W) = The ping watchdog counter will reset and restart by any received frame.<br>Reset type: SYSRSn |
| 7 | ECC_SEL | R/W | 0h | ECC Data Width Select bit<br>This bit selects between whether the ECC computation is done on 16-bit or 32-bit words.<br>0h (R/W) = 32-bit ECC is used.<br>1h (R/W) = 16-bit ECC is used.<br>Reset type: SYSRSn |
| 6-3 | N_WORDS | R/W | 0h | Number of Words to Receive<br>This field defines the number of words which will be received in a DATA_N_WORD frame. This is a user-defined field that must match the corresponding field in the transmitter. Set this bitfield to be one greater than the number of words to be received. This value is only applicable when the frame type received is DATA_N_WORD.<br>0h (R/W) = 1 data word frame (16-bit data).<br>1h (R/W) = 2 data word frame (32-bit data).<br>..<br>Fh (R/W) = 16 data word frame (256-bit data).<br>Reset type: SYSRSn |
| 2 | SPI_MODE | R/W | 0h | SPI Mode Enable bit<br>This bit enables and disables the SPI compatibility mode of the FSI RX. The received data must be formatted as an FSI frame in order for the data to properly be received. SPI compatibility mode will allow FSI RX to receive data that is sent using SPI signal format. Refer to the applicable section in the FSI TRM chapter for more information.<br>0h (R/W) = FSI is in normal mode of operation.<br>1h (R/W) = FSI is operating in SPI compatibility mode.<br>Reset type: SYSRSn |

**Table 29-18. RX_OPER_CTRL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1-0 | DATA_WIDTH | R/W | 0h | Receive Data Width Select bit<br>These bits decide the number of data lines used for receiving data.<br>0h (R/W) = Data will be received on one data line, RXD0.<br>1h (R/W) = Data will be received on two data lines, RXD0 and RXD1.<br>2h, 3h (R/W) = Reserved<br>Reset type: SYSRSn |

#### 29.5.1.1.3 RX_FRAME_INFO Register (Offset = 6h) [reset = 0h]

RX_FRAME_INFO is shown in Figure 29-17 and described in Table 29-19.

Return to Summary Table.

Receive frame control register

**Figure 29-17. RX_FRAME_INFO Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | FRAME_TYPE | | | |
| R-0h | | | | R-0h | | | |

**Table 29-19. RX_FRAME_INFO Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-4 | RESERVED | R | 0h | Reserved |
| 3-0 | FRAME_TYPE | R | 0h | Received Frame Type<br>This field indicates the type of frame that was successfully received last.<br>0000b (R/W) = A ping frame was received<br>0100b (R/W) = A DATA_1_WORD frame was received (16-bit data).<br>0101b (R/W) = A DATA_2_WORD frame was received (32-bit data).<br>0110b (R/W) = A DATA_4_WORD frame was received (64-bit data).<br>0111b (R/W) = A DATA_6_WORD frame was received (96-bit data).<br>0011b (R/W) = A DATA_N_WORD frame was received. The N_WORD field will determine the number of words (1 to 16) to be sent. The number of words received must equal the value programmed in RX_OPER_CTRL.N_WORDS.<br>1111b (R/W) = An error frame was received. This frame can be used during error conditions or any condition where the transmitter wants to signal the receiver for attention. However, the user software is at liberty to use this for any purpose.<br>0001b, 0010b, and 1000b through 1110b are Reserved and should not be used.<br>Reset type: SYSRSn |

#### 29.5.1.1.4 RX_FRAME_TAG_UDATA Register (Offset = 7h) [reset = 0h]

RX_FRAME_TAG_UDATA is shown in Figure 29-18 and described in Table 29-20.

Return to Summary Table.

Receive frame tag and user data register

**Figure 29-18. RX_FRAME_TAG_UDATA Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| USER_DATA | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | FRAME_TAG | | | | RESERVED |
| R-0h | | | R-0h | | | | R-0h |

**Table 29-20. RX_FRAME_TAG_UDATA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | USER_DATA | R | 0h | Received User Data<br>This field contains the 8-bit user data field of the last successfully received frame.<br>Reset type: SYSRSn |
| 7-5 | RESERVED | R | 0h | Reserved |
| 4-1 | FRAME_TAG | R | 0h | Received Frame Tag<br>This field contains the 4-bit frame tag from the last successfully received frame. This is intentionally shifted into bits 4:1 so that the register can be used as a 32-bit address index based on the received tag.<br>Reset type: SYSRSn |
| 0 | RESERVED | R | 0h | Reserved |

#### 29.5.1.1.5 RX_DMA_CTRL Register (Offset = 8h) [reset = 0h]

RX_DMA_CTRL is shown in Figure 29-19 and described in Table 29-21.

Return to Summary Table.

Receive DMA event control register

**Figure 29-19. RX_DMA_CTRL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | DMA_EVT_EN |
| R-0h | | | | | | | R/W-0h |

**Table 29-21. RX_DMA_CTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-1 | RESERVED | R | 0h | Reserved |
| 0 | DMA_EVT_EN | R/W | 0h | DMA Event Enable bit<br>This bit will enable a DMA Event to be generated upon the completion of a frame reception.<br>0h (R/W) = A DMA event will not be generated.<br>1h (R/W) = A DMA event will be generated upon the reception of a frame.<br>Note: The DMA event will only be generated for data frames.<br>Reset type: SYSRSn |

#### 29.5.1.1.6 RX_EVT_STS Register (Offset = Ah) [reset = 0h]

RX_EVT_STS is shown in Figure 29-20 and described in Table 29-22.

Return to Summary Table.

Receive event and error status flag register

**Figure 29-20. RX_EVT_STS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | DATA_FRAME | FRAME_OVERRUN | PING_FRAME | ERR_FRAME |
| R-0h | | | | R-0h | R-0h | R-0h | R-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BUF_UNDERRUN | FRAME_DONE | BUF_OVERRUN | EOF_ERR | TYPE_ERR | CRC_ERR | FRAME_WD_TO | PING_WD_TO |
| R-0h | R-0h | R/W-0h | R-0h | R-0h | R-0h | R-0h | R-0h |

**Table 29-22. RX_EVT_STS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11 | DATA_FRAME | R | 0h | Data Frame Received Flag<br>This bit indicates that an data frame has been received. Software can also force this bit to get set by writing to the RX_EVT_FRC register.<br>0h (R) = No data frame has been received.<br>1h (R) = A data frame has been received.<br>To clear this bit, write to the corresponding bit in the RX_EVT_CLR register.<br>Reset type: SYSRSn |
| 10 | FRAME_OVERRUN | R | 0h | Frame Overrun Flag<br>This bit indicates that a frame overrun condition has occured. This bit will be set when a new frame has been received while the FRAME_DONE flag is still set. Software can also force this bit to get set by writing to the RX_EVT_FRC register.<br>0h (R) = Frame overrun has not ocurred.<br>1h (R) = Frame overrun has ocurred.<br>To clear this bit, write to the corresponding bit in the RX_EVT_CLR register.<br>Reset type: SYSRSn |
| 9 | PING_FRAME | R | 0h | Ping Frame Received Flag<br>This bit indicates that an ping frame has been received. Software can also force this bit to get set by writing to the RX_EVT_FRC register.<br>0h (R) = No ping frame has been received.<br>1h (R) = A ping frame has been received.<br>To clear this bit, write to the corresponding bit in the RX_EVT_CLR register.<br>Reset type: SYSRSn |

**Table 29-22. RX_EVT_STS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 8 | ERR_FRAME | R | 0h | Error Frame Received Flag<br>This bit indicates that an error frame has been received. Software can also force this bit to get set by writing to the RX_EVT_FRC register.<br>0h (R) = No error frame has been received.<br>1h (R) = An error frame has been received.<br>To clear this bit, write to the corresponding bit in the RX_EVT_CLR register.<br>Reset type: SYSRSn |
| 7 | BUF_UNDERRUN | R | 0h | Receive Buffer Underrun Flag<br>This bit indicates that a buffer underrun condition has occured in the receive buffer. This will happen when software reads the buffer which is empty and has no valid data. Software can also force this bit to get set by writing to the RX_EVT_FRC register.<br>0h (R) = Receive Buffer Underrun has not ocurred.<br>1h (R) = Receive Buffer Underrun has ocurred.<br>To clear this bit, write to the corresponding bit in the RX_EVT_CLR register.<br>Reset type: SYSRSn |
| 6 | FRAME_DONE | R | 0h | Frame Done Flag<br>This bit indicates that a frame has been successfully received without error. Software can also force this bit to get set by writing to the RX_EVT_FRC register.<br>0h (R) = No frame has been successfully received.<br>1h (R) = A frame has been successfully received.<br>To clear this bit, write to the corresponding bit in the RX_EVT_CLR register.<br>Reset type: SYSRSn |
| 5 | BUF_OVERRUN | R/W | 0h | Receive Buffer Overrun Flag<br>This bit indicates that a buffer overrun condition has occured in the receive buffer. Software can also force this bit to get set by writing to the RX_EVT_FRC register.<br>0h (R) = Receive buffer overrun has not ocurred.<br>1h (R) = Receive buffer overrun has ocurred.<br>To clear this bit, write to the corresponding bit in the RX_EVT_CLR register.<br>Reset type: SYSRSn |
| 4 | EOF_ERR | R | 0h | End-of-Frame Error Flag<br>This bit indicates that an invalid end-of-frame bit pattern has been received. Software can also force this bit to get set by writing to the RX_EVT_FRC register.<br>0h (R) = Invalid end-of-frame has not been received.<br>1h (R) = Invalid end-of-frame has been received<br>To clear this bit, write to the corresponding bit in the RX_EVT_CLR register.<br>Reset type: SYSRSn |

**Table 29-22. RX_EVT_STS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3 | TYPE_ERR | R | 0h | Frame Type Error Flag<br>This bit inditcates that an invalid frame type has been received. Software can also force this bit to get set by writing to the RX_EVT_FRC register.<br>0h (R) = Invalid frame type has not been received.<br>1h (R) = Invalid frame type has been received<br>To clear this bit, write to the corresponding bit in the RX_EVT_CLR register.<br>Reset type: SYSRSn |
| 2 | CRC_ERR | R | 0h | CRC Error Flag<br>This bit indicates that a CRC error has occured. A CRC error will be generated on a data frame where the received CRC and the computed CRC do not match. Software can also force this bit to get set by writing to the RX_EVT_FRC register.<br>0h (R) = CRC error has not occured.<br>1h (R) = CRC error has occured.<br>To clear this bit, write to the corresponding bit in the RX_EVT_CLR register.<br>Reset type: SYSRSn |
| 1 | FRAME_WD_TO | R | 0h | Frame Watchdog Timeout Flag<br>This bit indicates that the frame watchdog timer has timed out. Software can also force this bit to get set by writing to the RX_EVT_FRC register.<br>0h (R) = Frame watchdog timeout has not occured.<br>1h (R) = Frame watchdog timeout has occured.<br>To clear this bit, write to the corresponding bit in the RX_EVT_CLR register.<br>Reset type: SYSRSn |
| 0 | PING_WD_TO | R | 0h | Ping Watchdog Timeout Flag<br>This bit indicates that the ping watchdog timer has timed out. Software can also force this bit to get set by writing to the RX_EVT_FRC register.<br>0h (R) = Ping watchdog timeout has not occured.<br>1h (R) = Ping watchdog timeout has occured.<br>To clear this bit, write to the corresponding bit in the RX_EVT_CLR register.<br>Reset type: SYSRSn |

#### 29.5.1.1.7 RX_CRC_INFO Register (Offset = Bh) [reset = 0h]

RX_CRC_INFO is shown in Figure 29-21 and described in Table 29-23.

Return to Summary Table.

Receive CRC info of received and computed CRC

**Figure 29-21. RX_CRC_INFO Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| CALC_CRC | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RX_CRC | | | | | | | |
| R-0h | | | | | | | |

**Table 29-23. RX_CRC_INFO Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | CALC_CRC | R | 0h | Harware Calculated CRC Value<br>This bitfield contains the CRC value that was calculated on the last received data. The contents of this bitfield are valid only when data frames are received.<br>Note: The contents of this bitfield are invalid for ping and error frames.<br>Reset type: SYSRSn |
| 7-0 | RX_CRC | R | 0h | Received CRC Value<br>This bitfield contains the CRC value that was last received a frame. The contents of this bitfield are valid only when data frames are received.<br>Note: The contents of this bitfield are invalid for ping and error frames.<br>Reset type: SYSRSn |

#### 29.5.1.1.8 RX_EVT_CLR Register (Offset = Ch) [reset = 0h]

RX_EVT_CLR is shown in Figure 29-22 and described in Table 29-24.

Return to Summary Table.

Receive event and error clear register

**Figure 29-22. RX_EVT_CLR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | DATA_FRAME | FRAME_OVERRUN | PING_FRAME | ERR_FRAME |
| R-0h | | | | W-0h | W-0h | W-0h | W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BUF_UNDERRUN | FRAME_DONE | BUF_OVERRUN | EOF_ERR | TYPE_ERR | CRC_ERR | FRAME_WD_TO | PING_WD_TO |
| W-0h | W-0h | W-0h | W-0h | W-0h | W-0h | W-0h | W-0h |

**Table 29-24. RX_EVT_CLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11 | DATA_FRAME | W | 0h | Data Frame Received Flag Clear bit<br>This bit clears the corresponding bit in the RX_EVT_STS register.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0.<br>Reset type: SYSRSn |
| 10 | FRAME_OVERRUN | W | 0h | Frame Overrun Flag Clear bit<br>This bit clears the corresponding bit in the RX_EVT_STS register.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0.<br>Reset type: SYSRSn |
| 9 | PING_FRAME | W | 0h | Ping Frame Received Flag Clear bit<br>This bit clears the corresponding bit in the RX_EVT_STS register.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0.<br>Reset type: SYSRSn |
| 8 | ERR_FRAME | W | 0h | Error Frame Received Flag Clear bit<br>This bit clears the corresponding bit in the RX_EVT_STS register.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0.<br>Reset type: SYSRSn |
| 7 | BUF_UNDERRUN | W | 0h | Receive Buffer Underrun Flag Clear bit<br>This bit clears the corresponding bit in the RX_EVT_STS register.<br>0h (R/W) = Writing a 0 to this bit will have no effect.<br>1h (R/W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0.<br>Reset type: SYSRSn |

**Table 29-24. RX_EVT_CLR Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 6 | FRAME_DONE | W | 0h | Frame Done Flag Clear bit<br>This bit clears the corresponding bit in the RX_EVT_STS register.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0.<br>Reset type: SYSRSn |
| 5 | BUF_OVERRUN | W | 0h | Receive Buffer Overrun Flag Clear bit<br>This bit clears the corresponding bit in the RX_EVT_STS register.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0.<br>Reset type: SYSRSn |
| 4 | EOF_ERR | W | 0h | End-of-Frame Error Flag Clear bit<br>This bit clears the corresponding bit in the RX_EVT_STS register.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0.<br>Reset type: SYSRSn |
| 3 | TYPE_ERR | W | 0h | Frame Type Error Flag Clear bit<br>This bit clears the corresponding bit in the RX_EVT_STS register.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0.<br>Reset type: SYSRSn |
| 2 | CRC_ERR | W | 0h | CRC Error Flag Clear bit<br>This bit clears the corresponding bit in the RX_EVT_STS register.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0.<br>Reset type: SYSRSn |
| 1 | FRAME_WD_TO | W | 0h | Frame Watchdog Timeout Flag Clear bit<br>This bit clears the corresponding bit in the RX_EVT_STS register.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0.<br>Reset type: SYSRSn |
| 0 | PING_WD_TO | W | 0h | Ping Watchdog Timeout Flag Clear bit<br>This bit clears the corresponding bit in the RX_EVT_STS register.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the RX_EVT_STS register to 0.<br>Reset type: SYSRSn |

#### 29.5.1.1.9 RX_EVT_FRC Register (Offset = Dh) [reset = 0h]

RX_EVT_FRC is shown in Figure 29-23 and described in Table 29-25.

Return to Summary Table.

Receive event and error flag force register

**Figure 29-23. RX_EVT_FRC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | DATA_FRAME | FRAME_OVERRUN | PING_FRAME | ERR_FRAME |
| R-0h | | | | W-0h | W-0h | W-0h | W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BUF_UNDERRUN | FRAME_DONE | BUF_OVERRUN | EOF_ERR | TYPE_ERR | CRC_ERR | FRAME_WD_TO | PING_WD_TO |
| W-0h | W-0h | W-0h | W-0h | W-0h | W-0h | W-0h | W-0h |

**Table 29-25. RX_EVT_FRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11 | DATA_FRAME | W | 0h | Data Frame Received Flag Force bit<br>This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Force the corresponding bit in the RX_EVT_STS Register.<br>Reset type: SYSRSn |
| 10 | FRAME_OVERRUN | W | 0h | Frame Overrun Flag Force bit<br>This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Force the corresponding bit in the RX_EVT_STS Register.<br>Reset type: SYSRSn |
| 9 | PING_FRAME | W | 0h | Ping Frame Received Flag Force bit<br>This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Force the corresponding bit in the RX_EVT_STS Register.<br>Reset type: SYSRSn |
| 8 | ERR_FRAME | W | 0h | Error Frame Received Flag Force bit<br>This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Force the corresponding bit in the RX_EVT_STS Register.<br>Reset type: SYSRSn |

**Table 29-25. RX_EVT_FRC Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7 | BUF_UNDERRUN | W | 0h | Receive Buffer Underrun Flag Force bit<br>This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Force the corresponding bit in the RX_EVT_STS Register.<br>Reset type: SYSRSn |
| 6 | FRAME_DONE | W | 0h | Frame Done Flag Force bit<br>This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Force the corresponding bit in the RX_EVT_STS Register.<br>Reset type: SYSRSn |
| 5 | BUF_OVERRUN | W | 0h | Receive Buffer Overrun Flag Force bit<br>This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Force the corresponding bit in the RX_EVT_STS Register.<br>Reset type: SYSRSn |
| 4 | EOF_ERR | W | 0h | End-of-Frame Error Flag Force bit<br>This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Force the corresponding bit in the RX_EVT_STS Register.<br>Reset type: SYSRSn |
| 3 | TYPE_ERR | W | 0h | Frame Type Error Flag Force bit<br>This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Force the corresponding bit in the RX_EVT_STS Register.<br>Reset type: SYSRSn |
| 2 | CRC_ERR | W | 0h | CRC Error Flag Force bit<br>This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Force the corresponding bit in the RX_EVT_STS Register.<br>Reset type: SYSRSn |
| 1 | FRAME_WD_TO | W | 0h | Frame Watchdog Timeout Flag Force bit<br>This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Force the corresponding bit in the RX_EVT_STS Register.<br>Reset type: SYSRSn |

**Table 29-25. RX_EVT_FRC Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | PING_WD_TO | W | 0h | Ping Watchdog Timeout Flag Force bit<br>This bit will cause the corresponding bit in the RX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Force the corresponding bit in the RX_EVT_STS Register.<br>Reset type: SYSRSn |

#### 29.5.1.1.10 RX_BUF_PTR_LOAD Register (Offset = Eh) [reset = 0h]

RX_BUF_PTR_LOAD is shown in Figure 29-24 and described in Table 29-26.

Return to Summary Table.

Receive buffer pointer load register

**Figure 29-24. RX_BUF_PTR_LOAD Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | BUF_PTR_LOAD | | | |
| R-0h | | | | R/W-0h | | | |

**Table 29-26. RX_BUF_PTR_LOAD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-4 | RESERVED | R | 0h | Reserved |
| 3-0 | BUF_PTR_LOAD | R/W | 0h | Buffer Pointer Load.<br>This is the value to be loaded into the receive word pointer when written. This is to allow software to force the receiver to start storing the received data starting at a specific location in the buffer.<br>NOTE: The value of the CURR_BUF_PTR in the RX_BUF_PTR_STS will not get reflected immediately. This will take effect only when there is a valid receive operation with incoming clocks after (3 RXCLK + 3 SYCLK) cycles.<br>Reset type: SYSRSn |

#### 29.5.1.1.11 RX_BUF_PTR_STS Register (Offset = Fh) [reset = 0h]

RX_BUF_PTR_STS is shown in Figure 29-25 and described in Table 29-27.

Return to Summary Table.

Receive buffer pointer status register

**Figure 29-25. RX_BUF_PTR_STS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | CURR_WORD_CNT | | | | |
| R-0h | | | R-0h | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | CURR_BUF_PTR | | | |
| R-0h | | | | R-0h | | | |

**Table 29-27. RX_BUF_PTR_STS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12-8 | CURR_WORD_CNT | R | 0h | Words Available in the Receive Buffer<br>This bitfield indicates the number of valid data words present in the receive buffer that have not been read by the application software. This bitfield is only valid when there is no active transfer.<br>Note: This value will not be valid if there has been a buffer overrun or underrun condition.<br>Reset type: SYSRSn |
| 7-4 | RESERVED | R | 0h | Reserved |
| 3-0 | CURR_BUF_PTR | R | 0h | Current Buffer Pointer Index<br>This bitfield will show the current index of the buffer pointer. This value is only valid when there is no active transmission.<br>Reset type: SYSRSn |

#### 29.5.1.1.12 RX_FRAME_WD_CTRL Register (Offset = 10h) [reset = 0h]

RX_FRAME_WD_CTRL is shown in Figure 29-26 and described in Table 29-28.

Return to Summary Table.

Receive frame watchdog control register

**Figure 29-26. RX_FRAME_WD_CTRL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | FRAME_WD_EN | FRAME_WD_CNT_RST |
| R-0h | | | | | | R/W-0h | R/W-0h |

**Table 29-28. RX_FRAME_WD_CTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-2 | RESERVED | R | 0h | Reserved |
| 1 | FRAME_WD_EN | R/W | 0h | Frame Watchdog Counter Enable bit<br>This bit will enable or disable the frame watchdog counter. The counter (RX_FRAME_WD_CNT) will begin counting from 0 when a valid start-of-frame pattern is received. When the reference value (RX_FRAME_WD_REF) is reached, it will generate a frame watchdog timeout event (RX_EVT_STS.FRAME_WD_TO) and the counter value will reset to 0 and continue counting on the next valid start-of-frame.<br>0h (R/W) = The frame watchdog counter is disabled and not running.<br>1h (R/W) = The frame watchdog counter logic is enabled and running.<br>Reset type: SYSRSn |
| 0 | FRAME_WD_CNT_RST | R/W | 0h | Frame Watchdog Counter Reset bit<br>This bit will reset the frame watchdog counter to 0. This bit will always be read as 0.<br>0h (R/W) = Writing a 0 to this bit has no effect.<br>1h (W) = The frame watchdog counter will be reset to 0.<br>Reset type: SYSRSn |

#### 29.5.1.1.13 RX_FRAME_WD_REF Register (Offset = 12h) [reset = 0h]

RX_FRAME_WD_REF is shown in Figure 29-27 and described in Table 29-29.

Return to Summary Table.

Receive frame watchdog counter reference

**Figure 29-27. RX_FRAME_WD_REF Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FRAME_WD_REF | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 29-29. RX_FRAME_WD_REF Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | FRAME_WD_REF | R/W | 0h | Frame Watchdog Counter Reference Value<br>This is the 32-bit reference value for the frame watchdog timeout counter. The counter will count up starting from 0 at a valid start-of-frame pattern and continue counting until this value is reached.<br>Reset type: SYSRSn |

#### 29.5.1.1.14 RX_FRAME_WD_CNT Register (Offset = 14h) [reset = 0h]

RX_FRAME_WD_CNT is shown in Figure 29-28 and described in Table 29-30.

Return to Summary Table.

Receive frame watchdog current count

**Figure 29-28. RX_FRAME_WD_CNT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FRAME_WD_CNT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 29-30. RX_FRAME_WD_CNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | FRAME_WD_CNT | R | 0h | Frame Watchdog Counter Value<br>This is the 32-bit read-only register which shows the current value of the frame watchdog counter. This counter is reset to 0 in a variety of ways: A write to FRME_WD_CNT_RST, a match with FRAME_WD_REF, or the reception of a successful data frame.<br>Reset type: SYSRSn |

#### 29.5.1.1.15 RX_PING_WD_CTRL Register (Offset = 16h) [reset = 0h]

RX_PING_WD_CTRL is shown in Figure 29-29 and described in Table 29-31.

Return to Summary Table.

Receive ping watchdog control register

**Figure 29-29. RX_PING_WD_CTRL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | PING_WD_EN | PING_WD_RST |
| R-0h | | | | | | R/W-0h | R/W-0h |

**Table 29-31. RX_PING_WD_CTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-2 | RESERVED | R | 0h | Reserved |
| 1 | PING_WD_EN | R/W | 0h | Ping Watchdog Counter Enable bit<br>This bit will enable or disable the ping watchdog counter. The counter (RX_PING_WD_CNT) will begin counting from 0 when it is enabled. When the reference value (RX_PING_WD_REF) is reached, it will generate a ping watchdog timeout event (RX_EVT_STS.PING_WD_TO) and the counter value will reset to 0, and resume counting<br>0h (R/W) = The ping watchdog counter is disabled and not running.<br>1h (R/W) = The ping watchdog counter logic is enabled and running.<br>Reset type: SYSRSn |
| 0 | PING_WD_RST | R/W | 0h | Ping Watchdog Counter Reset bit<br>This bit will reset the ping watchdog counter to 0. This bit will always be read as 0.<br>0h (R/W) = Writing a 0 to this bit has no effect.<br>1h (W) = The ping watchdog counter will be reset to 0.<br>Reset type: SYSRSn |

#### 29.5.1.1.16 RX_PING_TAG Register (Offset = 17h) [reset = 0h]

RX_PING_TAG is shown in Figure 29-30 and described in Table 29-32.

Return to Summary Table.

Receive ping tag register

**Figure 29-30. RX_PING_TAG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | PING_TAG | | | | RESERVED |
| R-0h | | | R-0h | | | | R-0h |

**Table 29-32. RX_PING_TAG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-5 | RESERVED | R | 0h | Reserved |
| 4-1 | PING_TAG | R | 0h | Received Ping Frame Tag<br>This field contains the 4-bit frame tag from the last successfully received ping frame. This is intentionally shifted into bits 4:1 so that the register can be used as a 32-bit address index based on the received tag.<br>Reset type: SYSRSn |
| 0 | RESERVED | R | 0h | Reserved |

#### 29.5.1.1.17 RX_PING_WD_REF Register (Offset = 18h) [reset = 0h]

RX_PING_WD_REF is shown in Figure 29-31 and described in Table 29-33.

Return to Summary Table.

Receive ping watchdog counter reference

**Figure 29-31. RX_PING_WD_REF Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PING_WD_REF | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 29-33. RX_PING_WD_REF Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | PING_WD_REF | R/W | 0h | Ping Watchdog Counter Reference Value<br>This is the 32-bit reference value for the ping watchdog timeout counter. The counter will count up starting from 0 and continue counting until this value is reached.<br>Reset type: SYSRSn |

#### 29.5.1.1.18 RX_PING_WD_CNT Register (Offset = 1Ah) [reset = 0h]

RX_PING_WD_CNT is shown in Figure 29-32 and described in Table 29-34.

Return to Summary Table.

Receive pingwatchdog current count

**Figure 29-32. RX_PING_WD_CNT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PING_WD_CNT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 29-34. RX_PING_WD_CNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | PING_WD_CNT | R | 0h | Ping Watchdog Counter Value<br>This is the 32-bit read-only register which shows the current value of the ping watchdog counter. This counter is reset to 0 in a variety of ways: A write to PING_WD_RST, a match with PING_WD_REF, or the reception of a ping frame.<br>Reset type: SYSRSn |

#### 29.5.1.1.19 RX_INT1_CTRL Register (Offset = 1Ch) [reset = 0h]

RX_INT1_CTRL is shown in Figure 29-33 and described in Table 29-35.

Return to Summary Table.

Receive interrupt control register for RX_INT1

**Figure 29-33. RX_INT1_CTRL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | INT1_EN_DATA_FRAME | INT1_EN_FRAME_OVERRUN | INT1_EN_PING_FRAME | INT1_EN_ERR_FRAME |
| R-0h | | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INT1_EN_UNDERRUN | INT1_EN_FRAME_DONE | INT1_EN_OVERRUN | INT1_EN_EOF_ERR | INT1_EN_TYPE_ERR | INT1_EN_CRC_ERR | INT1_EN_FRAME_WD_TO | INT1_EN_PING_WD_TO |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 29-35. RX_INT1_CTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11 | INT1_EN_DATA_FRAME | R/W | 0h | Enable Data Frame Received Interrupt to INT1 bit<br>This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT1.<br>1h (R/W) = A data frame received event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |
| 10 | INT1_EN_FRAME_OVERRUN | R/W | 0h | Enable Frame Overrun Interrupt to INT1 bit<br>This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT1.<br>1h (R/W) = A frame overrun event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |
| 9 | INT1_EN_PING_FRAME | R/W | 0h | Enable Ping Frame Received Interrupt to INT1 bit<br>This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT1.<br>1h (R/W) = A ping frame received event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |
| 8 | INT1_EN_ERR_FRAME | R/W | 0h | Enable ERROR Frame Received Interrupt to INT1 bit<br>This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT1.<br>1h (R/W) = A error frame received event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |

**Table 29-35. RX_INT1_CTRL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7 | INT1_EN_UNDERRUN | R/W | 0h | Enable Buffer Underrun Interrupt to INT1 bit<br>This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT1.<br>1h (R/W) = A buffer underrun event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |
| 6 | INT1_EN_FRAME_DONE | R/W | 0h | Enable Frame Done Interrupt to INT1 bit<br>This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT1.<br>1h (R/W) = A frame done event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |
| 5 | INT1_EN_OVERRUN | R/W | 0h | Enable Receive Buffer Overrun Interrupt to INT1 bit<br>This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT1.<br>1h (R/W) = A receive buffer overrun event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |
| 4 | INT1_EN_EOF_ERR | R/W | 0h | Enable End-of-Frame Error Interrupt to INT1 bit<br>This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT1.<br>1h (R/W) = An end-of-frame error event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |
| 3 | INT1_EN_TYPE_ERR | R/W | 0h | Enable Frame Type Error Interrupt to INT1 bit<br>This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT1.<br>1h (R/W) = A frame type error event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |
| 2 | INT1_EN_CRC_ERR | R/W | 0h | Enable CRC Error Interrupt to INT1 bit<br>This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT1.<br>1h (R/W) = A CRC error will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |

**Table 29-35. RX_INT1_CTRL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | INT1_EN_FRAME_WD_TO | R/W | 0h | Enable Frame Watchdog Timeout Interrupt to INT1 bit<br>This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT1.<br>1h (R/W) = A frame watchdog timeout event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |
| 0 | INT1_EN_PING_WD_TO | R/W | 0h | Enable Ping Watchdog Timeout Interrupt to INT1 bit<br>This is an enable register which decides whether an interrupt (RX_INT1) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT1.<br>1h (R/W) = A ping watchdog timeout event will trigger an interrupt on RX_INT1. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |

#### 29.5.1.1.20 RX_INT2_CTRL Register (Offset = 1Dh) [reset = 0h]

RX_INT2_CTRL is shown in Figure 29-34 and described in Table 29-36.

Return to Summary Table.

Receive interrupt control register for RX_INT2

**Figure 29-34. RX_INT2_CTRL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | INT2_EN_DATA_FRAME | INT2_EN_FRAME_OVERRUN | INT2_EN_PING_FRAME | INT2_EN_ERR_FRAME |
| R-0h | | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INT2_EN_UNDERRUN | INT2_EN_FRAME_DONE | INT2_EN_OVERRUN | INT2_EN_EOF_ERR | INT2_EN_TYPE_ERR | INT2_EN_CRC_ERR | INT2_EN_FRAME_WD_TO | INT2_EN_PING_WD_TO |
| R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 29-36. RX_INT2_CTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11 | INT2_EN_DATA_FRAME | R/W | 0h | Enable Data Frame Received Interrupt to INT2 bit<br>This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT2.<br>1h (R/W) = A data frame received event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |
| 10 | INT2_EN_FRAME_OVERRUN | R/W | 0h | Enable Frame Overrun Interrupt to INT2 bit<br>This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT2.<br>1h (R/W) = A frame overrun event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |
| 9 | INT2_EN_PING_FRAME | R/W | 0h | Enable Ping Frame Received Interrupt to INT2 bit<br>This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT2.<br>1h (R/W) = A ping frame received event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |
| 8 | INT2_EN_ERR_FRAME | R/W | 0h | Enable Error Frame Received Interrupt to INT2 bit<br>This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT2.<br>1h (R/W) = A error frame received event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |

**Table 29-36. RX_INT2_CTRL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 7 | INT2_EN_UNDERRUN | R/W | 0h | Enable Buffer Underrun Interrupt to INT2 bit<br>This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT2.<br>1h (R/W) = A buffer underrun event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |
| 6 | INT2_EN_FRAME_DONE | R/W | 0h | Enable Frame Done Interrupt to INT2 bit<br>This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT2.<br>1h (R/W) = A frame done event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |
| 5 | INT2_EN_OVERRUN | R/W | 0h | Enable Buffer Overrun Interrupt to INT2 bit<br>This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT2.<br>1h (R/W) = A buffer overrun event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |
| 4 | INT2_EN_EOF_ERR | R/W | 0h | Enable End-of-Frame Error Interrupt to INT2 bit<br>This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT2.<br>1h (R/W) = An end-of-frame error event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |
| 3 | INT2_EN_TYPE_ERR | R/W | 0h | Enable Frame Type Error Interrupt to INT2 bit<br>This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT2.<br>1h (R/W) = A frame type error event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |
| 2 | INT2_EN_CRC_ERR | R/W | 0h | Enable CRC Error Interrupt to INT2 bit<br>This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT2.<br>1h (R/W) = A CRC error will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |

**Table 29-36. RX_INT2_CTRL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 1 | INT2_EN_FRAME_WD_TO | R/W | 0h | Enable Frame Watchdog Timeout Interrupt to INT2 bit<br>This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT2.<br>1h (R/W) = A frame watchdog timeout event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |
| 0 | INT2_EN_PING_WD_TO | R/W | 0h | Enable Ping Watchdog Timeout Interrupt to INT2 bit<br>This is an enable register which decides whether an interrupt (RX_INT2) will be generated on the enabled event.<br>0h (R/W) = This event will not trigger an interrupt on RX_INT2.<br>1h (R/W) = A ping watchdog timeout event will trigger an interrupt on RX_INT2. The event itself will be latched in the corresponding bit in the RX_EVT_STS Register<br>Reset type: SYSRSn |

#### 29.5.1.1.21 RX_LOCK_CTRL Register (Offset = 1Eh) [reset = 0h]

RX_LOCK_CTRL is shown in Figure 29-35 and described in Table 29-37.

Return to Summary Table.

Receive lock control register

**Figure 29-35. RX_LOCK_CTRL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| KEY | | | | | | | |
| W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | LOCK |
| R-0h | | | | | | | R/W-0h |

**Table 29-37. RX_LOCK_CTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | KEY | W | 0h | Write Key.<br>In order to write to this register, 0xA5 must be written to this field at the same time. Otherwise, writes are ignored. The key is cleared immediately after writing, so it must be written again for every change to this register.<br>Reset type: SYSRSn |
| 7-1 | RESERVED | R | 0h | Reserved |
| 0 | LOCK | R/W | 0h | Control Register Lock Enable bit<br>This bit locks the contents of all the receive control registers that support a lock protection. Once locked, further writes will not take effect until SYSRS unlocks the register. Once set, further writes even to this bit will be ignored.<br>0h (R/W) = Receive control registers can be modified and are not locked.<br>1h (R/W) = Receive control registers are locked and cannot be modified until this bit is cleared by SYSRS. Any further writes to this bit are ignored.<br>Note: The KEY field must contatin 0xA5 for any write to this bit to take effect.<br>Reset type: SYSRSn |

#### 29.5.1.1.22 RX_ECC_DATA Register (Offset = 20h) [reset = 0h]

RX_ECC_DATA is shown in Figure 29-36 and described in Table 29-38.

Return to Summary Table.

Receive ECC data register

**Figure 29-36. RX_ECC_DATA Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA_HIGH | | | | | | | | | | | | | | | | DATA_LOW | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | R/W-0h | | | | | | | | | | | | | | | |

**Table 29-38. RX_ECC_DATA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | DATA_HIGH | R/W | 0h | Upper 16 bits of ECC Data<br>Writing to this bitfield will cause the ECC logic to compute the ECC(SEC-DED) the entire 32-bit register and update TX_ECC_VAL register with the results. Software should write to these 16 bits of the register in a 32-bit write when needing to compute ECC for 32-bits for the full TX_ECC_DATA register.<br>Reset type: SYSRSn |
| 15-0 | DATA_LOW | R/W | 0h | Lower 16 bits of ECC Data<br>Writing to this bitfield will cause the ECC logic to compute the ECC(SEC-DED) for these 16 bits and update the TX_ECC_VAL register with the results. Software should write to these register bits as a 16-bit write when needing to compute ECC for 16-bits.<br>Reset type: SYSRSn |

#### 29.5.1.1.23 RX_ECC_VAL Register (Offset = 22h) [reset = 0h]

RX_ECC_VAL is shown in Figure 29-37 and described in Table 29-39.

Return to Summary Table.

Receive ECC value register

**Figure 29-37. RX_ECC_VAL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ECC_VAL | | | | | | | |
| R-0h | | | | | | | |

**Table 29-39. RX_ECC_VAL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-0 | ECC_VAL | R | 0h | Computed ECC Value<br>This field contains the ECC value computed using SEC-DED either for 16-bit or 32-bit data in the TX_ECC_DATA register.<br>Reset type: SYSRSn |

#### 29.5.1.1.24 RX_ECC_SEC_DATA Register (Offset = 24h) [reset = 0h]

RX_ECC_SEC_DATA is shown in Figure 29-38 and described in Table 29-40.

Return to Summary Table.

Receive ECC corrected data register

**Figure 29-38. RX_ECC_SEC_DATA Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SEC_DATA | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 29-40. RX_ECC_SEC_DATA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | SEC_DATA | R | 0h | ECC Single Error Corrected Data<br>The ECC corrected data will be available in this register. This value is valid only when there are no bit errors, or a single bit error was detected. Otherwise, the contents of this register are invalid and should not be used.<br>Reset type: SYSRSn |

#### 29.5.1.1.25 RX_ECC_LOG Register (Offset = 26h) [reset = 3h]

RX_ECC_LOG is shown in Figure 29-39 and described in Table 29-41.

Return to Summary Table.

Receive ECC log and status register

**Figure 29-39. RX_ECC_LOG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | MBE | SBE |
| R-0h | | | | | | R-1h | R-1h |

**Table 29-41. RX_ECC_LOG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-2 | RESERVED | R | 0h | Reserved |
| 1 | MBE | R | 1h | Multiple Bit Errors Detected<br>This bit indicates the occurrence of multiple bit errors.The data is corrupted and cannot be corrected. If this bit is set, the data present in RX_ECC_SEC_DATA is invalid and should not be used.<br>0h (R) Multiple Bit Errors were not detected. Check the SBE bit for single bit errors.<br>1h (R) Multiple Bit Errors were detected. The data is not able to be corrected. The value present in RX_ECC_SEC_DATA is invalid and should not be used.<br>Reset type: SYSRSn |
| 0 | SBE | R | 1h | Single Bit Error Detected<br>This bit indicates the occurrence of a single bit error in the data. The data is autocorrected and placed into the RX_ECC_SEC_DATA register. This bit is valid only if MBE is 0.<br>0h (R) No bit errors were detected. The value in RX_ECC_SEC_DATA is correct.<br>1h (R) A single bit error was detected and corrected. The corrected data is present in RX_ECC_SEC_DATA.<br>Reset type: SYSRSn |

#### 29.5.1.1.26 RX_DLYLINE_CTRL Register (Offset = 30h) [reset = 0h]

RX_DLYLINE_CTRL is shown in Figure 29-40 and described in Table 29-42.

Return to Summary Table.

Receive delay line control register

**Figure 29-40. RX_DLYLINE_CTRL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | RXD1_DLY | | | | | RXD0_DLY | |
| R-0h | R/W-0h | | | | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RXD0_DLY | | | RXCLK_DLY | | | | |
| R/W-0h | | | R/W-0h | | | | |

**Table 29-42. RX_DLYLINE_CTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | RESERVED | R | 0h | Reserved |
| 14-10 | RXD1_DLY | R/W | 0h | Delay Line Tap Select for RXD1<br>This bitfield selects the number of delay elements inserted into the RXD1 path from the pin boundary to the receiver core.<br>0h (R/W) Zero delay elements are included in the RXD1 path. RXD1 is taken directly from the pin.<br>1h (R/W) One delay element is included in the RXD1 path.<br>2h (R/W) Two delay elements are included in the RXD1 path.<br>...<br>1Fh (R/W) 31 delay elements are included in the RXD1 path, the maximum.<br>Reset type: SYSRSn |
| 9-5 | RXD0_DLY | R/W | 0h | Delay Line Tap Select for RXD0<br>This bitfield selects the number of delay elements inserted into the RXD0 path from the pin boundary to the receiver core.<br>0h (R/W) Zero delay elements are included in the RXD0 path. RXD0 is taken directly from the pin.<br>1h (R/W) One delay element is included in the RXD0 path.<br>2h (R/W) Two delay elements are included in the RXD0 path.<br>...<br>1Fh (R/W) 31 delay elements are included in the RXD0 path, the maximum.<br>Reset type: SYSRSn |
| 4-0 | RXCLK_DLY | R/W | 0h | Delay Line Tap Select for RXCLK<br>This bitfield selects the number of delay elements inserted into the RXCLK path from the pin boundary to the receiver core.<br>0h (R/W) Zero delay elements are included in the RXCLK path. RXCLK is taken directly from the pin.<br>1h (R/W) One delay element is included in the RXCLK path.<br>2h (R/W) Two delay elements are included in the RXCLK path.<br>...<br>1Fh (R/W) 31 delay elements are included in the RXCLK path, the maximum.<br>Reset type: SYSRSn |

#### 29.5.1.1.27 RX_VIS_1 Register (Offset = 38h) [reset = 0h]

RX_VIS_1 is shown in Figure 29-41 and described in Table 29-43.

Return to Summary Table.

Receive debug visibility register 1

**Figure 29-41. RX_VIS_1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | RX_CORE_STS | RESERVED | | |
| R-0h | | | | R-0h | R-0h | | |

**Table 29-43. RX_VIS_1 Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-4 | RESERVED | R | 0h | Reserved |
| 3 | RX_CORE_STS | R | 0h | Receiver Core Status bit<br>This bit indicates the status of the receiver core. If this bit is set, the receiver should undergo a reset and subsequent resynchronization with the transmitter. This bit will be always be set when the receiver has detected and end of frame error or a frame type error. This bit can also be set if the receiver becomes corrupted due to noise on the signal lines. If the receiver has experienced a ping watchdog or frame watchdog timeout, this bit should be read to determine if the cause was due to a corrupt transaction, thus putting the receiver core into an unrecoverable state.<br>Only a soft reset will reset the recevier core and thus reset this bit.<br>0h (R) The receiver core is operating normally.<br>1h (R) The receiver core has entered into an error state and should be reset.<br>Reset type: SYSRSn |
| 2-0 | RESERVED | R | 0h | Reserved |

#### 29.5.1.1.28 RX_BUF_BASE Register (Offset = 40h) [reset = 0h]

RX_BUF_BASE is shown in Figure 29-42 and described in Table 29-44.

Return to Summary Table.

Base address for receive data buffer

**Figure 29-42. RX_BUF_BASE Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BASE_ADDRESS | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | |

**Table 29-44. RX_BUF_BASE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | BASE_ADDRESS | R/W | 0h | Receive Data Buffer Base Address<br>This is the base address of the 16-word data buffer used by the receiver.<br>Reset type: SYSRSn |

### 29.5.1.2 FSI_TX_REGS Registers

Table 29-45 lists the memory-mapped registers for the FSI_TX_REGS. All register offset addresses not listed in Table 29-45 should be considered as reserved locations and the register contents should not be modified.

**Table 29-45. FSI_TX_REGS Registers**

| Offset | Acronym | Register Name | Write Protection | Section |
|---|---|---|---|---|
| 0h | TX_MASTER_CTRL | Transmit master control register | EALLOW | Go |
| 2h | TX_CLK_CTRL | Transmit clock control register | EALLOW and LOCK | Go |
| 4h | TX_OPER_CTRL_LO | Transmit operation control register low | EALLOW and LOCK | Go |
| 5h | TX_OPER_CTRL_HI | Transmit operation control register high | EALLOW and LOCK | Go |
| 6h | TX_FRAME_CTRL | Transmit frame control register | | Go |
| 7h | TX_FRAME_TAG_UDATA | Transmit frame tag and user data register | | Go |
| 8h | TX_BUF_PTR_LOAD | Transmit buffer pointer control load register | EALLOW | Go |
| 9h | TX_BUF_PTR_STS | Transmit buffer pointer control status register | | Go |
| Ah | TX_PING_CTRL | Transmit ping control register | EALLOW and LOCK | Go |
| Bh | TX_PING_TAG | Transmit ping tag register | | Go |
| Ch | TX_PING_TO_REF | Transmit ping timeout counter reference | YES | Go |
| Eh | TX_PING_TO_CNT | Transmit ping timeout current count | | Go |
| 10h | TX_INT_CTRL | Transmit interrupt event control register | EALLOW and LOCK | Go |
| 11h | TX_DMA_CTRL | Transmit DMA event control register | EALLOW and LOCK | Go |
| 12h | TX_LOCK_CTRL | Transmit lock control register | EALLOW and LOCK | Go |
| 14h | TX_EVT_STS | Transmit event and error status flag register | | Go |
| 16h | TX_EVT_CLR | Transmit event and error clear register | EALLOW | Go |
| 17h | TX_EVT_FRC | Transmit event and error flag force register | EALLOW | Go |
| 18h | TX_USER_CRC | Transmit user-defined CRC register | | Go |
| 20h | TX_ECC_DATA | Transmit ECC data register | | Go |
| 22h | TX_ECC_VAL | Transmit ECC value register | | Go |
| 40h | TX_BUF_BASE | Base address for transmit buffer | | Go |

Complex bit access types are encoded to fit into small table cells. Table 29-46 shows the codes that are used for access types in this section.

**Table 29-46. FSI_TX_REGS Access Type Codes**

| Access Type | Code | Description |
|---|---|---|
| **Read Type** | | |
| R | R | Read |
| RX | R<br>X | Read |
| **Write Type** | | |
| W | W | Write |
| **Reset or Default Value** | | |
| *-n* | | Value after reset or the default value |
| **Register Array Variables** | | |

**Table 29-46. FSI_TX_REGS Access Type Codes (continued)**

| Access Type | Code | Description |
|---|---|---|
| i,j,k,l,m,n | | When these variables are used in a register name, an offset, or an address, they refer to the value of a register array where the register is part of a group of repeating registers. The register groups form a hierarchical structure and the array is represented with a formula. |
| y | | When this variable is used in a register name, an offset, or an address it refers to the value of a register array. |

#### 29.5.1.2.1 TX_MASTER_CTRL Register (Offset = 0h) [reset = 0h]

TX_MASTER_CTRL is shown in Figure 29-43 and described in Table 29-47.

Return to Summary Table.

Transmit master control register

**Figure 29-43. TX_MASTER_CTRL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| KEY | | | | | | | |
| W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | FLUSH | CORE_RST |
| R-0h | | | | | | R/W-0h | R/W-0h |

**Table 29-47. TX_MASTER_CTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | KEY | W | 0h | Write Key<br>In order to write to any bit in this register, 0xA5 must be written to this field at the same time. Otherwise, writes are ignored. The key is cleared immediately after writing, so it must be written again for every change to this register.<br>Reset type: SYSRSn |
| 7-2 | RESERVED | R | 0h | Reserved |
| 1 | FLUSH | R/W | 0h | Flush Operation Start bit<br>This bit will cause the transmitter to initiate a flush pattern of a single toggle on the TXD0 and TXD1 followed by five full cycles of TXCLK. This bit should be written only when the CORE_RST bit is 0 and the clock to the Transmitter core is turned on.<br>0h (R/W) = Clear this bit.<br>1h (R/W) = Setting this bit will Initiate flush sequence.<br>To properly execute a flush sequence, Set FLUSH to 1, wait for five TXCLK cycles then clear FLUSH to 0.<br>Note: The KEY field must contain 0xA5 for any write to this bit to take effect. The software must keep this bit set to 1 for at least five TXCLK cycles before setting it back to 0.<br>Reset type: SYSRSn |
| 0 | CORE_RST | R/W | 0h | Transmitter Master Core Reset bit<br>This bit controls the transmitter master core reset. In order to send any frame, this bit must be cleared.<br>0h (R/W) = Transmitter core is not in reset and can transmit frames.<br>1h (R/W) = Transmitter core is held in reset.<br>Note: The KEY field must contatin 0xA5 for any write to this bit to take effect.<br>Reset type: SYSRSn |

#### 29.5.1.2.2 TX_CLK_CTRL Register (Offset = 2h) [reset = 0h]

TX_CLK_CTRL is shown in Figure 29-44 and described in Table 29-48.

Return to Summary Table.

Transmit clock control register

**Figure 29-44. TX_CLK_CTRL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | PRESCALE_VAL | |
| R-0h | | | | | | R/W-0h | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PRESCALE_VAL | | | | | | CLK_EN | CLK_RST |
| R/W-0h | | | | | | R/W-0h | R/W-0h |

**Table 29-48. TX_CLK_CTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-10 | RESERVED | R | 0h | Reserved |
| 9-2 | PRESCALE_VAL | R/W | 0h | Clock Divider Prescale Value<br>The input clock is divided by this 8-bit value and fed into the transmitter core. This divided clock is the rate at which TXCLK will operate.<br>0h (R/W) = Input clock /1<br>1h (R/W) = Input clock /1<br>2h (R/W) = Input clock /2<br>3h (R/W) = Input clock /3<br>4h (R/W) = Input clock /4<br>...<br>FFh (R/W) = Input clock /255<br>Reset type: SYSRSn |
| 1 | CLK_EN | R/W | 0h | Clock Divider Enable bit<br>This bit will enable and disable the input clock divider and start the clock to the transmitter core.<br>0h (R/W) = The input clock divider is not enabled and the clock is not connected to the transmitter core.<br>1h (R/W) = The input clock to the transmitter core is being divided by the PRESCALE_VAL and enabled.<br>Reset type: SYSRSn |
| 0 | CLK_RST | R/W | 0h | Clock Divider Reset bit<br>This bit will reset the clock counter in the clock divider.<br>0h (R/W) = The clock divider is set based on the value in PRESCALE_VAL. The input clock will be divided by PRESCALE_VAL if CLK_EN is set.<br>1h (R/W) = The clock divider will be reset to 0 and will stay reset until software writes a 0 to this bit.<br>Reset type: SYSRSn |

#### 29.5.1.2.3 TX_OPER_CTRL_LO Register (Offset = 4h) [reset = 0h]

TX_OPER_CTRL_LO is shown in Figure 29-45 and described in Table 29-49.

Return to Summary Table.

Transmit operation control register low

**Figure 29-45. TX_OPER_CTRL_LO Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | SEL_PLLCLK |
| R-0h | | | | | | | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PING_TO_MODE | SW_CRC | START_MODE | | | SPI_MODE | DATA_WIDTH | |
| R/W-0h | R/W-0h | R/W-0h | | | R/W-0h | R/W-0h | |

**Table 29-49. TX_OPER_CTRL_LO Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-9 | RESERVED | R | 0h | Reserved |
| 8 | SEL_PLLCLK | R/W | 0h | Input Clock Select bit<br>This bit selects the input clock source for the transmitter core.<br>0h (R/W) = SYSCLK is the source of the transmitter clock into the clock prescaler.<br>1h (R/W) = PLLRAWCLK is the source of the transmitter core clock into the clock prescaler.<br>Reset type: SYSRSn |
| 7 | PING_TO_MODE | R/W | 0h | Ping Counter Reset Mode Select bit<br>This bit selects when the ping counter will reset.<br>0h (R/W) = The ping counter will reset and restart only on hardware initiated ping frames, when ping counter has timed out.<br>1h (R/W) = The ping counter will reset and restart on any software initiated frame as well as a ping counter timeout<br>Reset type: SYSRSn |
| 6 | SW_CRC | R/W | 0h | CRC Source Select bit<br>This bit selects the source of the CRC value that is transmitted.<br>0h (R/W) = The transmitted CRC value is computed by hardware.<br>1h (R/W) = The transmitted CRC value is sourced from the value programmed in the TX_USER_CRC register.<br>Reset type: SYSRSn |
| 5-3 | START_MODE | R/W | 0h | Transmission Start Mode Select bit<br>These bits select the method by which a new frame transmission is started.<br>0h (R/W) = Only a software write to TX_FRAME_CTRL.START initiate a new transmission.<br>1h (R/W) = The configured external trigger will initiate a new transmission.<br>2h (R/W) = Either writing to TX_FRAME_CTRL.START or the TX_FRAME_TAG_UDATA register will initiate a new transmission.<br>All other combinations of bits are illegal and reserved for future use.<br>Reset type: SYSRSn |

**Table 29-49. TX_OPER_CTRL_LO Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | SPI_MODE | R/W | 0h | SPI Mode Select bit<br>This bit enables and disables SPI compatibility mode.<br>0h (R/W) = FSI is in normal mode of operation.<br>1h (R/W) = FSI is operating in SPI compatibility mode.<br>Reset type: SYSRSn |
| 1-0 | DATA_WIDTH | R/W | 0h | Transmit Data Width Select bits<br>These bits define the number of data lines used by the transmitter.<br>0h (R/W) = Data will be transmitted on one data line (TXD0)<br>1h (R/W) = Data will be transmitted on two data lines (TXD0 and TXD1). The format of the data is described in the preceeding chapter.<br>2h, 3h (R/W) = Reserved<br>Reset type: SYSRSn |

#### 29.5.1.2.4 TX_OPER_CTRL_HI Register (Offset = 5h) [reset = 0h]

TX_OPER_CTRL_HI is shown in Figure 29-46 and described in Table 29-50.

Return to Summary Table.

Transmit operation control register high

**Figure 29-46. TX_OPER_CTRL_HI Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | ECC_SEL | FORCE_ERR | EXT_TRIG_SEL | | | | |
| R-0h | R/W-0h | R/W-0h | R/W-0h | | | | |

**Table 29-50. TX_OPER_CTRL_HI Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-7 | RESERVED | R | 0h | Reserved |
| 6 | ECC_SEL | R/W | 0h | ECC Data Width Select bit<br>This bit selects between 16-bit and 32-bit ECC computation.<br>0h (R/W) = 32-bit ECC is used.<br>1h (R/W) = 16-bit ECC is used.<br>Reset type: SYSRSn |
| 5 | FORCE_ERR | R/W | 0h | Error Frame Force bit<br>This bit will force the the CRC value of the transmitted data frame to 0 whenever there is a buffer overrun or underrun condition. This can be used to force a corrupted CRC as the data is not guaranteed to be reliable. The receiver will treat the data as invalid and can handle this as needed.<br>0h (R/W) = The CRC will not be forced to 0.<br>1h (R/W) = The CRC will be forced to 0 in a buffer overrun or underrun condition.<br>Reset type: SYSRSn |
| 4-0 | EXT_TRIG_SEL | R/W | 0h | External Trigger Select bit<br>These bits define which of the 32 external inputs will be used as the source for the external input trigger.<br>00h (R/W) = Trigger 1 is the source.<br>01h (R/W) = Trigger 2 is the source.<br>02h (R/W) = Trigger 3 is the source.<br>...<br>1Fh (R/W) = Trigger 32 is the source.<br>Reset type: SYSRSn |

#### 29.5.1.2.5 TX_FRAME_CTRL Register (Offset = 6h) [reset = 0h]

TX_FRAME_CTRL is shown in Figure 29-47 and described in Table 29-51.

Return to Summary Table.

Transmit frame control register

**Figure 29-47. TX_FRAME_CTRL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| START | RESERVED | | | | | | |
| R/W-0h | R-0h | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| N_WORDS | | | | FRAME_TYPE | | | |
| R/W-0h | | | | R/W-0h | | | |

**Table 29-51. TX_FRAME_CTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15 | START | R/W | 0h | Start Transmission bit<br>This bit will cause the FSI to start transmitting the next frame.<br>0h (R/W) = Writing a 0 to this bit will have no effect.<br>1h (R/W) = Start the next transmission. This bit will be cleared by hardware.<br>Reset type: SYSRSn |
| 14-8 | RESERVED | R | 0h | Reserved |
| 7-4 | N_WORDS | R/W | 0h | Number of Words to be Transmitted<br>This field defines the number of words which will be transmitted in a DATA_N_WORD frame. This is a user-defined field that must match the corresponding field in the receiver. Set this bitfield to be one greater than the number of words to be transmitted.<br>0h (R/W) = 1 data word frame (16-bit data).<br>1h (R/W) = 2 data word frame (32-bit data).<br>..<br>Fh (R/W) = 16 data word frame (256-bit data).<br>Reset type: SYSRSn |

**Table 29-51. TX_FRAME_CTRL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 3-0 | FRAME_TYPE | R/W | 0h | Transmit Frame Type<br>This field determines the type of frame that will be transmitted next.<br>0000b (R/W) = Ping Frame. This frame can be sent either by software or automatically by hardware.<br>0100b (R/W) = DATA_1_WORD Frame. One word data frame (16-bit data).<br>0101b (R/W) = DATA_2_WORD Frame. Two word data frame (32-bit data).<br>0110b (R/W) = DATA_4_WORD Frame. Four word data frame (64-bit data).<br>0111b (R/W) = DATA_6_WORD Frame. Six word data frame (96-bit data).<br>0011b (R/W) = DATA_N_WORD Frame. The N_WORDS field will determine the number of words (1 to 16) to be sent. Both the transmitter and receiver must have the same value programmed.<br>1111b (R/W) = Error Frame. This frame can be used during error conditions or any condition where the transmitter wants to notify the receiver of a high priorty status. However, the user software is at liberty to use this for any purpose.<br>0001b, 0010b, and 1000b through 1110b are Reserved and should not be used.<br>Reset type: SYSRSn |

#### 29.5.1.2.6 TX_FRAME_TAG_UDATA Register (Offset = 7h) [reset = 0h]

TX_FRAME_TAG_UDATA is shown in Figure 29-48 and described in Table 29-52.

Return to Summary Table.

Transmit frame tag and user data register

**Figure 29-48. TX_FRAME_TAG_UDATA Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| USER_DATA | | | | | | | |
| R/W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | FRAME_TAG | | | |
| R-0h | | | | R/W-0h | | | |

**Table 29-52. TX_FRAME_TAG_UDATA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | USER_DATA | R/W | 0h | User Data bits<br>This is a user-defined value that will be loaded into the the user data phase of the frame. This 8-bit value can be used by the receiver for any application need. This value will not impact any hardware behavior.<br>Reset type: SYSRSn |
| 7-4 | RESERVED | R | 0h | Reserved |
| 3-0 | FRAME_TAG | R/W | 0h | Frame tag bits<br>This is a user-defined value that will be loaded into the frame tag phase of the next transmission. The receiver may use the frame tag for any application need. This value will not impact any hardware behavior<br>Reset type: SYSRSn |

#### 29.5.1.2.7 TX_BUF_PTR_LOAD Register (Offset = 8h) [reset = 0h]

TX_BUF_PTR_LOAD is shown in Figure 29-49 and described in Table 29-53.

Return to Summary Table.

Transmit buffer pointer control load register

**Figure 29-49. TX_BUF_PTR_LOAD Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | BUF_PTR_LOAD | | | |
| R-0h | | | | R/W-0h | | | |

**Table 29-53. TX_BUF_PTR_LOAD Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-4 | RESERVED | R | 0h | Reserved |
| 3-0 | BUF_PTR_LOAD | R/W | 0h | Buffer Pointer Load bits<br>These bits are used to force the transmit buffer pointer to a desired index within the transmit buffer. The next transmission will begin picking data from this index and increment appropriately. This value will be reflected in TX_BUF_PTR_STS only after a minimum 3 SYSCLK cycles + 3 TXCLK cycles.<br>This value should not be written while there is an active transmission as it may corrupt the ongoing frame or other undefined behavior.<br>Reset type: SYSRSn |

#### 29.5.1.2.8 TX_BUF_PTR_STS Register (Offset = 9h) [reset = 0h]

TX_BUF_PTR_STS is shown in Figure 29-50 and described in Table 29-54.

Return to Summary Table.

Transmit buffer pointer control status register

**Figure 29-50. TX_BUF_PTR_STS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | CURR_WORD_CNT | | | | |
| R-0h | | | R-0h | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | CURR_BUF_PTR | | | |
| R-0h | | | | R-0h | | | |

**Table 29-54. TX_BUF_PTR_STS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-13 | RESERVED | R | 0h | Reserved |
| 12-8 | CURR_WORD_CNT | R | 0h | Words Remaining in the transmit buffer<br>This value indicates the number of words present in the data buffer which have not yet been transmitted. This value is only valid when there is no active transmission.<br>Note: This value will not be valid if there is a buffer overrun or underrun condition.<br>Reset type: SYSRSn |
| 7-4 | RESERVED | R | 0h | Reserved |
| 3-0 | CURR_BUF_PTR | R | 0h | Current Buffer Pointer Index<br>This bitfield will show the current index of the buffer pointer. This value is only valid when there is no active transmission.<br>Reset type: SYSRSn |

#### 29.5.1.2.9 TX_PING_CTRL Register (Offset = Ah) [reset = 0h]

TX_PING_CTRL is shown in Figure 29-51 and described in Table 29-55.

Return to Summary Table.

Transmit ping control register

**Figure 29-51. TX_PING_CTRL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EXT_TRIG_SEL | | | | | EXT_TRIG_EN | TIMER_EN | CNT_RST |
| R/W-0h | | | | | R/W-0h | R/W-0h | R/W-0h |

**Table 29-55. TX_PING_CTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-3 | EXT_TRIG_SEL | R/W | 0h | External Trigger Select bits<br>This bitfield will select one of the 32 external trigger inputs to as the source to generate a ping frame. A ping frame will only be generated if the EXT_TRIG_EN bit is set.<br>0h (R/W) = Trigger 1 will be used to generate a ping frame.<br>1h (R/W) = Trigger 2 will be used to generate a ping frame.<br>..<br>1Fh (R/W) = Trigger 32 will be used to generate a ping frame.<br>Reset type: SYSRSn |
| 2 | EXT_TRIG_EN | R/W | 0h | External Trigger Enable bit<br>This bit will allow the external trigger logic to generate a ping frame.<br>0h (R/W) = External triggers will not be used to generate ping frames.<br>1h (R/W) = The selected external trigger (selected by EXT_TRIG_SEL bits) will be able to generate a ping frame. The ping timer will be ignored if this bit is set.<br>Reset type: SYSRSn |
| 1 | TIMER_EN | R/W | 0h | Ping Timer Enable bit<br>This bit will enable the ping timer for generating periodic ping frames.<br>0h (R/W) = The ping timer is disabled and will not generate ping frames.<br>1h (R/W) = The ping timer is enabled and can be used to generate ping frames.Once the timer count reaches the value set by the TX_PING_TO_REF register, it will initiate a ping frame transmission.<br>Note: If the ping timer is used, EXT_TRIG_EN should not be set as it will override this function.<br>Reset type: SYSRSn |
| 0 | CNT_RST | R/W | 0h | Ping Counter Reset bit<br>This bit will reset the the ping counter to 0. This bit will always be read as 0.<br>0h (R/W) = Writing a 0 to this bit has no effect.<br>1h (R/W) = The ping counter will be reset to 0.<br>Reset type: SYSRSn |

#### 29.5.1.2.10 TX_PING_TAG Register (Offset = Bh) [reset = 0h]

TX_PING_TAG is shown in Figure 29-52 and described in Table 29-56.

Return to Summary Table.

Transmit ping tag register

**Figure 29-52. TX_PING_TAG Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | TAG | | | |
| R-0h | | | | R/W-0h | | | |

**Table 29-56. TX_PING_TAG Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-4 | RESERVED | R | 0h | Reserved |
| 3-0 | TAG | R/W | 0h | Ping Frame Tag<br>This field contains a 4-bit tag which will be sent in any ping frame that is initiated by an external trigger or the ping timer. This field is user-defined and can be set based on the application requirement. If a ping frame is generated manually, the transmitted tag will be from TX_FRAME_TAG_UDATA.FRAME_TAG, not this value.<br>Reset type: SYSRSn |

#### 29.5.1.2.11 TX_PING_TO_REF Register (Offset = Ch) [reset = 0h]

TX_PING_TO_REF is shown in Figure 29-53 and described in Table 29-57.

Return to Summary Table.

Transmit ping timeout counter reference

**Figure 29-53. TX_PING_TO_REF Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TO_REF | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 29-57. TX_PING_TO_REF Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | TO_REF | R/W | 0h | Ping Timer Reference Value.<br>This is the 32-bit reference value for the ping timer. The timer will increment the counter starting from 0. When the reference value is reached, it will generate a timeout event, triggering a ping frame transmission. The counter will then reset to 0 and continue counting.<br>Reset type: SYSRSn |

#### 29.5.1.2.12 TX_PING_TO_CNT Register (Offset = Eh) [reset = 0h]

TX_PING_TO_CNT is shown in Figure 29-54 and described in Table 29-58.

Return to Summary Table.

Transmit ping timeout current count

**Figure 29-54. TX_PING_TO_CNT Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TO_CNT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 29-58. TX_PING_TO_CNT Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-0 | TO_CNT | R | 0h | Ping Timer Counter Value<br>This register contains the current value of the ping timer counter. After reset, this counter will increment until it reaches the reference value (TX_PING_TO_REF), at which point it generates a ping frame transmission. After this point, the counter will reset to 0 and continue counting. This is a free-running counter<br>Reset type: SYSRSn |

#### 29.5.1.2.13 TX_INT_CTRL Register (Offset = 10h) [reset = 0h]

TX_INT_CTRL is shown in Figure 29-55 and described in Table 29-59.

Return to Summary Table.

Transmit interrupt event control register

**Figure 29-55. TX_INT_CTRL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | INT2_EN_PING_TO | INT2_EN_BUF_OVERRUN | INT2_EN_BUF_UNDERRUN | INT2_EN_FRAME_DONE |
| R-0h | | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | INT1_EN_PING_TO | INT1_EN_BUF_OVERRUN | INT1_EN_BUF_UNDERRUN | INT1_EN_FRAME_DONE |
| R-0h | | | | R/W-0h | R/W-0h | R/W-0h | R/W-0h |

**Table 29-59. TX_INT_CTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-12 | RESERVED | R | 0h | Reserved |
| 11 | INT2_EN_PING_TO | R/W | 0h | Enable PING Timer Interrupt to INT2<br>This bit allows the event to generate an interrupt on the INT2 line.<br>0h (R/W) = This event will not trigger an interrupt on TX_INT2.<br>1h (R/W) = The ping timer event will trigger an interrupt on TX_INT2.<br>Reset type: SYSRSn |
| 10 | INT2_EN_BUF_OVERRUN | R/W | 0h | Enable Buffer Overrun Interrupt to INT2<br>This bit allows the event to generate an interrupt on the INT2 line.<br>0h (R/W) = This event will not trigger an interrupt on TX_INT2.<br>1h (R/W) = A Buffer Overrun condition will trigger an interrupt on TX_INT2.<br>Reset type: SYSRSn |
| 9 | INT2_EN_BUF_UNDERRUN | R/W | 0h | Enable Buffer Underrun Interrupt to INT2<br>This bit allows the event to generate an interrupt on the INT2 line.<br>0h (R/W) = This event will not trigger an interrupt on TX_INT2.<br>1h (R/W) = A Buffer Underrun condition will trigger an interrupt on TX_INT2.<br>Reset type: SYSRSn |
| 8 | INT2_EN_FRAME_DONE | R/W | 0h | Enable Frame Done interrupt to INT2<br>This bit allows the event to generate an interrupt on the INT2 line.<br>0h (R/W) = This event will not trigger an interrupt on TX_INT2.<br>1h (R/W) = A Frame Done event will trigger an interrupt on TX_INT2.<br>Reset type: SYSRSn |
| 7-4 | RESERVED | R | 0h | Reserved |
| 3 | INT1_EN_PING_TO | R/W | 0h | Enable Ping Timer Interrupt to INT1<br>This bit allows the event to generate an interrupt on the INT1 line.<br>0h (R/W) = This event will not trigger an interrupt on TX_INT1.<br>1h (R/W) = The ping timer event will trigger an interrupt on TX_INT1.<br>Reset type: SYSRSn |

**Table 29-59. TX_INT_CTRL Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 2 | INT1_EN_BUF_OVERRUN | R/W | 0h | Enable Buffer Overrun Interrupt to INT1<br>This bit allows the event to generate an interrupt on the INT1 line.<br>0h (R/W) = This event will not trigger an interrupt on TX_INT1.<br>1h (R/W) = A Buffer Overrun condition will trigger an interrupt on TX_INT1.<br>Reset type: SYSRSn |
| 1 | INT1_EN_BUF_UNDERRUN | R/W | 0h | Enable Buffer Underrun Interrupt to INT1<br>This bit allows the event to generate an interrupt on the INT1 line.<br>0h (R/W) = This event will not trigger an interrupt on TX_INT1.<br>1h (R/W) = A Buffer Underrun condition will trigger an interrupt on TX_INT1.<br>Reset type: SYSRSn |
| 0 | INT1_EN_FRAME_DONE | R/W | 0h | Enable Frame Done interrupt to INT1<br>This bit allows the event to generate an interrupt on the INT1 line.<br>0h (R/W) = This event will not trigger an interrupt on TX_INT1.<br>1h (R/W) = A Frame Done event will trigger an interrupt on TX_INT1.<br>Reset type: SYSRSn |

#### 29.5.1.2.14 TX_DMA_CTRL Register (Offset = 11h) [reset = 0h]

TX_DMA_CTRL is shown in Figure 29-56 and described in Table 29-60.

Return to Summary Table.

Transmit DMA event control register

**Figure 29-56. TX_DMA_CTRL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | DMA_EVT_EN |
| R-0h | | | | | | | RX-0h |

**Table 29-60. TX_DMA_CTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-1 | RESERVED | R | 0h | Reserved |
| 0 | DMA_EVT_EN | RX | 0h | DMA Event Enable bit<br>This bit will enable the DMA event to be generated upon the completion of a transmit frame.<br>0h (R/W) = A DMA event will not be generated.<br>1h (R/W) = A DMA event will be generated upon the completion of a transmitted frame.<br>Note: The DMA event will only be generated for data frames.<br>Reset type: SYSRSn |

#### 29.5.1.2.15 TX_LOCK_CTRL Register (Offset = 12h) [reset = 0h]

TX_LOCK_CTRL is shown in Figure 29-57 and described in Table 29-61.

Return to Summary Table.

Transmit lock control register

**Figure 29-57. TX_LOCK_CTRL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| KEY | | | | | | | |
| W-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | | | | LOCK |
| R-0h | | | | | | | R/W-0h |

**Table 29-61. TX_LOCK_CTRL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | KEY | W | 0h | Write Key<br>In order to write to this register, 0xA5 must be written to this field at the same time. Otherwise, writes are ignored. The key is cleared immediately after writing, so it must be written again for every change to this register.<br>Reset type: SYSRSn |
| 7-1 | RESERVED | R | 0h | Reserved |
| 0 | LOCK | R/W | 0h | Control Register Lock Enable bit<br>This bit locks the contents of all the transmit control registers that support a lock protection. Once locked, further writes will not take effect until a SYSRS has reset this register. Once set, further writes to this bit will be ignored.<br>0h (R/W) = Transmit control registers can be modified and are not locked.<br>1h (R/W) = Transmit control registers are locked and cannot be modified until this bit is cleared by SYSRS. Any further writes to this bit are ignored.<br>Note: The KEY field must contatin 0xA5 for any write to this bit to take effect.<br>Reset type: SYSRSn |

#### 29.5.1.2.16 TX_EVT_STS Register (Offset = 14h) [reset = 0h]

TX_EVT_STS is shown in Figure 29-58 and described in Table 29-62.

Return to Summary Table.

Transmit event and error status flag register

**Figure 29-58. TX_EVT_STS Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | PING_TRIGGERED | BUF_OVERRUN | BUF_UNDERRUN | FRAME_DONE |
| R-0h | | | | R-0h | R-0h | R-0h | R-0h |

**Table 29-62. TX_EVT_STS Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | PING_TRIGGERED | R | 0h | Ping Frame Triggered Flag Bit<br>This bit indicates that a ping frame has been triggered. This bit is set by hardware when either the ping timer or an external trigger event have occured. Software can also force this bit to get set by writing to the TX_EVT_FRC register.<br>0h (R) = A ping frame has not been triggered.<br>1h (R) = A ping frame has been triggered by either the ping timer or external trigger.<br>To clear this bit, write to the corresponding bit in the TX_EVT_CLR register.<br>Reset type: SYSRSn |
| 2 | BUF_OVERRUN | R | 0h | Buffer Overrun Flag Bit<br>This bit inditcates that buffer overrun has occured.Software can also force this bit to get set by writing to the TX_EVT_FRC register.<br>0h (R) = Buffer Overrun has not occured.<br>1h (R) = Buffer Overrun has occured.<br>To clear this bit, write to the corresponding bit in the TX_EVT_CLR register.<br>Reset type: SYSRSn |
| 1 | BUF_UNDERRUN | R | 0h | Buffer Underrun Flag Bit<br>This bit inditcates that buffer underrun has occured.Software can also force this bit to get set by writing to the TX_EVT_FRC register.<br>0h (R) = Buffer Underrun has not occured.<br>1h (R) = Buffer Underrun has occured.<br>To clear this bit, write to the corresponding bit in the TX_EVT_CLR register.<br>Reset type: SYSRSn |

**Table 29-62. TX_EVT_STS Register Field Descriptions (continued)**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 0 | FRAME_DONE | R | 0h | Frame Done Flag Bit<br>This bit inditcates a Frame Done condition. This bit is set by hardware when a frame transmission has been completed. Software can also force this bit to get set by writing to the TX_EVT_FRC register.<br>0h (R) = Frame Done condition has not occured.<br>1h (R) = Frame Done condition has occured.<br>To clear this bit, write to the corresponding bit in the TX_EVT_CLR register.<br>Reset type: SYSRSn |

#### 29.5.1.2.17 TX_EVT_CLR Register (Offset = 16h) [reset = 0h]

TX_EVT_CLR is shown in Figure 29-59 and described in Table 29-63.

Return to Summary Table.

Transmit event and error clear register

**Figure 29-59. TX_EVT_CLR Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | PING_TRIGGERED | BUF_OVERRUN | BUF_UNDERRUN | FRAME_DONE |
| R-0h | | | | W-0h | W-0h | W-0h | W-0h |

**Table 29-63. TX_EVT_CLR Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | PING_TRIGGERED | W | 0h | Ping Frame Triggered Flag Clear bit<br>This bit clears the corresponding bit in the TX_EVT_STS register.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the TX_EVT_STS register to 0.<br>Note: This bit may not always be cleared when writing to the corresponding TX_EVT_CLR bit. If PING_TIMEOUT MODE is configured to be 0, a hardware ping timeout may occur when another frame is actively being transmitted. In this case, if this bit still shows as 1 after the clear bit is written then the ping frame has been triggered but not serviced. This bit does not indicate that the ping frame has been completely sent, only that it has been triggered by the timeout event.<br>Reset type: SYSRSn |
| 2 | BUF_OVERRUN | W | 0h | Buffer Overrun Flag Clear bit<br>This bit clears the corresponding bit in the TX_EVT_STS register.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the TX_EVT_STS register to 0.<br>Reset type: SYSRSn |
| 1 | BUF_UNDERRUN | W | 0h | Buffer Underrun Flag Clear bit<br>This bit clears the corresponding bit in the TX_EVT_STS register.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the TX_EVT_STS register to 0.<br>Reset type: SYSRSn |
| 0 | FRAME_DONE | W | 0h | Frame Done Flag Clear bit<br>This bit clears the corresponding bit in the TX_EVT_STS register.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Writing a 1 to this bit will clear the corresponding bit in the TX_EVT_STS register to 0.<br>Reset type: SYSRSn |

#### 29.5.1.2.18 TX_EVT_FRC Register (Offset = 17h) [reset = 0h]

TX_EVT_FRC is shown in Figure 29-60 and described in Table 29-64.

Return to Summary Table.

Transmit event and error flag force register

**Figure 29-60. TX_EVT_FRC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RESERVED | | | | PING_TRIGGERED | BUF_OVERRUN | BUF_UNDERRUN | FRAME_DONE |
| R-0h | | | | W-0h | W-0h | W-0h | W-0h |

**Table 29-64. TX_EVT_FRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-4 | RESERVED | R | 0h | Reserved |
| 3 | PING_TRIGGERED | W | 0h | Ping Frame Triggered Flag Force bit<br>This bit will cause the corresponding bit in the TX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Force the corresponding flag bit in the TX_EVT_STS Register.<br>Reset type: SYSRSn |
| 2 | BUF_OVERRUN | W | 0h | Buffer Overrun Flag Force bit<br>This bit will cause the corresponding bit in the TX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.<br>0h (R/W) = Writing a 0 to this bit will have no effect.<br>1h (R/W) = Force the corresponding flag bit in the TX_EVT_STS Register.<br>Reset type: SYSRSn |
| 1 | BUF_UNDERRUN | W | 0h | Buffer Underrun Flag Force bit<br>This bit will cause the corresponding bit in the TX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Force the corresponding flag bit in the TX_EVT_STS Register.<br>Reset type: SYSRSn |
| 0 | FRAME_DONE | W | 0h | Frame Done Flag Force bit<br>This bit will cause the corresponding bit in the TX_EVT_STS register to get set. The purpose of this register is to allow software to simulate the effect of the event and test the associated software/ISR.<br>0h (W) = Writing a 0 to this bit will have no effect.<br>1h (W) = Force the corresponding flag bit in the TX_EVT_STS Register.<br>Reset type: SYSRSn |

#### 29.5.1.2.19 TX_USER_CRC Register (Offset = 18h) [reset = 0h]

TX_USER_CRC is shown in Figure 29-61 and described in Table 29-65.

Return to Summary Table.

Transmit user-defined CRC register

**Figure 29-61. TX_USER_CRC Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| USER_CRC | | | | | | | |
| R/W-0h | | | | | | | |

**Table 29-65. TX_USER_CRC Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-0 | USER_CRC | R/W | 0h | User-defined CRC<br>This register contains the 8-bit CRC value to be transmitted in the next frame if the transmission is set for user-defined CRC option (TX_OPER_CTRL_LO.SW_CRC = 1). This register is ignored if the hardware CRC generation is enabled.<br>Reset type: SYSRSn |

#### *29.5.1.2.20 TX_ECC_DATA Register (Offset = 20h) [reset = 0h]*

TX_ECC_DATA is shown in Figure 29-62 and described in Table 29-66.

Return to Summary Table.

Transmit ECC data register

**Figure 29-62. TX_ECC_DATA Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA_HIGH | | | | | | | | | | | | | | | | DATA_LOW | | | | | | | | | | | | | | | |
| R/W-0h | | | | | | | | | | | | | | | | R/W-0h | | | | | | | | | | | | | | | |

**Table 29-66. TX_ECC_DATA Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-16 | DATA_HIGH | R/W | 0h | Upper 16 bits of ECC Data<br>Writing to this bitfield will cause the ECC logic to compute the ECC(SEC-DED) the entire 32-bit register and update TX_ECC_VAL register with the results. Software should write to these 16 bits of the register in a 32-bit write when needing to compute ECC for 32-bits for the full TX_ECC_DATA register.<br>Reset type: SYSRSn |
| 15-0 | DATA_LOW | R/W | 0h | Lower 16 bits of ECC Data<br>Writing to this bitfield will cause the ECC logic to compute the ECC(SEC-DED) for these 16 bits and update the TX_ECC_VAL register with the results. Software should write to these register bits as a 16-bit write when needing to compute ECC for 16-bits.<br>Reset type: SYSRSn |

#### 29.5.1.2.21 TX_ECC_VAL Register (Offset = 22h) [reset = Ch]

TX_ECC_VAL is shown in Figure 29-63 and described in Table 29-67.

Return to Summary Table.

Transmit ECC value register

**Figure 29-63. TX_ECC_VAL Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | |
| R-0h | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ECC_VAL | | | | | | | |
| R-Ch | | | | | | | |

**Table 29-67. TX_ECC_VAL Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-8 | RESERVED | R | 0h | Reserved |
| 7-0 | ECC_VAL | R | Ch | Computed ECC Value<br>This field contains the ECC value computed using SEC-DED either for 16-bit or 32-bit data in the TX_ECC_DATA register.<br>Reset type: SYSRSn |

#### 29.5.1.2.22 TX_BUF_BASE Register (Offset = 40h) [reset = 0h]

TX_BUF_BASE is shown in Figure 29-64 and described in Table 29-68.

Return to Summary Table.

Base address for transmit buffer

**Figure 29-64. TX_BUF_BASE Register**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BASE_ADDRESS | | | | | | | | | | | | | | | |
| R-0h | | | | | | | | | | | | | | | |

**Table 29-68. TX_BUF_BASE Register Field Descriptions**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-0 | BASE_ADDRESS | R | 0h | Transmit Data Buffer Base Address<br>This is the base address of the 16-word data buffer used by the transmitter.<br>Reset type: SYSRSn |

# Revision History

**Changes from February 1, 2017 to December 15, 2017** **Page**

- Chapter 2: **C28x Processor**: ..... 75
- Chapter 2: Added links to related documents to the abstract. ..... 75
- Chapter 3: **System Control** ..... 78
- Chapter 3: Added links to related documents to the abstract. ..... 78
- Section 3.3: Removed the sentence indicating "the configuration registers are DC1..." ..... 80
- Figure 3-6: Revised the note undereath GPIO18. ..... 98
- Section 3.7.10: Revised Step 2 in the list of procedures. ..... 103
- Figure 3-11: Revised the Watchdog Timer Module figure. ..... 107
- Section 3.10.2: Created this new section; removed all previous references to STANDBY Mode. ..... 110
- Section 3.12.2: Added the last bullet beginning "Users must check and install updates..."; also changed reference to SPNU628 in F021 Flash API Library ..... 120
- Section 3.12.10: Added a sentence beginning "ECC memory space in Flash..." ..... 127
- Section 3.12.10.3: Deleted the bullet beginning "write 64-bit data in FDATAx_TEST..." ..... 129
- Figure 3-20: Following this figure, provided a note indicating USER OTP is ECC protected. ..... 137
- Section 3.13.3: Revised the first paragraph. ..... 138
- Section 3.13.7.8: Added the Device Unique ID section. ..... 145
- Chapter 4: **ROM Code and Peripheral Booting** ..... 601
- Table 4-17: Added thed Boot ROM Memory Map (Silicon revision B) table. ..... 612
- Table 4-20: Added the Reserved RAM and Flash Memory Map (silicon revision B) table. ..... 614
- Section 4.6.6.1: Revised the paragraph. ..... 614
- Chapter 8: **General-Purpose Input/Output** ..... 822
- Section 8.2: Removed reference to Standby mode. ..... 824
- Chapter 9: **Cross (XBAR)** ..... 921
- Section 9.1: Added text indicating the digital input of AIOs are available. ..... 922
- Chapter 10: **Direct Memory Access (DMA)** ..... 1176
- Figure 10-1: Revised the block diagram. ..... 1178
- Chapter 12: **Analog Subsystem** ..... 1261
- Section 12.1.1: Revised the Features section. ..... 1262
- Table 12-1 (Analog Pins and Internal Connections): Updated COMPARATOR SUBSYSTEM (MUX) columns. ..... 1267
- Table 12-3: Added the Reference Summary table. ..... 1269
- Chapter 13: **Analog-to-Digital Converter (ADC)** ..... 1288
- Chapter 13: Changed Type 4 to Type 5. ..... 1288
- Figure 13-1: Modified the ADC Module block diagram ..... 1290
- Section 13.1.3.2.3: Added the Ganged References section. ..... 1292
- Section 13.1.10: Added the OSDETECT section. ..... 1307
- Section 13.2.1: Revised the ADC Timing Diagrams section. ..... 1310
- Section 13.3.2: Revised the Choosing an Acquisition Window Duration section. ..... 1314
- Section 13.3.5: Revisedthe Internal Temp Sensor section. ..... 1316
- Chapter 15: Buffered Digital-to-Analog Converter ..... 1478
- Figure 15-1: Revised the block diagram. ..... 1479
- Section 15.2.2: Added this new section. ..... 1480
- Figure 17-2: Revised the block diagram. ..... 1528
- **Chapter 18: Enhanced Pulse Width Modulator (ePWM)** ..... 1607
- Section 18.1: Added Type0 to Type1 Enhancements section. ..... 1609
- Section 18.3.2.1: Removed the Note. ..... 1617
- Figure 18-5: Revised the Time-Base Submodule Signals and Registers figure to reflect the register definition in mux and number. ..... 1618
- Section 18.3.2.5: Removed the note in this section. ..... 1623
- Table 18-10: Revised table values. ..... 1653

## IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265