TMS320x2833x, 2823x System Control and Interrupts

Reference Guide

Literature Number: SPRUFB0D
September 2007–Revised March 2010

List of Figures

List of Tables

Read This First

About This Manual

This reference guide is applicable for the systems control and interrupts found on the TMS320F2833x/TMS320F2823x Delfino digital signal controllers (DSCs) .

This guide describes how various 2833x/2823x DSC system controls and interrupts work. It includes information on the:

- Flash and one-time programmable (OTP) memories
- Code security module (CSM), which is a security feature incorporated in TMS320C28x™ devices.
- Clocking mechanisms including the oscillator, PLL, XCLKOUT, watchdog module, and the low-power modes. In addition, the 32-bit CPU Timers are also described.
- GPIO multiplexing (MUX) registers used to select the operation of shared pins on the device.
- Accessing the peripheral frames to write to and read from various peripheral registers on the device.
- Interrupt sources both external and the peripheral interrupt expansion (PIE) block that multiplexes numerous interrupt sources into a smaller set of interrupt inputs.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h or with a leading 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 below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure designate a bit that is used for future device expansion.

Related Documentation From Texas Instruments

The following books describe the 2833x and related support tools that are available on the TI website:

Data Manual and Errata—

SPRS439— TMS320F28335, TMS320F28334, TMS320F28332, TMS320F28235, TMS320F28234, TMS320F28232 Digital Signal Controllers (DSCs) Data Manual contains the pinout, signal descriptions, as well as electrical and timing specifications for the F2833x/2823x devices.

SPRZ272— TMS320F28335, F28334, F28332, TMS320F28235, F28234, F28232 Digital Signal Controllers (DSCs) Silicon Errata describes the advisories and usage notes for different versions of silicon.

CPU User's Guides—

SPRU430 — TMS320C28x CPU and Instruction Set Reference Guide describes the central processing unit (CPU) and the assembly language instructions of the TMS320C28x fixed-point digital signal processors (DSPs). It also describes emulation features available on these DSPs.

SPRUEO2 — TMS320C28x Floating Point Unit and Instruction Set Reference Guide describes the floating-point unit and includes the instructions for the FPU.

Peripheral Guides—

SPRU566 — TMS320x28xx, 28xxx DSP Peripheral Reference Guide describes the peripheral reference guides of the 28x digital signal processors (DSPs).

SPRUFB0 — TMS320x2833x, 2823x System Control and Interrupts Reference Guide describes the various interrupts and system control features of the 2833x and 2823x digital signal controllers (DSCs).

SPRU812 — TMS320x2833x, 2823x Analog-to-Digital Converter (ADC) Reference Guide describes how to configure and use the on-chip ADC module, which is a 12-bit pipelined ADC.

SPRU949 — TMS320x2833x, 2823x DSC External Interface (XINTF) Reference Guide describes the XINTF, which is a nonmultiplexed asynchronous bus, as it is used on the 2833x and 2823x devices.

SPRU963 — TMS320x2833x, 2823x Boot ROM Reference Guide describes the purpose and features of the bootloader (factory-programmed boot-loading software) and provides examples of code. It also describes other contents of the device on-chip boot ROM and identifies where all of the information is located within that memory.

SPRUFB7 — TMS320x2833x, 2823x Multichannel Buffered Serial Port (McBSP) Reference Guide describes the McBSP available on the 2833x and 2823x devices. The McBSPs allow direct interface between a DSP and other devices in a system.

SPRUFB8 — TMS320x2833x, 2823x Direct Memory Access (DMA) Module Reference Guide describes the DMA on the 2833x and 2823x devices.

SPRUG04 — TMS320x2833x, 2823x Enhanced Pulse Width Modulator (ePWM) Module Reference Guide describes the main areas of the enhanced pulse width modulator that include digital motor control, switch mode power supply control, UPS (uninterruptible power supplies), and other forms of power conversion.

SPRUG02 — TMS320x2833x, 2823x High-Resolution Pulse Width Modulator (HRPWM) Reference Guide describes the operation of the high-resolution extension to the pulse width modulator (HRPWM).

SPRUFG4 — TMS320x2833x, 2823x Enhanced Capture (eCAP) Module Reference Guide describes the enhanced capture module. It includes the module description and registers.

SPRUG05 — TMS320x2833x, 2823x Enhanced Quadrature Encoder Pulse (eQEP) Module Reference Guide describes the eQEP module, which is used for interfacing with a linear or rotary incremental encoder to get position, direction, and speed information from a rotating machine in high-performance motion and position control systems. It includes the module description and registers.

SPRUEU1 — TMS320x2833x, 2823x Enhanced Controller Area Network (eCAN) Reference Guide describes the eCAN that uses established protocol to communicate serially with other controllers in electrically noisy environments.

SPRUFZ5 — TMS320x2833x, 2823x Serial Communications Interface (SCI) Reference Guide describes the SCI, which 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.

SPRUEU3 — TMS320x2833x, 2823x DSC Serial Peripheral Interface (SPI) Reference Guide describes the SPI - a high-speed synchronous serial input/output (I/O) port - that allows a serial bit stream of programmed length (one to sixteen bits) to be shifted into and out of the device at a programmed bit-transfer rate.

SPRUG03 — TMS320x2833x, 2823x Inter-Integrated Circuit (I2C) Module Reference Guide describes the features and operation of the inter-integrated circuit (I2C) module.

Tools Guides—

SPRU513 — TMS320C28x Assembly Language Tools v5.0.0 User's Guide describes the assembly language tools (assembler and other tools used to develop assembly language code), assembler directives, macros, common object file format, and symbolic debugging directives for the TMS320C28x device.

SPRU514 — TMS320C28x Optimizing C/C++ Compiler v5.0.0 User's Guide describes the TMS320C28x™ C/C++ compiler. This compiler accepts ANSI standard C/C++ source code and produces TMS320 DSP assembly language source code for the TMS320C28x device.

SPRU608 — TMS320C28x Instruction Set Simulator Technical Overview describes the simulator, available within the Code Composer Studio for TMS320C2000 IDE, that simulates the instruction set of the C28x™ core.

SPRU625 — TMS320C28x DSP/BIOS 5.32 Application Programming Interface (API) Reference Guide describes development using DSP/BIOS.

Flash and OTP Memory Blocks

This chapter describes the proper sequence to configure the wait states and operating mode of flash and one-time programmable (OTP) memories. It also includes information on flash and OTP power modes and how to improve flash performance by enabling the flash pipeline mode.

1 Flash and OTP Memory

This section describes how to configure flash and one-time programmable (OTP) memory.

1.1 Flash Memory

The on-chip flash is uniformly mapped in both program and data memory space. This flash memory is always enabled and features:

- **Multiple sectors**
  The minimum amount of flash memory that can be erased is a sector. Having multiple sectors provides the option of leaving some sectors programmed and only erasing specific sectors.
- **Code security**
  The flash is protected by the Code Security Module (CSM). By programming a password into the flash, the user can prevent access to the flash by unauthorized persons. See Section 4 for information in using the Code Security Module.
- **Low power modes**
  To save power when the flash is not in use, two levels of low power modes are available. See Section 2 for more information on the available flash power modes.
- **Configurable wait states**
  Configurable wait states can be adjusted based on CPU frequency to give the best performance for a given execution speed.
- **Enhanced performance**
  A flash pipeline mode is provided to improve performance of linear code execution.

1.2 OTP Memory

The 1K x 16 block of one-time programmable (OTP) memory is uniformly mapped in both program and data memory space. Thus, the OTP can be used to program data or code. This block, unlike flash, can be programmed only one time and cannot be erased.

2 Flash and OTP Power Modes

The following operating states apply to the flash and OTP memory:

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

the standby state, a CPU data read or opcode fetch to the flash or OTP memory map area will automatically initiate a change in power modes to the active state. During this transition time to the active state, the CPU will automatically be stalled. Once the flash/OTP has reached the active state, the CPU access will complete as normal.

- **Active or Read State**

  In this state, the bank and pump are in active power mode state (highest power). The CPU read or fetch access wait states to the flash/OTP memory map area is controlled by the FBANKWAIT and FOTPWAIT registers. A prefetch mechanism called flash pipeline can also be enabled to improve fetch performance for linear code execution.

NOTE: During the boot process, the Boot ROM performs a dummy read of the Code Security Module (CSM) password locations located in the flash. This read is performed to unlock a new or erased device that has no password stored in it so that flash programming or loading of code into CSM protected SARAM can be performed. On devices with a password stored, this read has no affect and the CSM remains locked (see Section 4 for information on the CSM). One effect of this read is that the flash will transition from the sleep (reset) state to the active state.

The flash/OTP bank and pump are always in the same power mode. See Figure 1 for a graphic depiction of the available power states. You can change the current flash/OTP memory power state as follows:

- **To move to a lower power state**

  Change the PWR mode bits from a higher power mode to a lower power mode. This change instantaneously moves the flash/OTP bank to the lower power state. This register should be accessed only by code running outside the flash/OTP memory.

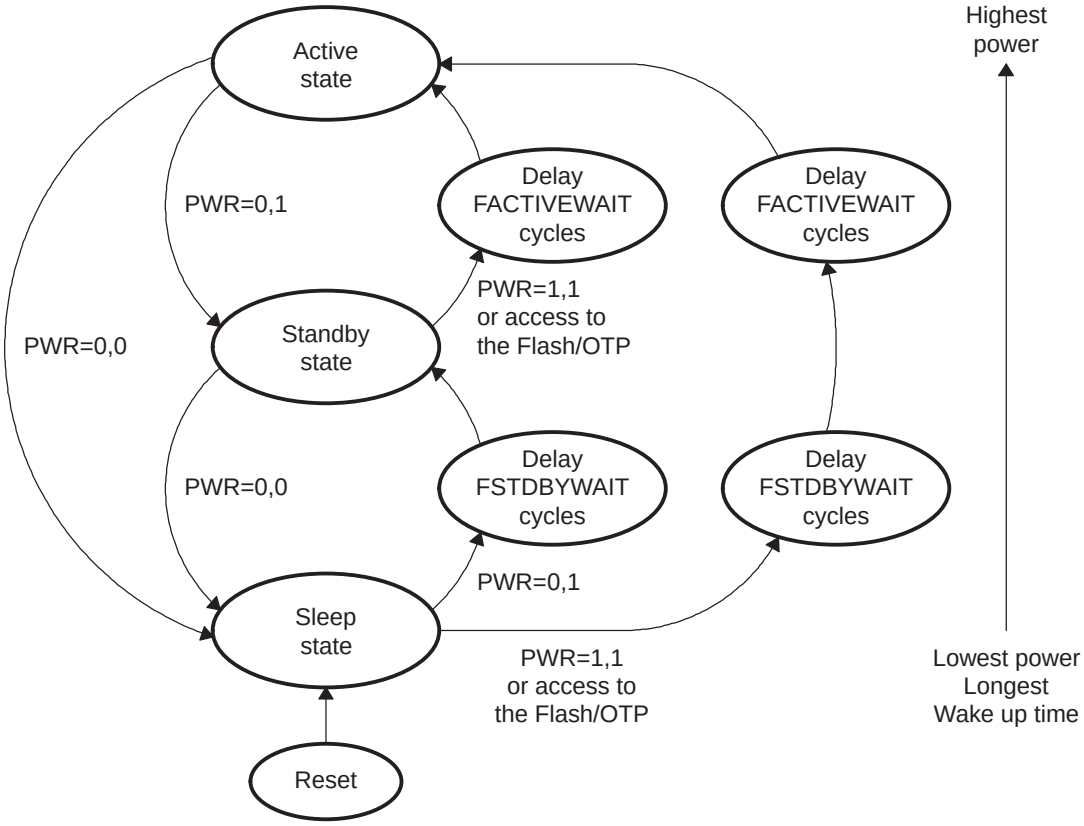
- **To move to a higher power state**

  To move from a lower power state to a higher power state, there are two options.

  1. Change the FPWR register from a lower state to a higher state. This access brings the flash/OTP memory to the higher state.
  2. Access the flash or OTP memory by a read access or program opcode fetch access. This access automatically brings the flash/OTP memory to the active state.

There is a delay when moving from a lower power state to a higher one. See Figure 1. This delay is required to allow the flash to stabilize at the higher power mode. If any access to the flash/OTP memory occurs during this delay the CPU automatically stalls until the delay is complete.

Figure 1. Flash Power Mode State Diagram



The duration of the delay is determined by the FSTDBYWAIT and FACTIVEWAIT registers. Moving from the sleep state to a standby state is delayed by a count determined by the FSTDBYWAIT register. Moving from the standby state to the active state is delayed by a count determined by the FACTIVEWAIT register. Moving from the sleep mode (lowest power) to the active mode (highest power) is delayed by FSTDBYWAIT + FACTIVEWAIT. These registers should be left in their default state.

2.1 Flash and OTP Performance

CPU read or data fetch operations to the flash/OTP can take one of the following forms:

- 32-bit instruction fetch
- 16-bit or 32-bit data space read
- 16-bit program space read

Once flash is in the active power state, then a read or fetch access to the bank memory map area can be classified as a flash access or an OTP access.

The main flash array is organized into rows and columns. The rows contain 2048 bits of information. Accesses to flash and OTP are one of three types:

1. **Flash Memory Random Access**

   The first access to a 2048-`bit row is considered a random access.

2. **Flash Memory Paged Access**

   While the first access to a row is considered a random access, subsequent accesses within the same row are termed paged accesses.

   The number of wait states for both a random and a paged access can be configured by programming the FBANKWAIT register. The number of wait states used by a random access is controlled by the RANDWAIT bits and the number of wait states used by a paged access is controlled by the PAGEWAIT bits. The FBANKWAIT register defaults to a worst-case wait state count and, thus, 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. F2833x/F2823x devices require a minimum of 1 wait-state for PAGE/RANDOM flash access. This assumes that the CPU speed is low enough to accommodate the access time. To determine the random and paged access time requirements, refer to the Data Manual for your particular device.

3. **OTP Access**

   Read or fetch accesses to the OTP are controlled by the OTPWAIT bits in the FOTPWAIT register. Accesses to the OTP take longer than the flash and there is no paged mode. To determine OTP access time requirements, see the data manual for your particular device.

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

- CPU writes to the flash or OTP memory map area are ignored. They complete in a single cycle.
- When the Code Security Module (CSM) is secured, reads to the flash/OTP memory map area from outside the secure zone take the same number of cycles as a normal access. However, the read operation returns a zero.
- Reads of the CSM password locations are hardwired for 16 wait-states. The PAGEWAIT and RANDOMWAIT bits have no effect on these locations. See Section 4 for more information on the CSM.

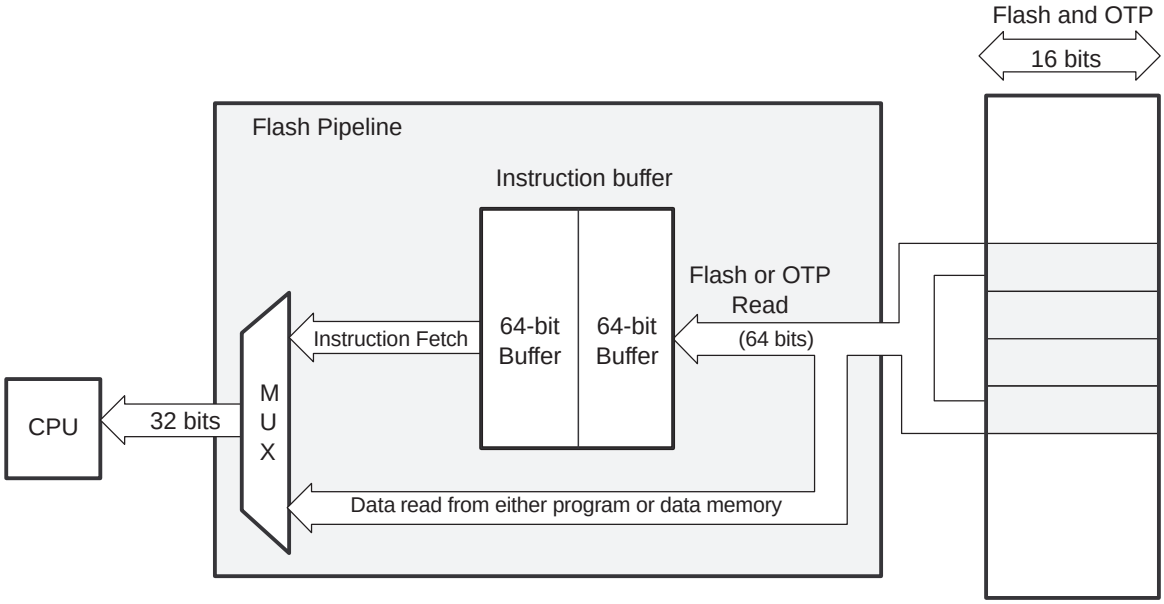
2.2 Flash Pipeline Mode

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 pipeline mode has been implemented. The flash pipeline feature is disabled by default. Setting the ENPIPE bit in the FOPT register enables this mode. The flash pipeline mode is independent of the CPU pipeline.

An instruction fetch from the flash or OTP reads out 64 bits per access. The starting address of the access from flash is automatically aligned to a 64-bit boundary such that the instruction location is within the 64 bits to be fetched. With flash pipeline mode enabled (see Figure 2), the 64 bits read from the instruction fetch are stored in a 64-bit wide by 2-level deep instruction pre-fetch buffer. The contents of this pre-fetch buffer are then sent to the CPU for processing as required.

Up to two 32-bit instructions or up to four 16-bit instructions can reside within a single 64-bit access. The majority of C28x instructions are 16 bits, so for every 64-bit instruction fetch from the flash bank it is likely that there are up to four instructions in the pre-fetch buffer ready to process through the CPU. During the time it takes to process these instructions, the flash pipeline automatically initiates another access to the flash bank to pre-fetch the next 64 bits. In this manner, the flash pipeline mode works in the background to keep the instruction pre-fetch buffers as full as possible. Using this technique, the overall efficiency of sequential code execution from flash or OTP is improved significantly.

Figure 2. Flash Pipeline



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

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

2.3 Reserved Locations Within Flash and OTP

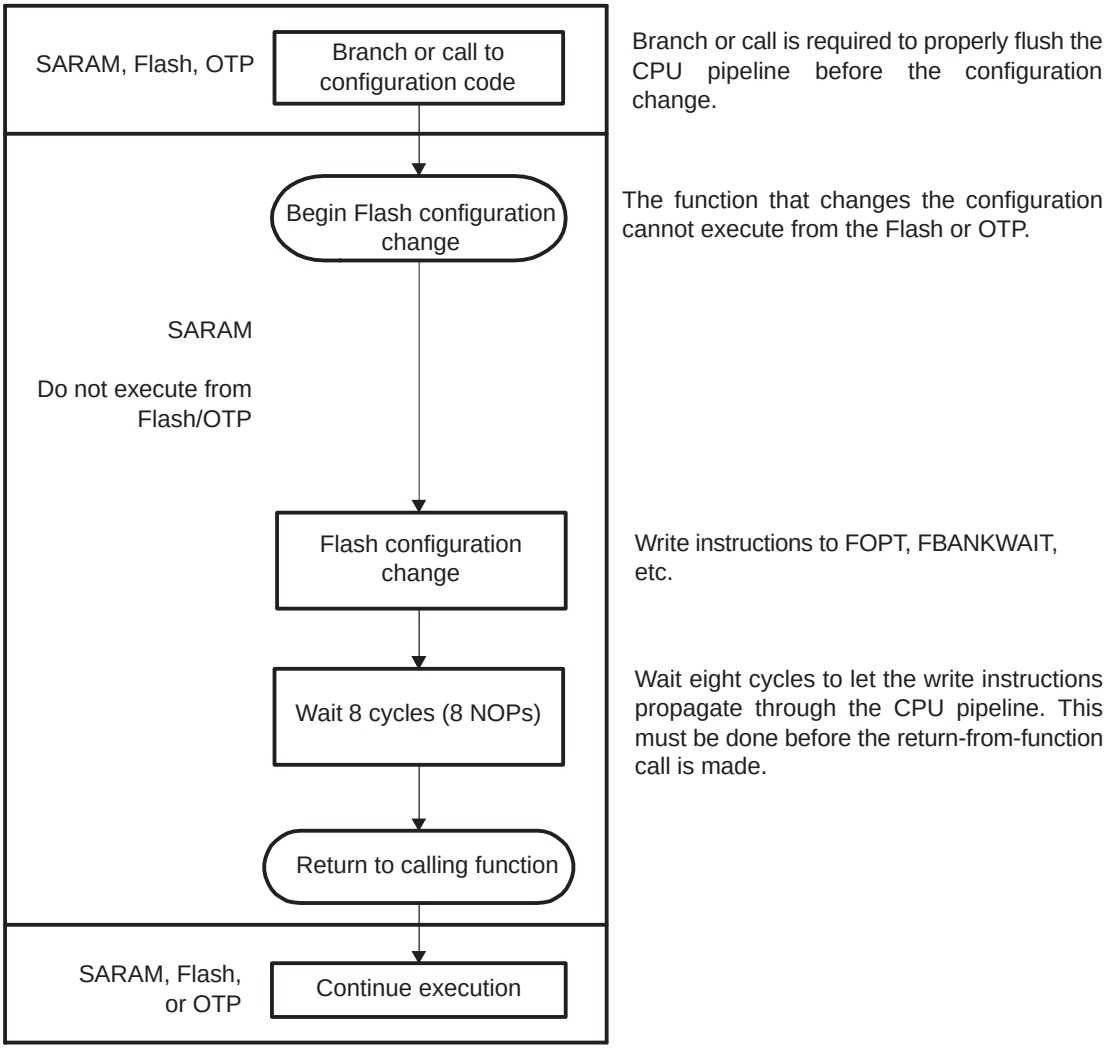
When allocating code and data to flash and OTP memory, keep the following in mind:

1. Address locations 0x33 FFF6 and 0x33 FFF7 are reserved for an "entry into flash" branch instruction. When the "boot to flash" boot option is used, the boot ROM will jump to address 0x33 FFF6. If you program a branch instruction here that will then re-direct code execution to the entry point of the application.
2. For code security operation, all addresses between 0x33 FF80 and 0x33 FFF5 cannot be used for program code or data, but must be programmed to 0x0000 when the Code Security Password is programmed. If security is not a concern, addresses 0x33 FF80 through 0x33 FFF5 may be used for code or data. See Section 4 for information in using the Code Security Module.
3. Addresses from 0x33 FFF0 to 0x33 FFF5 are reserved for data variables and should not contain program code.

2.4 Procedure to Change the Flash Configuration 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 pre-fetch operations. To be sure that no access takes place during the configuration change, you should follow the procedure shown in Figure 3 for any code that modifies the FOPT, FPWR, FBANKWAIT, or FOTPWAIT registers.

Figure 3. Flash Configuration Access Flow Diagram



3 Flash and OTP Registers

The flash and OTP memory can be configured by the registers shown in Table 1. The configuration registers are all EALLOW protected. The bit descriptions are in Figure 4 through Figure 10.

Table 1. Flash/OTP Configuration Registers

Name [1] [2]	Address	Size (x16)	Description	Bit Description
FOPT	0x0A80	1	Flash Option Register	Figure 4
Reserved	0x0A81	1	Reserved	
FPWR	0x0A82	1	Flash Power Modes Register	Figure 5
FSTATUS	0x0A83	1	Status Register	Figure 6
FSTDBYWAIT [3]	0x0A84	1	Flash Sleep To Standby Wait Register	Figure 7
FACTIVEWAIT [3]	0x0A85	1	Flash Standby To Active Wait Register	Figure 8
FBANKWAIT	0x0A86	1	Flash Read Access Wait State Register	Figure 9
FOTPWAIT	0x0A87	1	OTP Read Access Wait State Register	Figure 10

(1) These registers are EALLOW protected. See Section 7.2 for information.
(2) These registers are protected by the Code Security Module (CSM). See Section 4 for more information.
(3) These registers should be left in their default state.

NOTE: The flash configuration registers should not be written to by code that is running from OTP or flash memory or while an access to flash or OTP may be in progress. All register accesses to the flash registers should be made from code executing outside of flash/OTP memory and an access should not be attempted until all activity on the flash/OTP has completed. No hardware is included to protect against this.

To summarize, you can read the flash registers from code executing in flash/OTP; however, do not write to the registers.

CPU write access to the flash configuration registers can be enabled only by executing the EALLOW instruction. Write access is disabled when the EDIS instruction is executed. This protects the registers from spurious accesses. Read access is always available. The registers can be accessed through the JTAG port without the need to execute EALLOW. See Section 7.2 for information on EALLOW protection. These registers support both 16-bit and 32-bit accesses.

Figure 4. Flash Options Register (FOPT)

15	1	0
Reserved		ENPIPE
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 2. Flash Options Register (FOPT) Field Descriptions

Bit	Field	Value	Description [1] [2] [3]
15-1	Reserved		
0	ENPIPE		Enable Flash Pipeline Mode Bit. Flash pipeline mode is active when this bit is set. The pipeline mode improves performance of instruction fetches by pre-fetching instructions. See Section 2.2 for more information. When pipeline mode is enabled, the flash wait states (paged and random) must be greater than zero. On flash devices, ENPIPE affects fetches from flash and OTP.
		0	Flash Pipeline mode is not active. (default)
		1	Flash Pipeline mode is active.

(1) This register is EALLOW protected. See Section 7.2 for more information.
(2) This register is protected by the Code Security Module (CSM). See Section 4 for more information.
(3) When writing to this register, follow the procedure described in Section 2.4.

Figure 5. Flash Power Register (FPWR)

15	2	1	0
Reserved		PWR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3. Flash Power Register (FPWR) Field Descriptions

Bit	Field	Value	Description [1] [2]
15-2	Reserved		
1-0	PWR		Flash Power Mode Bits. Writing to these bits changes the current power mode of the flash bank and pump. See section Section 2 for more information on changing the flash bank power mode.
		00	Pump and bank sleep (lowest power)
		01	Pump and bank standby
		10	Reserved (no effect)
		11	Pump and bank active (highest power)

(1) This register is EALLOW protected. See Section 7.2 for more information.
(2) This register is protected by the Code Security Module (CSM). See Section 4 for more information.

Figure 6. Flash Status Register (FSTATUS)

15					9	8
Reserved						3VSTAT
R-0						R/W1C-0
7	6	5	4	3	2	1	0
Reserved				ACTIVEWAITS	STDBYWAITS	PWRS	
R-0				R-0	R-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; W1C = Write 1 to clear; -*n* = value after reset

Table 4. Flash Status Register (FSTATUS) Field Descriptions

Bit	Field	Value	Description [1] [2]
15-9	Reserved		Reserved
8	3VSTAT		Flash Voltage (V_{DD3VFL}) Status Latch Bit. When set, this bit indicates that the 3VSTAT signal from the pump module went to a high level. This signal indicates that the flash 3.3-V supply went out of the allowable range.
		0	Writes of 0 are ignored.
		1	When this bit reads 1, it indicates that the flash 3.3-V supply went out of the allowable range.
			Clear this bit by writing a 1.
7-4	Reserved		Reserved
3	ACTIVEWAITS		Bank and Pump Standby To Active Wait Counter Status Bit. This bit indicates whether the respective wait counter is timing out an access.
		0	The counter is not counting.
		1	The counter is counting.
2	STDBYWAITS		Bank and Pump Sleep To Standby Wait Counter Status Bit. This bit indicates whether the respective wait counter is timing out an access.
		0	The counter is not counting.
		1	The counter is counting.
1-0	PWRS		Power Modes Status Bits. These bits indicate which power mode the flash/OTP is currently in.
			The PWRS bits are set to the new power mode only after the appropriate timing delays have expired.
		00	Pump and bank in sleep mode (lowest power)
		01	Pump and bank in standby mode
		10	Reserved
		11	Pump and bank active and in read mode (highest power)

(1) This register is EALLOW protected. See Section 7.2 for more information.

(2) This register is protected by the Code Security Module (CSM). See Section 4 for more information.

Figure 7. Flash Standby Wait Register (FSTDBYWAIT)

15	9	8	0
Reserved		STDBYWAIT	
R-0		R/W-0x1FF	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 5. Flash Standby Wait Register (FSTDBYWAIT) Field Descriptions

Bit	Field	Value	Description [1] [2]
15-9	Reserved	0	Reserved
8-0	STDBYWAIT		**This register should be left in its default state.**
			Bank and Pump Sleep To Standby Wait Count.
		111111111	511 SYSCLKOUT cycles (default)

(1) This register is EALLOW protected. See Section 7.2 for more information.
(2) This register is protected by the Code Security Module (CSM). See Section 4 for more information.

Figure 8. Flash Standby to Active Wait Counter Register (FACTIVEWAIT)

7	9	8	0
Reserved		ACTIVEWAIT	
R-0		R/W-0x1FF	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 6. Flash Standby to Active Wait Counter Register (FACTIVEWAIT) Field Descriptions

Bits	Field	Value	Description [1] [2]
15-9	Reserved	0	Reserved
8-0	ACTIVEWAIT		**This register should be left in its default state.**
			Bank and Pump Standby To Active Wait Count:
		111111111	511 SYSCLKOUT cycles (default)

(1) This register is EALLOW protected. See Section 7.2 for more information.
(2) This register is protected by the Code Security Module (CSM). See Section 4 for more information.

Figure 9. Flash Wait-State Register (FBANKWAIT)

15 12	11 8	7 4	3 0
Reserved	PAGEWAIT	Reserved	RANDWAIT
R-0	R/W-0xF	R-0	R/W-0xF

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 7. Flash Wait-State Register (FBANKWAIT) Field Descriptions

Bits	Field	Value	Description (1) (2) (3)
15-12	Reserved		Reserved
11-8	PAGEWAIT		Flash Paged Read Wait States. These register bits specify the number of wait states for a paged read operation in CPU clock cycles (0..15 SYSCLKOUT cycles) to the flash bank. See Section 2.1 for more information. See the device-specific data manual for the minimum time required for a PAGED flash access. You must set RANDWAIT to a value greater than or equal to the PAGEWAIT setting. No hardware is provided to detect a PAGEWAIT value that is greater then RANDWAIT.
		0000	Illegal value. PAGEWAIT must be set greater then 0.
		0001	One wait state per paged flash access or a total of two SYSCLKOUT cycles per access.
		0010	Two wait states per paged flash access or a total of three SYSCLKOUT cycles per access.
		0011	Three wait states per paged flash access or a total of four SYSCLKOUT cycles per access.
		. . .	. . .
		1111	15 wait states per paged flash access or a total of 16 SYSCLKOUT cycles per access. (default)
7-4	Reserved		Reserved
3-0	RANDWAIT		Flash Random Read Wait States. These register bits specify the number of wait states for a random read operation in CPU clock cycles (1..15 SYSCLKOUT cycles) to the flash bank. See Section 2.1 for more information. See the device-specific data manual for the minimum time required for a RANDOM flash access. RANDWAIT must be set greater than 0. That is, at least 1 random wait state must be used. In addition, you must set RANDWAIT to a value greater than or equal to the PAGEWAIT setting. The device will not detect and correct a PAGEWAIT value that is greater then RANDWAIT.
		0000	Illegal value. RANDWAIT must be set greater then 0.
		0001	One wait state per random flash access or a total of two SYSCLKOUT cycles per access.
		0010	Two wait states per random flash access or a total of three SYSCLKOUT cycles per access.
		0011	Three wait states per random flash access or a total of four SYSCLKOUT cycles per access.
		. . .	. . .
		1111	15 wait states per random flash access or a total of 16 SYSCLKOUT cycles per access. (default)

(1) This register is EALLOW protected. See Section 7.2 for more information.
(2) This register is protected by the Code Security Module (CSM). See Section 4 for more information.
(3) When writing to this register, follow the procedure described in Section 2.4.

Figure 10. OTP Wait-State Register (FOTPWAIT)

15	5	4	0
Reserved		OTPWAIT	
R-0		R/W-0x1F	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 8. OTP Wait-State Register (FOTPWAIT) Field Descriptions

Bit(s)	Field	Value	Description (1) (2) (3)
15-5	Reserved	0	Reserved
4-0	OTPWAIT		OTP Read Wait States. These register bits specify the number of wait states for a read operation in CPU clock cycles (1..31 SYSCLKOUT cycles) to the OTP. See CPU Read Or Fetch Access From flash/OTP section for details. There is no PAGE mode in the OTP. OTPWAIT must be set greater than 0. That is, a minimum of 1 wait state must be used. See the device-specific data manual for the minimum time required for an OTP access.
		00000	Illegal value. OTPWAIT must be set to 1 or greater.
		00001	One wait state will be used each OTP access for a total of two SYSCLKOUT cycles per access.
		00010	Two wait states will be used for each OTP access for a total of three SYSCLKOUT cycles per access.
		00011	Three wait states will be used for each OTP access for a total of four SYSCLKOUT cycles per access.
		. . .	. . .
		11111	31 wait states will be used for an OTP access for a total of 32 SYSCLKOUT cycles per access.

(1) This register is EALLOW protected. See Section 7.2 for more information.
(2) This register is protected by the Code Security Module (CSM). See Section 4 for more information.
(3) When writing to this register, follow the procedure described in Section 2.4.

4 Code Security Module (CSM)

The code security module (CSM) is a security feature incorporated in 28x devices. It prevents access/visibility to on-chip memory to unauthorized persons—i.e., it prevents duplication/reverse engineering of proprietary code.

The word secure means access to on-chip memory is protected. The word unsecure means access to on-chip secure memory is not protected — i.e., the contents of the memory could be read by any means (through a debugging tool such as Code Composer Studio™, for example).

4.1 Functional Description

The security module restricts the CPU access to certain on-chip memory without interrupting or stalling CPU execution. When a read occurs to a protected 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 various memories through the JTAG port or external peripherals. Security is defined with respect to the access of on-chip memory and prevents unauthorized copying of proprietary code or data.

The device is secure when CPU access to the on-chip secure memory locations is restricted. When secure, two levels of protection are possible, depending on where the program counter is currently pointing. If code is currently running from inside secure memory, only an access through JTAG is blocked (i.e., through the emulator). This allows secure code to access secure data. Conversely, if code is running from nonsecure memory, all accesses to secure memories are blocked. User code can dynamically jump in and out of secure memory, thereby allowing secure function calls from nonsecure memory. Similarly, interrupt service routines can be placed in secure memory, even if the main program loop is run from nonsecure memory.

Security is protected by a password of 128-bits of data (eight 16-bit words) that is used to secure or unsecure the device. This password is stored at the end of flash in 8 words referred to as the password locations.

The device is unsecured by executing the password match flow (PMF), described Section 4.3.2. Table 9 shows the levels of security.

Table 9. Security Levels

PMF Executed With Correct Password?	Operating Mode	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. JTAG port cannot read the secured memory contents.
Yes	Not Secure	Anywhere	Full access for CPU and JTAG port to secure memory

The password is stored in code security password locations (PWL) in flash memory (0x33 FFF8 - 0x33 FFFF). These locations store the password predetermined by the system designer.

If the password locations have all 128 bits as ones, the device is labeled unsecure. Since new flash devices have erased flash (all ones), only a read of the password locations is required to bring the device into unsecure mode. If the password locations have all 128 bits as zeros, the device is secure, regardless of the contents of the KEY registers. Do not use all zeros as a password or reset the device during an erase of the flash. Resetting the device during an erase routine can result in either an all zero or unknown password. If a device is reset when the password locations are all zeros, the device cannot be unlocked by the password match flow described in Section 4.3.2. Using a password of all zeros will seriously limit your ability to debug secure code or reprogram the flash.

NOTE: If a device is reset while the password locations are all zero or an unknown value, the device will be permanently locked unless a method to run the flash erase routine from secure SARAM is embedded into the flash or OTP. Care must be taken when implementing this procedure to avoid introducing a security hole.

User accessible registers (eight 16-bit words) that are used to unsecure the device are referred to as key registers. These registers are mapped in the memory space at addresses 0x00 0AE0 - 0x00 0AE7 and are EALLOW protected.

In addition to the CSM, the emulation code security logic (ECSL) has been implemented to prevent unauthorized users from stepping through secure code. Any code or data access to flash, user OTP, L0, L1, L2 or L3 memory 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, you must write the correct value into the lower 64 bits of the KEY register, which matches the value stored in the lower 64 bits of the password locations within the flash. Note that dummy reads of all 128 bits of the password in the flash must still be performed. If the lower 64 bits of the password locations are all ones (unprogrammed), then the KEY value does not need to match.

When initially debugging a device with the password locations in flash programmed (i.e., 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. If this happens, the ECSL will trip and cause the emulator connection to be cut. Two solutions to this problem exist:

1. The first is to use the Wait-In-Reset emulation mode, which will hold the device in reset until the emulator takes control. The emulator must support this mode for this option.
2. The second option is to use the “Branch to check boot mode” boot option. This will sit in a loop and continuously poll the boot mode select pins. You can select this boot mode and then exit this mode once the emulator is connected by re-mapping the PC to another address or by changing the boot mode selection pin to the desired boot mode.

NOTE: Reserved Flash Locations When Using Code Security

For code security operation, **all addresses between 0x33 FF80 and 0x33 FFF5 cannot be used as program code or data, but must be programmed to 0x0000** when the Code Security Password is programmed. If security is not a concern, addresses 0x33 FF80 through 0x33 FFF5 may be used for code or data. The 128-bit password (at 0x33 FFF8 - 0x33 FFFF) must not be programmed to zeros. Doing so would permanently lock the device.

Addresses 0x33 FFF0 through 0x33 FFF5 are reserved for data variables and should not contain program code.

Disclaimer: Code Security Module Disclaimer

The Code Security Module ("CSM") included on this device was designed to password protect the data stored in the associated memory and is warranted by Texas Instruments (TI), in accordance with its standard terms and conditions, to conform to TI's published specifications for the warranty period applicable for this device.

TI DOES NOT, HOWEVER, WARRANT OR REPRESENT THAT THE CSM CANNOT BE COMPROMISED OR BREACHED OR THAT THE DATA STORED IN THE ASSOCIATED MEMORY CANNOT BE ACCESSED THROUGH OTHER MEANS. MOREOVER, EXCEPT AS SET FORTH ABOVE, TI MAKES NO WARRANTIES OR REPRESENTATIONS CONCERNING THE CSM OR OPERATION OF THIS DEVICE, INCLUDING ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

IN NO EVENT SHALL TI BE LIABLE FOR ANY CONSEQUENTIAL, SPECIAL, INDIRECT, INCIDENTAL, OR PUNITIVE DAMAGES, HOWEVER CAUSED, ARISING IN ANY WAY OUT OF YOUR USE OF THE CSM OR THIS DEVICE, WHETHER OR NOT TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. EXCLUDED DAMAGES INCLUDE, BUT ARE NOT LIMITED TO LOSS OF DATA, LOSS OF GOODWILL, LOSS OF USE OR INTERRUPTION OF BUSINESS OR OTHER ECONOMIC LOSS.

4.2 CSM Impact on Other On-Chip Resources

The CSM affects access to the on-chip resources listed in Table 10:

Table 10. Resources Affected by the CSM

Address	Block
0x00 0A80 - 0x00 0A87	Flash Configuration Registers
0x00 8000 - 0x00 8FFF	L0 SARAM (4K X 16)
0x00 9000 - 0x00 9FFF	L1 SARAM (4K X 16)
0x00 A000 - 0x00 AFFF	L2 SARAM (4K X 16)
0x00 B000 - 0x00 BFFF	L3 SARAM (4K X 16)
0x30 0000 - 0x33 FFFF	Flash (64K X 16, 32 X 16, or 16 X 16)
0x38 0000 - 0x38 03FF	TI One-Time Programmable (OTP)(1) (1K X 16)
0x38 0400 - 0x38 07FF	User One-Time Programmable (OTP) (1K X 16)
0x3F 8000 - 0x3F 8FFF	L0 SARAM (4K X 16), mirror
0x3F 9000 - 0x3F 9FFF	L1 SARAM (4K X 16), mirror
0x3F A000 - 0x3F AFFF	L2 SARAM (4K X 16), mirror
0x3F B000 - 0x3F BFFF	L3 SARAM (4K X 16), mirror

(1) Not affected by ECSL

The Code Security Module has no impact whatsoever on the following on-chip resources:

- Single-access RAM (SARAM) blocks not designated as secure - These memory blocks can be freely accessed and code run from them, whether the device is in secure or unsecure mode.
- Boot ROM contents - Visibility to the boot ROM contents is not impacted by the CSM.
- On-chip peripheral registers - The peripheral registers can be initialized by code running from on-chip or off-chip memory, whether the device is in secure or unsecure mode.
- PIE Vector Table - Vector tables can be read and written regardless of whether the device is in secure or unsecure mode. Table 10 and Table 11 show which on-chip resources are affected (or are not affected) by the CSM.

Table 11. Resources Not Affected by the CSM

Address	Block
0x00 0000 - 0x00 03FF	M0 SARAM (1K X 16)
0x00 0400 - 0x00 07FF	M1 SARAM (1K X16)
0x00 0800 - 0x00 0CFF	Peripheral Frame 0 (2K X 16)
0x00 0D00 - 0x00 0FFF	PIE Vector RAM (256 X 16)
0x00 6000 - 0x00 6FFF	Peripheral Frame 1 (4K X 16)
0x00 7000 - 0x00 7FFF	Peripheral Frame 2 (4K X 16)
0x00 C000 - 0x00 CFFF	L4 SARAM (4K X 16)
0x00 D000 - 0x00 DFFF	L5 SARAM (4K X 16)
0x00 E000 - 0x00 EFFF	L6 SARAM (4K X 16)
0x00 F000 - 0x00 FFFF	L7 SARAM (4K X 16)
0x3F F000 - 0x3F FFFF	Boot ROM (4K X 16)

To summarize, it is possible to load code onto the unprotected on-chip program SARAM shown in Table 11 via the JTAG connector without any impact from the Code Security Module. The code can be debugged and the peripheral registers initialized, independent of whether the device is in secure or unsecure mode.

4.3 Incorporating Code Security in User Applications

Code security is typically not required in the development phase of a project; however, security is needed once a robust code is developed. Before such a code is programmed in the flash memory, a password should be chosen to secure the device. Once a password is in place, the device is secured (i.e., programming a password at the appropriate locations and either performing a device reset or setting the FORCESEC bit (CSMSCR.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 etc.) requires the supply of a valid password. A password is not needed to run the code out of secure memory (such as in a typical end-customer usage); however, access to secure memory contents for debug purpose requires a password.

Table 12. Code Security Module (CSM) Registers

Memory Address	Register Name	Reset Values	Register Description
KEY Registers			
0x00 - 0AE0	KEY0(1)	0xFFFF	Low word of the 128-bit KEY register
0x00 - 0AE1	KEY1(1)	0xFFFF	Second word of the 128-bit KEY register
0x00 - 0AE2	KEY2(1)	0xFFFF	Third word of the 128-bit KEY register
0x00 - 0AE3	KEY3(1)	0xFFFF	Fourth word of the 128-bit key
0x00 - 0AE4	KEY4(1)	0xFFFF	Fifth word of the 128-bit key
0x00 - 0AE5	KEY5(1)	0xFFFF	Sixth word of the 128-bit key
0x00 - 0AE6	KEY6(1)	0xFFFF	Seventh word of the 128-bit key
0x00 - 0AE7	KEY7(1)	0xFFFF	High word of the 128-bit KEY register
0x00 - 0AEF	CSMSCR(1)	0x005F	CSM status and control register
Password Locations (PWL) in Flash Memory - Reserved for the CSM password only			
0x33 - FFF8	PWL0	User defined	Low word of the 128-bit password
0x33 - FFF9	PWL1	User defined	Second word of the 128-bit password
0x33 - FFFA	PWL2	User defined	Third word of the 128-bit password
0x33 - FFFB	PWL3	User defined	Fourth word of the 128-bit password
0x33 - FFFC	PWL4	User defined	Fifth word of the 128-bit password
0x33 - FFFD	PWL5	User defined	Sixth word of the 128-bit password
0x33 - FFFE	PWL6	User defined	Seventh word of the 128-bit password
0x33 - FFFF	PWL7	User defined	High word of the 128-bit password

(1) These registers are EALLOW protected. Refer to Section 7.2 for more information.

Figure 11. CSM Status and Control Register (CSMSCR)

15	14	7	6	1	0
FORCESEC	Reserved		Reserved		SECURE
R/W-1	R-0		R-10111		R-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 13. CSM Status and Control Register (CSMSCR) Field Descriptions

Bits	Field	Value	Description (1)
15	FORCESEC		Writing a 1 clears the KEY registers and secures the device.
		0	A read always returns a zero.
		1	Clears the KEY registers and secures the device. The password match flow described in Section 4.3.2 must be followed to unsecure the device again.
14-1	Reserved		Reserved
0	SECURE		Read-only bit that reflects the security state of the device.
		0	Device is unsecure (CSM unlocked).
		1	Device is secure (CSM locked).

(1) This register is EALLOW protected. Refer to Section 7.2 for more information.

4.3.1 Environments That Require Security Unlocking

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™ F28xx On-Chip Flash Programmer plug-in.
  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/program the flash.
- Executing code from on-chip unsecure memory and requiring access to secure memory for lookup table. This is not a suggested operating condition as supplying the password from external code could compromise code security.

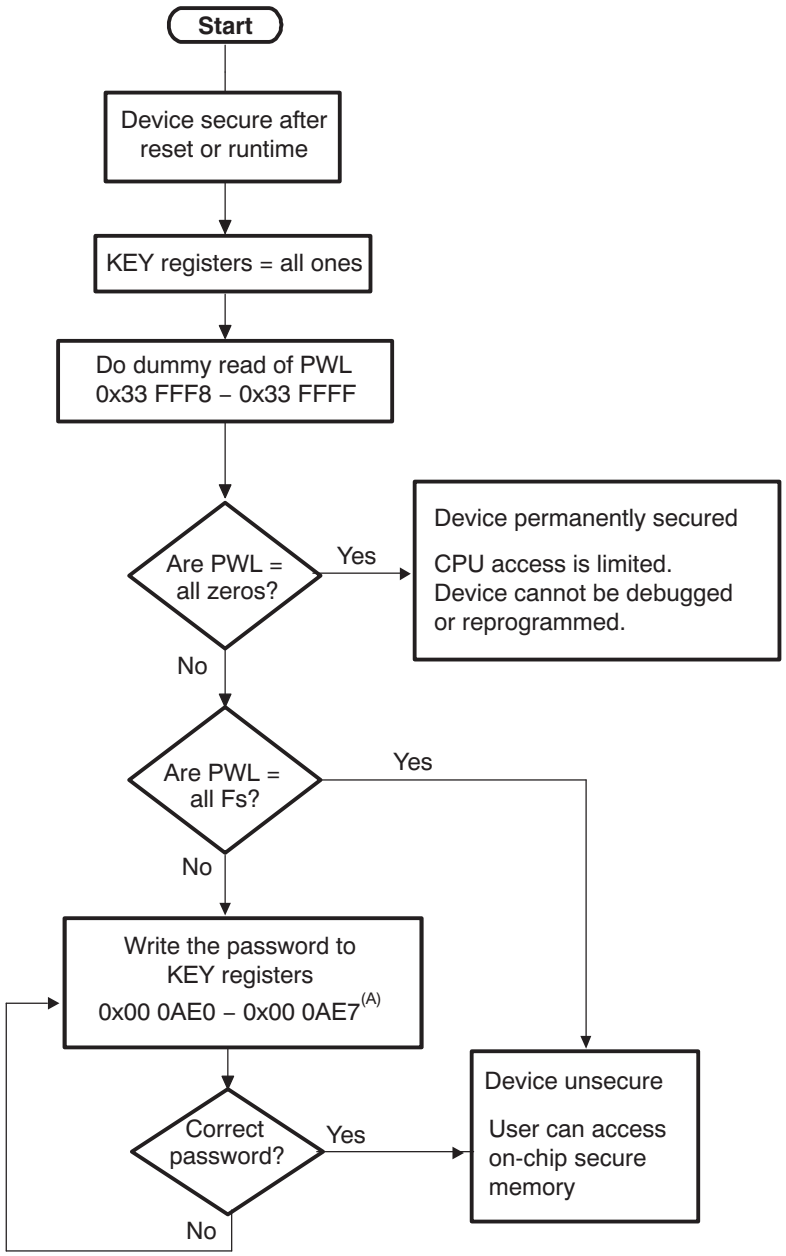
The unsecuring sequence is identical in all the above situations. This sequence is referred to as the *password match flow (PMF)* for simplicity. Figure 12 explains the sequence of operation that is required every time the user attempts to unsecure a device. A code example is listed for clarity.

4.3.2 Password Match Flow

Password match flow (PMF) is essentially a sequence of eight dummy reads from password locations (PWL) followed by eight writes to KEY registers.

Figure 12 shows how the PMF helps to initialize the security logic registers and disable security logic.

Figure 12. Password Match Flow (PMF)



A The KEY registers are EALLOW protected.

4.3.3 Unsecuring Considerations for Devices With/Without Code Security

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

Case 1: Device With Code Security

A device with code security should have a predetermined password stored in the password locations (0x33 FFF8 - 0x33 FFFF in memory). In addition, locations 0x33 FF80 - 0x33 FFF5 should be programmed with all 0x0000 and not used for program and/or data storage. The following are steps to unsecure this device:

1. Perform a dummy read of the password locations.
2. Write the password into the KEY registers (locations 0x00 0AE0 - 0x00 0AE7 in memory).
3. If the password is correct, the device becomes unsecure; otherwise, it stays secure.

Case 2: Device Without Code Security

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

1. At reset, the CSM will lock memory regions protected by the CSM.
2. Perform a dummy read of the password locations.
3. Since the password is all ones, this alone will unlock all memory regions. Secure memory is fully accessible immediately after this operation is completed.

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

4.3.3.1 C Code Example to Unsecure

```
volatile int *CSM = (volatile int *)0x000AE0; //CSM register file
volatile int *PWL = (volatile int *)0x0033FFF8; //Password location
volatile int tmp;
int I;
   // Read the 128-bits of the password locations (PWL)
   // in flash at address 0x33 FFF8 - 0x33 FFFF
   // If the device is secure, then the values read will
   // not actually be loaded into the temp variable, so
   // this is called a dummy read.
for (I=0; i<8; I++) tmp = *PWL++;
   // If the password locations (PWL) are all = ones (0xFFFF),
   // then the device 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 KEY registers
   // If this password matches that stored in the
   // PWL then the CSM will become unsecure. If it does not
   // match, then the device will remain secure.
   // An example password of:
   // 0x11112222333344445555666677778888 is used.
asm(" EALLOW");
   // Key registers are EALLOW protected *CSM++ = 0x1111;
   // Register KEY0 at 0xAE0 *CSM++ = 0x2222;
   // Register KEY1 at 0xAE1 *CSM++ = 0x3333;
   // Register KEY2 at 0xAE2 *CSM++ = 0x4444;
   // Register KEY3 at 0xAE3 *CSM++ = 0x5555;
   // Register KEY4 at 0xAE4 *CSM++ = 0x6666;
   // Register KEY5 at 0xAE5 *CSM++ = 0x7777;
   // Register KEY6 at 0xAE6 *CSM++ = 0x8888;
   // Register KEY7 at 0xAE7
asm(" EDIS");
```

4.3.3.2 C Code Example to Resecure

```
volatile int *CSMSCR = 0x00AEF; //CSMSCR register
                                //Set FORCESEC bit
asm(" EALLOW");                 //CSMSCR register is EALLOW protected.
*CSMSCR = 0x8000;
asm("EDIS");
```

4.4 Do's and Don'ts to Protect Security Logic

4.4.1 Do's

- To keep the debug and code development phase simple, use the device in the unsecure mode; i.e., use all 128 bits as ones in the password locations (or use a password that is easy to remember). Use a password after the development phase when the code is frozen.
- Recheck the password stored in the password locations before programming the COFF file using flash utilities.
- The flow of code execution can freely toggle back and forth between secure memory and unsecure memory without compromising security. To access data variables located in secure memory when the device is secured, code execution must currently be running from secure memory.
- Program locations 0x33 FF80 - 0x33 FFF5 with 0x0000 when using the CSM.

4.4.2 Don'ts

- If code security is desired, do not embed the password in your application anywhere other than in the password locations or security can be compromised.
- Do not use 128 bits of all zeros as the password. This automatically secures the device, regardless of the contents of the KEY register. The device is not debuggable nor reprogrammable.
- Do not pull a reset during an erase operation on the flash array. This can leave either zeros or an unknown value in the password locations. If the password locations are all zero during a reset, the device will always be secure, regardless of the contents of the KEY register.
- Do not use locations 0x33 FF80 - 0x33 FFF5 to store program and/or data. These locations should be programmed to 0x0000 when using the CSM.

4.5 CSM Features - Summary

1. The flash is secured after a reset until the password match flow described in Section 4.3.2 is executed.
2. The standard way of running code out of the flash is to program the flash with the code and power up the DSP. Since instruction fetches are always allowed from secure memory, regardless of the state of the CSM, the code functions correctly even without executing the password match flow.
3. Secure memory cannot be modified by code executing from unsecure memory while the device is secured.
4. Secure memory cannot be read from any code running from unsecure memory while the device is secured.
5. Secure memory cannot be read or written to by the debugger (i.e., Code Composer Studio™) at any time that the device is secured.
6. Complete access to secure memory from both the CPU code and the debugger is granted while the device is unsecured.

5 Clocking and System Control

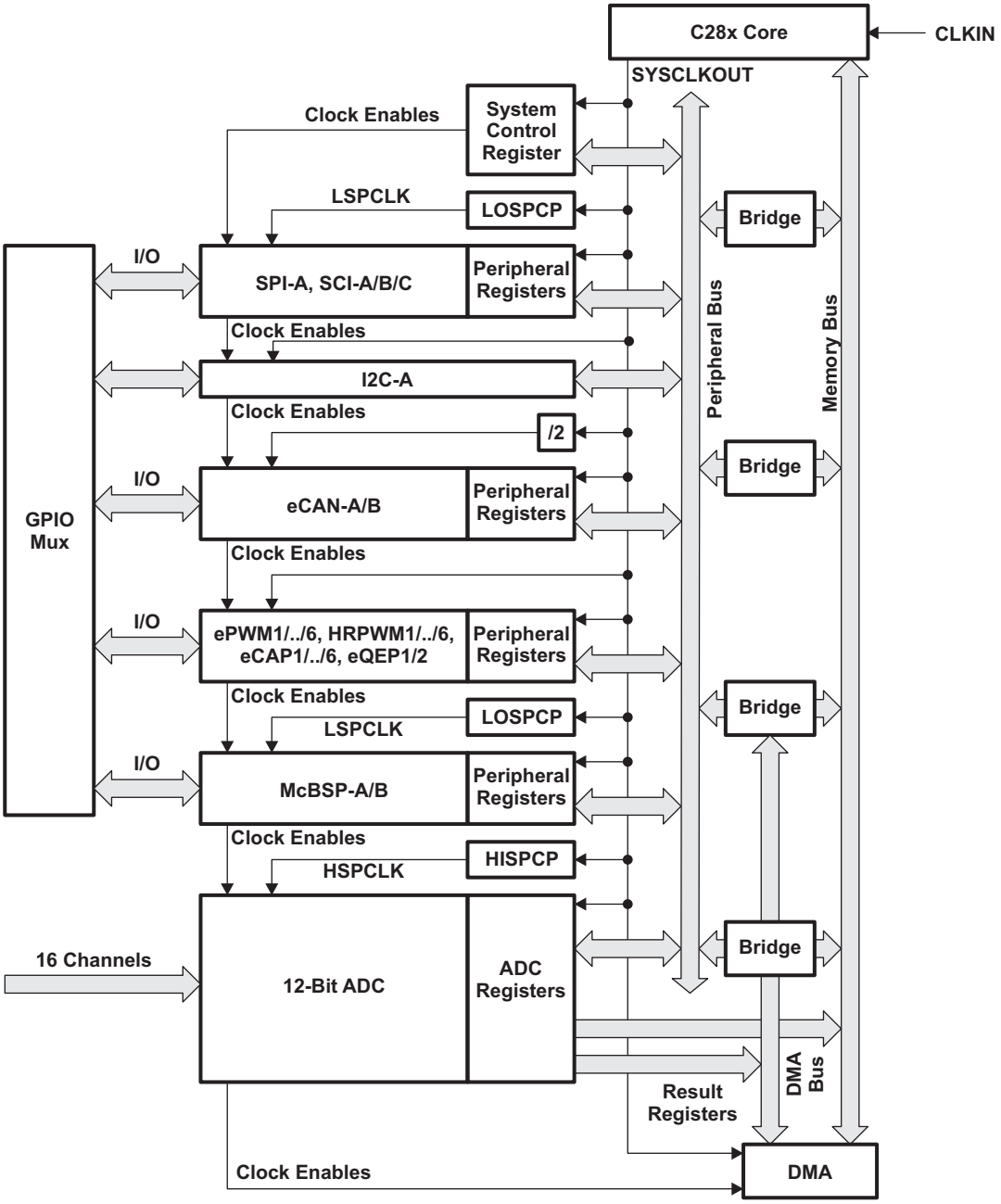
This section describes the oscillator, PLL and clocking mechanisms, the watchdog function, and the low-power modes.

5.1 Clocking

Figure 13 shows the various clock and reset domains.

The PLL, clocking, watchdog and low-power modes, are controlled by the registers listed in Table 14 .

Figure 13. Clock and Reset Domains



A CLKIN is the clock into the CPU. It is passed out of the CPU as SYSCLKOUT (that is, CLKIN is the same frequency as SYSCLKOUT).

Table 14. PLL, Clocking, Watchdog, and Low-Power Mode Registers

Name	Address	Size (x16)	Description [1]	Bit Description
PLLSTS [2]	0x7011	1	PLL Status Register	Figure 24
HISPCP	0x701A	1	High-Speed Peripheral Clock (HSPCLK) Prescaler Register	Figure 17
LOSPCP	0x701B	1	Low-Speed Peripheral Clock (LSPCLK) Prescaler Register	Figure 18
PCLKCR0	0x701C	1	Peripheral Clock Control Register 0	Figure 14
PCLKCR1	0x701D	1	Peripheral Clock Control Register 1	Figure 15
LPMCR0	0x701E	1	Low Power Mode Control Register 0	Figure 18
PCLKCR3	0x7020	1	Peripheral Clock Control Register 3	Figure 16
PLLCR [2]	0x7021	1	PLL Control Register	Figure 23
SCSR	0x7022	1	System Control & Status Register	Figure 27
WDCNTR	0x7023	1	Watchdog Counter Register.	Figure 28
WDKEY	0x7025	1	Watchdog Reset Key Register	Figure 29
WDCR	0x7029	1	Watchdog Control Register	Figure 30

(1) All of the registers in this table are EALLOW protected. See Section 7.2 for more information.
(2) The PLL control register (PLLCR) and PLL Status Register (PLLSTS) are reset to a known state by the $\overline{XRS}$ signal or a watchdog reset only. A reset issued by the debugger or the missing clock detect logic have no effect.

5.1.1 Enabling/Disabling Clocks to the Peripheral Modules

The PCLKCR0 /1/3 registers enable/disable clocks to the various peripheral modules. There is a 2-SYSCLKOUT cycle delay from when a write to the PCLKCR0 /1/3 registers occurs to when the action is valid. This delay must be taken into account before attempting to access the peripheral configuration registers. Due to the peripheral/GPIO MUXing, all peripherals cannot be used at the same time. While it is possible to turn on the clocks to all the peripherals at the same time, such a configuration is not useful. If this is done, the current drawn will be more than required. To avoid this, only enable the clocks required by the application.

Figure 14. Peripheral Clock Control 0 Register (PCLKCR0)

15	14	13	12	11	10	9	8
ECANBENCLK	ECANAENCLK	MCBSPBENCLK	MCBSPAENCLK	SCIBENCLK	SCIAENCLK	Reserved	SPIAENCLK
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0
7	6	5	4	3	2	1	0
Reserved		SCICENCLK	I2CAENCLK	ADCENCLK	TBCLKSYNC	Reserved	
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 15. Peripheral Clock Control 0 Register (PCLKCR0) Field Descriptions

Bit	Field	Value	Description [1]
15	ECANBENCLK		ECAN-B Clock enable
		0	The eCAN-B module is not clocked. (default) [2]
		1	The eCAN-B module is clocked (SYSCLKOUT/2).
14	ECANAENCLK		ECAN-A clock enable
		0	The eCAN-A module is not clocked. (default) [2]
		1	The eCAN-A module is clocked (SYSCLKOUT/2).

(1) This register is EALLOW protected. See Section 7.2 for more information.
(2) If a peripheral block is not used, the clock to that peripheral can be turned off to minimize power consumption.

Table 15. Peripheral Clock Control 0 Register (PCLKCR0) Field Descriptions (continued)

Bit	Field	Value	Description [1]
13	MCBSPBENCLK		McBSP-B Clock Enable. This bit is reserved on devices without the McBSP-B module.[3]
		0	The McBSP-B module is not clocked. (default)
		1	The McBSP-B module is clocked by the low-speed clock (LSPCLK).
12	MCBSPAENCLK		McBSP-A Clock Enable
		0	The McBSP-A module is not clocked. (default)
		1	The McBSP-A module is clocked by the low-speed clock (LSPCLK).
11	SCIBENCLK		SCI-B clock enable
		0	SCI-B module is not clocked. (default) [2]
		1	The SCI-B module is clocked by the low-speed clock (LSPCLK).
10	SCIAENCLK		SCI-A clock enable
		0	The SCI-A module is not clocked. (default) [2]
		1	The SCI-A module is clocked by the low-speed clock (LSPCLK).
9	Reserved	0	Reserved
8	SPIAENCLK		SPI-A clock enable
		0	The SPI-A module is not clocked. (default) [2]
		1	The SPI-A module is clocked by the low-speed clock (LSPCLK).
7:6	Reserved	0	Reserved
5	SCICENCLK		SCI-C clock enable. This bit is reserved on devices without the SCI-C module.[3]
		0	The SCI-C module is not clocked. (default)
		1	The SCI-C module is clocked by the low-speed clock (LSPCLK).
4	I2CAENCLK		I2C clock enable
		0	The I2C module is not clocked. (default) [4]
		1	The I2C module is clocked by SYSCLKOUT.
3	ADCENCLK		ADC clock enable
		0	The ADC is not clocked. (default) [4]
		1	The ADC module is clocked by the high-speed clock (HSPCLK)
2	TBCLKSYNC		ePWM Module Time Base Clock (TBCLK) Sync: Allows the user to globally synchronize all enabled ePWM modules to the time base clock (TBCLK):
		0	The TBCLK (Time Base Clock) within each enabled ePWM module is stopped. (default). If, however, the ePWM clock enable bit is set in the PCLKCR1 register, then the ePWM module will still be clocked by SYSCLKOUT even if TBCLKSYNC is 0.
		1	All enabled ePWM module clocks are started with the first 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 ePWM clocks is as follows: • Enable ePWM module clocks in the PCLKCR1 register. • Set TBCLKSYNC to 0. • Configure prescaler values and ePWM modes. • Set TBCLKSYNC to 1.
1-0	Reserved		Reserved

(3) On devices without a particular peripheral, the clock selection bit is reserved. On these devices, the bit should not be written to with a 1.
(4) If a peripheral block is not used, the clock to that peripheral can be turned off to minimize power consumption.

Figure 15. Peripheral Clock Control 1 Register (PCLKCR1)

15	14	13	12	11	10	9	8
EQEP2ENCLK	EQEP1ENCLK	ECAP6ENCLK	ECAP5ENCLK	ECAP4ENCLK	ECAP3ENCLK	ECAP2ENCLK	ECAP1ENCLK
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
Reserved		EPWM6ENCLK	EPWM5ENCLK	EPWM4ENCLK	EPWM3ENCLK	EPWM2ENCLK	EPWM1ENCLK
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 16. Peripheral Clock Control 1 Register (PCLKCR1) Field Descriptions

Bits	Field	Value	Description [1]
15	EQEP2ENCLK		eQEP2 clock enable
		0	The eQEP2 module is not clocked. (default) [2]
		1	The eQEP2 module is clocked by the system clock (SYSCLKOUT).
14	EQEP1ENCLK		eQEP1 clock enable
		0	The eQEP1 module is not clocked. (default) [2]
		1	The eQEP1 module is clocked by the system clock (SYSCLKOUT).
14	EQEP1ENCLK		eQEP1 clock enable
		0	The eQEP1 module is not clocked. (default) [2]
		1	The eQEP1 module is clocked by the system clock (SYSCLKOUT).
13-9	Reserved		
13	ECAP6ENCLK		eCAP6 clock enable. This bit is reserved on devices without the eCAP6 module.
		0	The eCAP6 module is not clocked. (default)
		1	The eCAP6 module is clocked by the system clock (SYSCLKOUT).
12	ECAP5ENCLK		eCAP5 clock enable. This bit is reserved on devices without the eCAP5 module.
		0	The eCAP5 module is not clocked. (default)
		1	The eCAP5 module is clocked by the system clock (SYSCLKOUT).
11	ECAP4ENCLK		eCAP4 clock enable
		0	The eCAP4 module is not clocked. (default) [2]
		1	The eCAP4 module is clocked by the system clock (SYSCLKOUT).
10	ECAP3ENCLK		eCAP3 clock enable
		0	The eCAP3 module is not clocked. (default) [2]
		1	The eCAP3 module is clocked by the system clock (SYSCLKOUT).
9	ECAP2ENCLK		eCAP2 clock enable
		0	The eCAP2 module is not clocked. (default) [2]
		1	The eCAP2 module is clocked by the system clock (SYSCLKOUT).
8	ECAP1ENCLK		eCAP1 clock enable
		0	The eCAP1 module is not clocked. (default) [2]
		1	The eCAP1 module is clocked by the system clock (SYSCLKOUT).
7:6	Reserved	0	Reserved
5	EPWM6ENCLK		ePWM6 clock enable [3]
		0	The ePWM6 module is not clocked. (default) [2]
		1	The ePWM6 module is clocked by the system clock (SYSCLKOUT).
4	EPWM5ENCLK		ePWM5 clock enable [3]
		0	The ePWM5 module is not clocked. (default) [2]
		1	The ePWM5 module is clocked by the system clock (SYSCLKOUT).

(1) This register is EALLOW protected. See Section 7.2 for more information.
(2) If a peripheral block is not used, the clock to that peripheral can be turned off to minimize power consumption.
(3) To start the ePWM Time-base clock (TBCLK) within the ePWM modules, the TBCLKSYNC bit in PCLKCR0 must also be set.

Table 16. Peripheral Clock Control 1 Register (PCLKCR1) Field Descriptions (continued)

Bits	Field	Value	Description [1]
3	EPWM4ENCLK		ePWM4 clock enable. [3]
		0	The ePWM4 module is not clocked. (default) [2]
		1	The ePWM4 module is clocked by the system clock (SYSCLKOUT).
2	EPWM3ENCLK		ePWM3 clock enable. [3]
		0	The ePWM3 module is not clocked. (default) [2]
		1	The ePWM3 module is clocked by the system clock (SYSCLKOUT).
1	EPWM2ENCLK		ePWM2 clock enable. [3]
		0	The ePWM2 module is not clocked. (default) [2]
		1	The ePWM2 module is clocked by the system clock (SYSCLKOUT).
0	EPWM1ENCLK		ePWM1 clock enable. [3]
		0	The ePWM1 module is not clocked. (default) [2]
		1	The ePWM1 module is clocked by the system clock (SYSCLKOUT).

Figure 16. Peripheral Clock Control 3 Register (PCLKCR3)

15	14	13	12	11	10	9	8
Reserved		GPIOINENCLK	XINTFENCLK	DMAENCLK	CPUTIMER2ENCLK	CPUTIMER1ENCLK	CPUTIMER0ENCLK
R-0		R/W-1	R/W-0	R/W-0	R/W-1	R/W-1	R/W-1

7	0
Reserved	
R-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 17. Peripheral Clock Control 3 Register (PCLKCR3) Field Descriptions

Bit	Field	Value	Description
15:`14	Reserved		Reserved
13	GPIOINENCLK		GPIO Input Clock Enable
		0	The GPIO module is not clocked.
		1	The GPIO module is clocked.
12	XINTFENCLK		External Interface (XINTF) Clock Enable
		0	The external memory interface is not clocked.
		1	The external memory interface is clocked.
11	DMAENCLK		DMA Clock Enable
		0	The DMA module is not clocked.
		1	The DMA module is clocked.
10	CPUTIMER2ENCLK		CPU Timer 2 Clock Enable
		0	The CPU Timer 2 is not clocked.
		1	The CPU Timer 2 is clocked.
9	CPUTIMER1ENCLK		CPU Timer 1 Clock Enable
		0	The CPU Timer 1 is not clocked.
		1	The CPU Timer 1 is clocked.
8	CPUTIMER0ENCLK		CPU Timer 0 Clock Enable
		0	The CPU Timer 0 is not clocked.
		1	The CPU Timer 0 is clocked.
7:0	Reserved		Reserved

The high speed peripheral and low speed peripheral clock prescale (HISPCP and LOSPCP) registers are used to configure the high- and low-speed peripheral clocks, respectively. See Figure 17 for the HISPCP bit layout and Figure 18 for the LOSPCP layout.

Figure 17. High-Speed Peripheral Clock Prescaler (HISPCP) Register

15	3	2	0
Reserved		HSPCLK	
R-0		R/W-001	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18. High-Speed Peripheral Clock Prescaler (HISPCP) Field Descriptions

Bits	Field	Value	Description (1)
15-3	Reserved		Reserved
2-0	HSPCLK		These bits configure the high-speed peripheral clock (HSPCLK) rate relative to SYSCLKOUT:
			If HISPCP(2) ≠ 0, HSPCLK = SYSCLKOUT/(HISPCP X 2)
			If HISPCP = 0, HSPCLK = SYSCLKOUT
		000	High speed clock = SYSCLKOUT/1
		001	High speed clock = SYSCLKOUT/2 (reset default)
		010	High speed clock = SYSCLKOUT/4
		011	High speed clock = SYSCLKOUT/6
		100	High speed clock = SYSCLKOUT/8
		101	High speed clock = SYSCLKOUT/10
		110	High speed clock = SYSCLKOUT/12
		111	High speed clock = SYSCLKOUT/14

(1) This register is EALLOW protected. See Section 7.2 for more information.
(2) HISPCP in this equation denotes the value of bits 2:0 in the HISPCP register.

Figure 18. Low-Speed Peripheral Clock Prescaler Register (LOSPCP)

15	3	2	0
Reserved		LSPCLK	
R-0		R/W-010	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 19. Low-Speed Peripheral Clock Prescaler Register (LOSPCP) Field Descriptions

Bits	Field	Value	Description (1)
15-3	Reserved		Reserved
2-0	LSPCLK		These bits configure the low-speed peripheral clock (LSPCLK) rate relative to SYSCLKOUT:
			If LOSPCP(2) ≠ 0, then LSPCLK = SYSCLKOUT/(LOSPCP X 2)
			If LOSPCP = 0, then LSPCLK = SYSCLKOUT
		000	Low speed clock = SYSCLKOUT/1
		001	Low speed clock= SYSCLKOUT/2
		010	Low speed clock= SYSCLKOUT/4 (reset default)
		011	Low speed clock= SYSCLKOUT/6
		100	Low speed clock= SYSCLKOUT/8
		101	Low speed clock= SYSCLKOUT/10
		110	Low speed clock= SYSCLKOUT/12
		111	Low speed clock= SYSCLKOUT/14

(1) This register is EALLOW protected. See Section 7.2 for more information.
(2) LOSPCP in this equation denotes the value of bits 2:0 in the LOSPCP register.

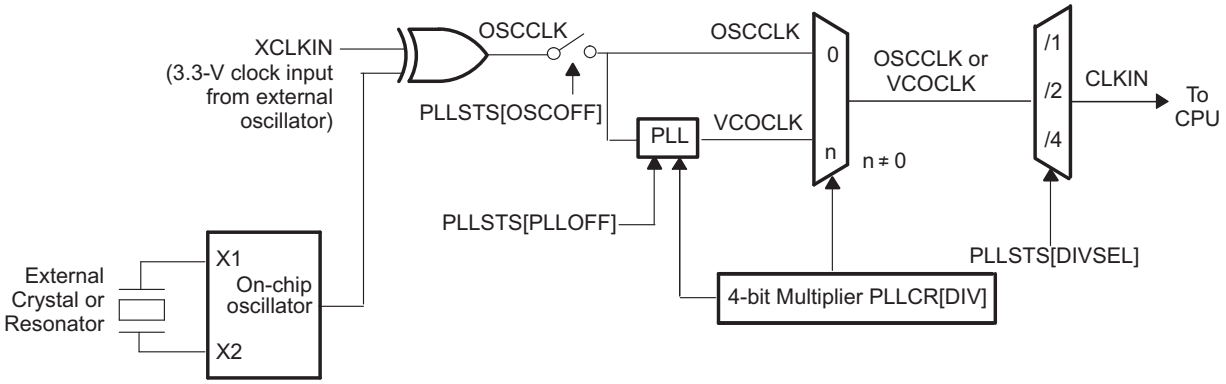
5.2 OSC and PLL Block

The on-chip oscillator and phase-locked loop (PLL) block provides the clocking signals for the device, as well as control for low-power mode (LPM) entry.

5.2.1 PLL-Based Clock Module

The 2833x, 2823x devices have an on-chip, PLL-based clock module. The PLL has a 4-bit ratio control to select different CPU clock rates. Figure 19 shows the OSC and PLL block.

Figure 19. OSC and PLL Block



The PLL-based clock module provides two modes of operation:

- **Crystal/Resonator Operation:**

  The on-chip oscillator enables the use of an external crystal/resonator to be attached to the device to provide the time base to the device. The crystal/resonator is connected to the X1/X2 pins and XCLKIN is tied low.

- **External clock source operation:**

  If the on-chip oscillator is not used, this mode allows the internal oscillator to be bypassed. The device clocks are generated from an external clock source input on either the X1 or the XCLKIN pin.

  Option 1: External clock on the XCLKIN pin:

  When using XCLKIN as the external clock source, you must tie X1 low and leave X2 disconnected. In this case, an external oscillator clock is connected to the XCLKIN pin, which allows for a 3.3-V clock source to be used.

  Option 2: External clock on the X1 pin:

  When using X1 as the clock source, you must tie XCLKIN low and leave X2 disconnected. In this case, an external oscillator clock is connected to the X1 pin, which allows for a 1.8-V clock source to be used.

The OSC circuit enables attachment of a crystal using the X1 and X2 pins. If a crystal is not used, then an external oscillator can be directly connected to the XCLKIN pin, the X2 pin is left unconnected, and the X1 pin is tied low. See the *TMS320F28335, TMS320F28334, TMS28332, TMS320F28235, TMS320F28234, TMS28232 Digital Signal Controllers (DSCs) Data Manual* (literature number SPRS439). The input clock and PLLCR[DIV] bits should be chosen in such a way that the output frequency of the PLL (VCOCLK) does not exceed 300 MHz .

Table 20. Possible PLL Configuration Modes

PLL Mode	Remarks	PLLSTS[DIVSEL] (1)	SYSCLKOUT
PLL Off	Invoked by the user setting the PLLOFF bit in the PLLSTS register. The PLL block is disabled in this mode. This can be useful to reduce system noise and for low power operation. The PLLCR register must first be set to 0x0000 (PLL Bypass) before entering this mode. The CPU clock (CLKIN) is derived directly from the input clock on either X1/X2, X1 or XCLKIN.	0, 1 2 3	OSCCLK/4 OSCCLK/2 OSCCLK/1
PLL Bypass	PLL Bypass is the default PLL configuration upon power-up or after an external reset ($\overline{XRS}$). This mode is selected when the PLLCR register is set to 0x0000 or while the PLL locks to a new frequency after the PLLCR register has been modified. In this mode, the PLL itself is bypassed but the PLL is not turned off.	0, 1 2 3	OSCCLK/4 OSCCLK/2 OSCCLK/1
PLL Enabled	Achieved by writing a non-zero value n into the PLLCR register. Upon writing to the PLLCR, the device will switch to PLL Bypass mode until the PLL locks.	0, 1 2	OSCCLK*n/4 OSCCLK*n/2

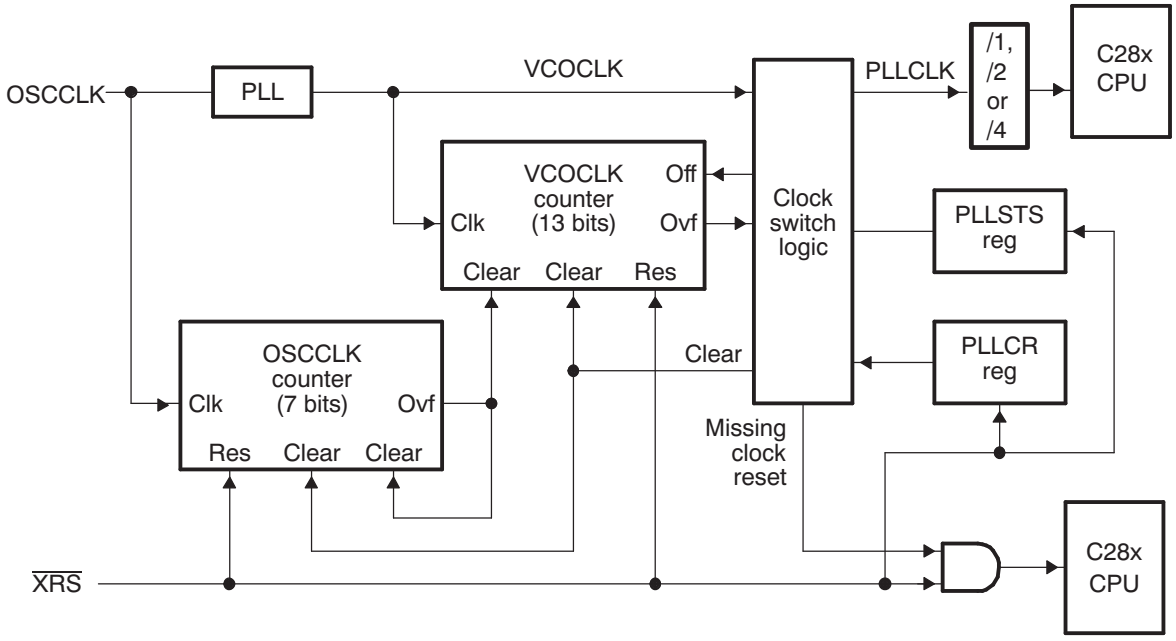
(1) PLLSTS[DIVSEL] must be 0 before writing to the PLLCR and should be changed only after PLLSTS[PLLLOCKS] = 1. See Figure 22.

5.2.2 Main Oscillator Fail Detection

Due to vibrations, it is possible for the external clock source to the DSP to become detached and fail to clock the device. When the PLL is not disabled, the main oscillator fail logic allows the device to detect this condition and default to a known state as described in this section.

Two counters are used to monitor the presence of the OSCCLK signal as shown in Figure 20. The first counter is incremented by the OSCCLK signal itself either from the X1/X2 or XCLKIN input. When the PLL is not turned off, the second counter is incremented by the VCOCLK coming out of the PLL block. These counters are configured such that when the 7-bit OSCCLK counter overflows, it clears the 13-bit VCOCLK counter. In normal operating mode, as long as OSCCLK is present, the VCOCLK counter will never overflow.

Figure 20. Oscillator Fail-Detection Logic Diagram



If the OSCCLK input signal is missing, then the PLL will output a default "limp mode" frequency and the VCOCLK counter will continue to increment. Since the OSCCLK signal is missing, the OSCCLK counter will not increment and, therefore, the VCOCLK counter is not periodically cleared. Eventually, the VCOCLK counter overflows and, if required, the device switches the CLKIN input to the CPU to the limp mode output frequency of the PLL.

When the VCOCLK counter overflows, the missing clock detection logic resets the CPU, peripherals, and other device logic. The reset generated is known as a missing clock detect logic reset ($\overline{\text{MCLKRES}}$). The $\overline{\text{MCLKRES}}$ is an internal reset only. The external $\overline{\text{XRS}}$ pin of the device is not pulled low by $\overline{\text{MCLKRES}}$ and the PLLCR and PLLSTS registers are not reset.

In addition to resetting the device, the missing oscillator logic sets the PLLSTS[MCLKSTS] register bit. When the MCLKCSTS bit is 1, this indicates that the missing oscillator detect logic reset the part and that the CPU is now running either at or one-half of the limp mode frequency.

Software should check the PLLSTS[MCLKSTS] bit after a reset to determine if the device was reset by $\overline{\text{MCLKRES}}$ due to a missing clock condition. If MCLKSTS is set, then the firmware should take the action appropriate for the system such as a system shutdown. The missing clock status can be cleared by writing a 1 to the PLLSTS[MCLKCLR] bit. This will reset the missing clock detection circuits and counters. If OSCCLK is still missing after writing to the MCLKCLR bit, then the VCOCLK counter again overflows and the process will repeat.

NOTE: Applications in which the correct CPU operating frequency is absolutely critical should implement a mechanism by which the DSP will be held in reset should the input clocks ever fail. For example, an R-C circuit may be used to trigger the $\overline{\text{XRS}}$ pin of the DSP should the capacitor ever get fully charged. An I/O pin may be used to discharge the capacitor on a periodic basis to prevent it from getting fully charged. Such a circuit would also help in detecting failure of the flash memory and the V_{DD3VFL} rail.

The following precautions and limitations should be kept in mind:

- **Use the proper procedure when changing the PLL Control Register.**
  Always follow the procedure outlined in Figure 22 when modifying the PLLCR register.
- **Do not write to the PLLCR register when the device is operating in limp mode.**
  When writing to the PLLCR register, the device switches to the CPU's CLKIN input to OSCCLK/2. When operating after limp mode has been detected, OSCCLK may not be present and the clocks to the system will stop. Always check that the PLLSTS[MCLKSTS] bit = 0 before writing to the PLLCR register as described in Figure 22.
- **The watchdog is not functional without an external clock.**
  The watchdog is not functional and cannot generate a reset when OSCCLK is not present. No special hardware has been added to switch the watchdog to the limp mode clock should OSCCLK become missing.
- **Limp mode may not work from power up.**
  The PLL may not generate a limp mode clock if OSCCLK is missing from power-up. Only if OSCCLK is initially present will a limp mode clock be generated by the PLL.
- **Do not enter HALT low power mode when the device is operating in limp mode.**
  If you try to enter HALT mode when the device is already operating in limp mode then the device may not properly enter HALT. The device may instead enter STANDBY mode or may hang and you may not be able to exit HALT mode. For this reason, always check that the PLLSTS[MCLKSTS] bit = 0 before entering HALT mode.

The following list describes the behavior of the missing clock detect logic in various operating modes:

- **PLL by-pass mode**

  When the PLL control register is set to 0x0000, the PLL is by-passed. Depending on the state of the PLLSTS[DIVSEL] bit, OSCCLK, OSCCLK/2, or OSCCLK/4 is connected directly to the CPU's input clock, CLKIN. If the OSCCLK is detected as missing, the device will automatically switch to the PLL, set the missing clock detect status bit, and generate a missing clock reset. The device will now run at the PLL limp mode frequency or one-half of the PLL limp mode frequency.
- **PLL enabled mode**

  When the PLL control register is non-zero (PLLCR = n, where n ≠ 0x0000), the PLL is enabled. In this mode, OSCCLK*n, OSCCLK*n/2, or OSCCLK*n/4 is connected to CLKIN of the CPU. If OSCCLK is detected as missing, the missing clock detect status bit will be set and the device will generate a missing clock reset. The device will now run at one-half of the PLL limp mode frequency.
- **STANDBY low power mode**

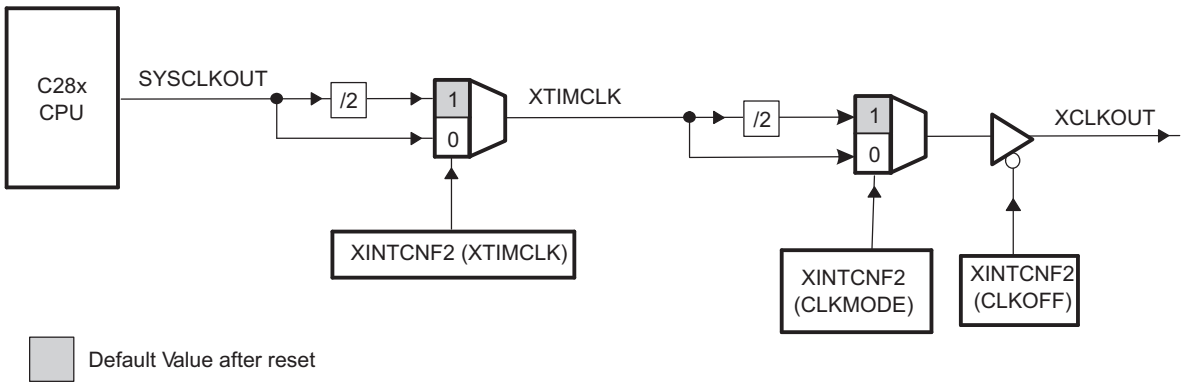
  In this mode, the CLKIN to the CPU is stopped. If a missing input clock is detected, the missing clock status bit will be set and the device will generate a missing clock reset. If the PLL is in by-pass mode when this occurs, then one-half of the PLL limp frequency will automatically be routed to the CPU. The device will now run at the PLL limp mode frequency or at one-half or one-fourth of the PLL limp mode frequency, depending on the state of the PLLSTS[DIVSEL] bit.
- **HALT low power mode**

  In HALT low power mode, all of the clocks to the device are turned off. When the device comes out of HALT mode, the oscillator and PLL will power up. The counters that are used to detect a missing input clock (VCOCLK and OSCCLK) will be enabled only after this power-up has completed. If VCOCLK counter overflows, the missing clock detect status bit will be set and the device will generate a missing clock reset. If the PLL is in by-pass mode when the overflow occurs, then one-half of the PLL limp frequency will automatically be routed to the CPU. The device will now run at the PLL limp mode frequency or at one-half or one-fourth of the PLL limp mode frequency depending on the state of the PLLSTS[DIVSEL] bit.

5.2.3 XCLKOUT Generation

The XCLKOUT signal is directly derived from the system clock SYSCLKOUT as shown in Figure 21. XCLKOUT can be either equal to, one-half, or one-fourth of SYSCLKOUT. By default, at power-up, XCLKOUT = SYSCLKOUT/4 or XCLKOUT = OSCCLK/16 .

Figure 21. XCLKOUT Generation



The XCLKOUT signal is active when reset is active. Since XCLKOUT should reflect SYSCLKOUT/4 when reset is low, you can monitor this signal to detect if the device is being properly clocked during debug. There is no internal pullup or pulldown on the XCLKOUT pin.

If XCLKOUT is not being used, it can be turned off by setting the CLKOFF bit to 1 in the XINTCNF2 register.

5.2.4 PLL Control (PLLCR) Register

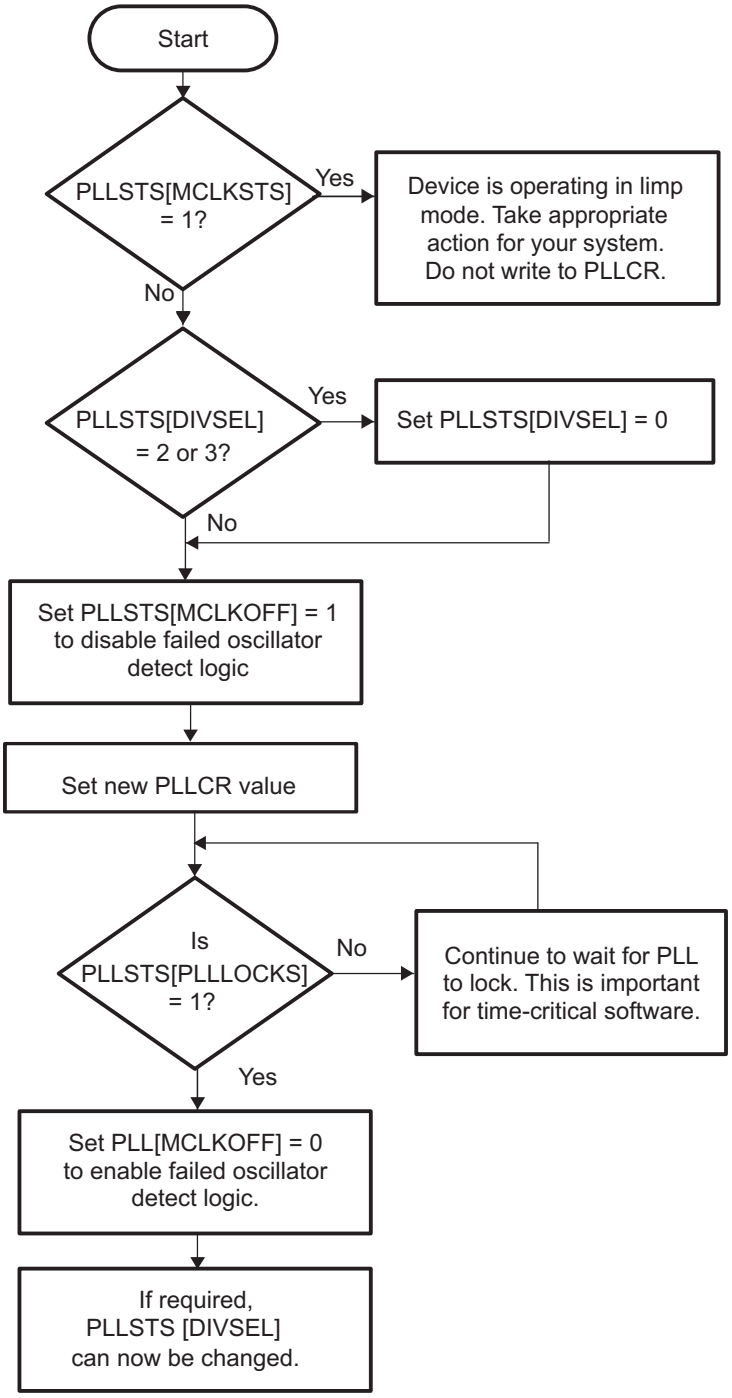
The PLLCR register is used to change the PLL multiplier of the device. Before writing to the PLLCR register, the following requirements must be met:

- The PLLSTS[DIVSEL] bit must be 0 (CLKIN divide by 4 enabled). Change PLLSTS[DIVSEL] only after the PLL has completed locking, i.e., after PLLSTS[PLLLOCKS] = 1.
- The device must not be operating in "limp mode". That is, the PLLSTS[MCLKSTS] bit must be 0.

Once the PLL is stable and has locked at the new specified frequency, the PLL switches CLKIN to the new value as shown in Table 21. When this happens, the PLLLOCKS bit in the PLLSTS register is set, indicating that the PLL has finished locking and the device is now running at the new frequency. User software can monitor the PLLLOCKS bit to determine when the PLL has completed locking. Once PLLSTS[PLLLOCKS] = 1, DIVSEL can be changed.

Follow the procedure in Figure 22 any time you are writing to the PLLCR register.

Figure 22. PLLCR Change Procedure Flow Chart



5.2.5 PLL Control, Status and XCLKOUT Register Descriptions

The DIV field in the PLLCR register controls whether the PLL is bypassed or not and sets the PLL clocking ratio when it is not bypassed. PLL bypass is the default mode after reset. Do not write to the DIV field if the PLLSTS[DIVSEL] bit is 10 or 01, or if the PLL is operating in limp mode as indicated by the PLLSTS[MCLKSTS] bit being set . See the procedure for changing the PLLCR described in Figure 22.

Figure 23. PLLCR Register Layout

15	4	3	0
Reserved		DIV	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21. PLLCR Bit Descriptions[1]

	SYSCLKOUT (CLKIN) [2]		
PLLCR[DIV] Value [3]	**PLLSTS[DIVSEL] = 0 or 1**	**PLLSTS[DIVSEL] = 2**	**PLLSTS[DIVSEL] = 3**
0000 (PLL bypass)	OSCCLK/4 (Default)	OSCCLK/2	OSCCLK
0001	(OSCCLK * 1)/4	(OSCCLK*1)/2	–
0010	(OSCCLK * 2)/4	(OSCCLK*2)/2	–
0011	(OSCCLK * 3)/4	(OSCCLK*3)/2	–
0100	(OSCCLK * 4)/4	(OSCCLK*4)/2	–
0101	(OSCCLK * 5)/4	(OSCCLK*5)/2	–
0110	(OSCCLK * 6)/4	(OSCCLK*6)/2	–
0111	(OSCCLK * 7)/4	(OSCCLK*7)/2	–
1000	(OSCCLK * 8)/4	(OSCCLK*8)/2	–
1001	(OSCCLK * 9)/4	(OSCCLK*9)/2	–
1010	(OSCCLK * 10)/4	(OSCCLK*10)/2	–
1011 - 1111	Reserved	Reserved	Reserved

(1) This register is EALLOW protected. See Section 7.2 for more information.
(2) PLLSTS[DIVSEL] must be 0 before writing to the PLLCR and should be changed only after PLLSTS[PLLLOCKS] = 1. See Figure 22.
(3) The PLL control register (PLLCR) and PLL Status Register (PLLSTS) are reset to their default state by the $\overline{\text{XRS}}$ signal or a watchdog reset only. A reset issued by the debugger or the missing clock detect logic have no effect.

Figure 24. PLL Status Register (PLLSTS)

15	9	8
Reserved		DIVSEL
R-0		R/W-0

7	6	5	4	3	2	1	0
DIVSEL	MCLKOFF	OSCOFF	MCLKCLR	MCLKSTS	PLLOFF	Reserved	PLLLOCKS
R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R-0	R-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22. PLL Status Register (PLLSTS) Field Descriptions

Bits	Field	Value	Description (1) (2)
15-9	Reserved		Reserved
8:7	DIVSEL		Divide Select: This bit selects between /4, /2, and /1 for CLKIN to the CPU. The configuration of the DIVSEL bit is as follows:
		00, 01	Select Divide By 4 for CLKIN
		10	Select Divide By 2 for CLKIN
		11	Select Divide By 1 for CLKIN. (This mode can be used only when PLL is off or bypassed.)
6	MCLKOFF		Missing clock-detect off bit
		0	Main oscillator fail-detect logic is enabled. (default)
		1	Main oscillator fail-detect logic is disabled and the PLL will not issue a limp-mode clock. Use this mode when code must not be affected by the detection circuit. For example, if external clocks are turned off.
5	OSCOFF		Oscillator Clock Off Bit
		0	The OSCCLK signal from X1, X1/X2 or XCLKIN is fed to the PLL block. (default)
		1	The OSCCLK signal from X1, X1/X2 or XCLKIN is not fed to the PLL block. This does not shut down the internal oscillator. The OSCOFF bit is used for testing the missing clock detection logic. When the OSCOFF bit is set, do not enter HALT or STANDBY modes or write to PLLCR as these operations can result in unpredictable behavior. When the OSCOFF bit is set, the behavior of the watchdog is different depending on which input clock source (X1, X1/X2 or XCLKIN) is being used: • X1 or X1/X2: The watchdog is not functional. • XCLKIN: The watchdog is functional and should be disabled before setting OSCOFF.
4	MCLKCLR		Missing Clock Clear Bit.
		0	Writing a 0 has no effect. This bit always reads 0.
		1	Forces the missing clock detection circuits to be cleared and reset. If OSCCLK is still missing, the detection circuit will again generate a reset to the system, set the missing clock status bit (MCLKSTS), and the CPU will be powered by the PLL operating at a "limp mode" frequency.
3	MCLKSTS		Missing Clock Status Bit. Check the status of this bit after a reset to determine whether a missing oscillator condition was detected. Under normal conditions, this bit should be 0. Writes to this bit are ignored. This bit will be cleared by writing to the MCLKCLR bit or by forcing an external reset.
		0	Indicates normal operation. A missing clock condition has not been detected.
		1	Indicates that OSCCLK was detected as missing. The main oscillator fail detect logic has reset the device and the CPU is now clocked by the PLL operating at the limp mode frequency.
2	PLLOFF		PLL Off Bit. This bit turns off the PLL. This is useful for system noise testing. This mode must only be used when the PLLCR register is set to 0x0000.
		0	PLL On (default)
		1	PLL Off. While the PLLOFF bit is set the PLL module will be kept powered down. The device must be in PLL bypass mode (PLLCR = 0x0000) before writing a 1 to PLLOFF. While the PLL is turned off (PLLOFF = 1), do not write a non-zero value to the PLLCR. The STANDBY and HALT low power modes will work as expected when PLLOFF = 1. After waking up from HALT or STANDBY the PLL module will remain powered down.
1	Reserved		Reserved
0	PLLLOCKS		PLL Lock Status Bit.
		0	Indicates that the PLLCR register has been written to and the PLL is currently locking. The CPU is clocked by OSCCLK/2 until the PLL locks.
		1	Indicates that the PLL has finished locking and is now stable.

(1) This register is reset to its default state only by the $\overline{\text{XRS}}$ signal or a watchdog reset. It is not reset by a missing clock or debugger reset.
(2) This register is EALLOW protected. See Section 7.2 for more information.

5.2.6 External Reference Oscillator Clock Option

TI recommends that customers have the resonator/crystal vendor characterize the operation of their device with the DSP chip. The resonator/crystal vendor has the equipment and expertise to tune the tank circuit. The vendor can also advise the customer regarding the proper tank component values to provide proper start-up and stability over the entire operating range.

5.3 Low-Power Modes Block

Table 23 summarizes the various modes.

The various low-power modes operate as shown in Table 24.

See the *TMS320F28335, TMS320F28334, TMS320F28332, TMS320F28235, TMS320F28234, TMS320F28232 Digital Signal Controllers (DSCs) Data Manual* (literature number SPRS439) for exact timing for entering and exiting the low power modes.

Table 23. Low-Power Mode Summary

Mode	LPMCR0[1:0]	OSCCLK	CLKIN	SYSCLKOUT	Exit (1)
IDLE	00	On	On	On(2)	$\overline{\text{XRS}}$, Watchdog interrupt, Any enabled interrupt
STANDBY	01	On (watchdog still running)	Off	Off	$\overline{\text{XRS}}$, Watchdog interrupt, GPIO Port A signal, Debugger(3)
HALT	1X	Off (oscillator and PLL turned off, watchdog not functional)	Off	Off	$\overline{\text{XRS}}$, GPIO Port A Signal, Debugger(3)

(1) The Exit column lists which signals or under what conditions the low power mode is exited. This signal must be kept low long enough for an interrupt to be recognized by the device. Otherwise the IDLE mode is not exited and the device goes back into the indicated low power mode.

(2) The IDLE mode on the 28x behaves differently than on the 24x/240x. On the 28x, the clock output from the CPU (SYSCLKOUT) is still functional while on the 24x/240x the clock is turned off.

(3) On the 28x, the JTAG port can still function even if the clock to the CPU (CLKIN) is turned off.

Table 24. Low Power Modes

Mode	Description
IDLE Mode:	This mode is exited by any enabled interrupt or an NMI. The LPM block itself performs no tasks during this mode.
STANDBY Mode:	If the LPM bits in the LPMCR0 register are set to 01, the device enters STANDBY mode when the IDLE instruction is executed. In STANDBY mode the clock input to the CPU (CLKIN) is disabled, which disables all clocks derived from SYSCLKOUT. The oscillator and PLL and watchdog will still function. Before entering the STANDBY mode, you should perform the following tasks: • Enable the WAKEINT interrupt in the PIE module. This interrupt is connected to both the watchdog and the low power mode module interrupt. • If desired, specify one of the GPIO port A signals to wake the device in the GPIOLPMSEL register. The GPIOLPMSEL register is part of the GPIO module. In addition to the selected GPIO signal, the $\overline{\text{XRS}}$ input and the watchdog interrupt, if enabled in the LPMCR0 register, can wake the device from the STANDBY mode. • Select the input qualification in the LPMCR0 register for the signal that will wake the device. When the selected external signal goes low, it must remain low a number of OSCCLK cycles as specified by the qualification period in the LPMCR0 register. If the signal should be sampled high during this time, the qualification will restart. At the end of the qualification period, the PLL enables the CLKIN to the CPU and the WAKEINT interrupt is latched in the PIE block. The CPU then responds to the WAKEINT interrupt if it is enabled.
HALT Mode:	If the LPM bits in the LPMCR0 register are set to 1x, the device enters the HALT mode when the IDLE instruction is executed. In HALT mode all of the device clocks, including the PLL and oscillator, are shut down. Before entering the HALT mode, you should perform the following tasks: • Enable the WAKEINT interrupt in the PIE module (PIEIER1.8 = 1). This interrupt is connected to both the watchdog and the Low-Power-Mode module interrupt. • Specify one of the GPIO port A signals to wake the device in the GPIOLPMSEL register. The GPIOLPMSEL register is part of the GPIO module. In addition to the selected GPIO signal, the XRS input can also wake the device from the HALT mode • Disable all interrupts with the possible exception of the HALT mode wakeup interrupt. The interrupts can be re-enabled after the device is brought out of HALT mode. • For device to exit HALT mode properly, the following conditions must be met: Bit 7 (INT1.8) of PIEIER1 register should be 1. Bit 0 (INT1) of IER register must be 1. • If the above conditions are met, – WAKE_INT ISR will be executed first, followed by the instruction(s) after IDLE, if INTM = 0. – WAKE_INT ISR will not be executed and instruction(s) after IDLE will be executed, if INTM = 1.

Table 24. Low Power Modes (continued)

Mode	Description
	Do not enter HALT low power mode when the device is operating in limp mode (PLLSTS[MCLKSTS] = 1). If you try to enter HALT mode when the device is already operating in limp mode then the device may not properly enter HALT. The device may instead enter STANDBY mode or may hang and you may not be able to exit HALT mode. For this reason, always check that the PLLSTS[MCLKSTS] bit = 0 before entering HALT mode. When the selected external signal goes low, it is fed asynchronously to the LPM block. The oscillator is turned on and begins to power up. You must hold the signal low long enough for the oscillator to complete power up. Once the oscillator has stabilized, the PLL lock sequence is initiated . Once the PLL has locked, it feeds the CLKIN to the CPU at which time the CPU responds to the WAKEINT interrupt if enabled.

The low-power modes are controlled by the LPMCR0 register (Figure 25).

Figure 25. Low Power Mode Control 0 Register (LPMCR0)

15	14 8	7 2	1 0
WDINTE	Reserved	QUALSTDBY	LPM
R/W-0	R-0	R/W-1	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 25. Low Power Mode Control 0 Register (LPMCR0) Field Descriptions

Bits	Field	Value	Description (1)
15	WDINTE		Watchdog interrupt enable
		0	The watchdog interrupt is not allowed to wake the device from STANDBY. (default)
		1	The watchdog is allowed to wake the device from STANDBY. The watchdog interrupt must also be enabled in the SCSR Register.
14-8	Reserved		Reserved
7-2	QUALSTDBY		Select number of OSCCLK clock cycles to qualify the selected GPIO inputs that wake the device from STANDBY mode. This qualification is only used when in STANDBY mode. The GPIO signals that can wake the device from STANDBY are specified in the GPIOLPMSEL register.
		000000	2 OSCCLKs (default)
		000001	3 OSCCLKs
		. . .	. . .
		111111	65 OSCCLKs
1-0	LPM(2)		These bits set the low power mode for the device.
		00	Set the low power mode to IDLE (default)
		01	Set the low power mode to STANDBY
		10	Set the low power mode to HALT (3)
		11	Set the low power mode to HALT (3)

(1) This register is EALLOW protected. See Section 7.2 for more information.

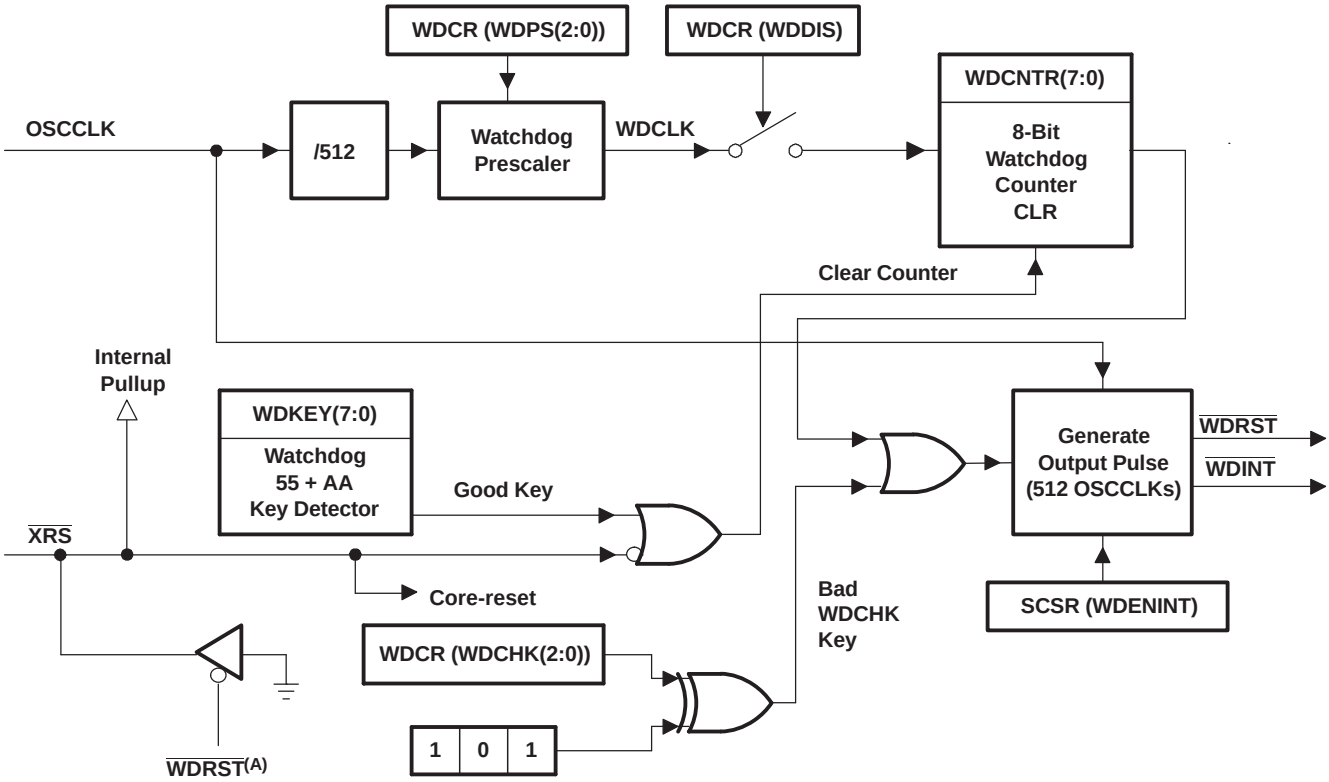
(2) The low power mode bits (LPM) only take effect when the IDLE instruction is executed. Therefore, you must set the LPM bits to the appropriate mode before executing the IDLE instruction.

(3) If you try to enter HALT mode when the device is already operating in limp mode then the device may not properly enter HALT. The device may instead enter STANDBY mode or may hang and you may not be able to exit HALT mode. For this reason, always check that the PLLSTS[MCLKSTS] bit = 0 before entering HALT mode.

5.4 Watchdog Block

The watchdog module generates an output pulse, 512 oscillator-clocks (OSCCLK) wide whenever the 8-bit watchdog up counter has reached its maximum value. To prevent this, the user can either disable the counter or the software must periodically write a 0x55 + 0xAA sequence into the watchdog key register which resets the watchdog counter. Figure 26 shows the various functional blocks within the watchdog module.

Figure 26. Watchdog Module



A The WDRST and XRS signals are driven low for 512 OSCCLK cycles when a watchdog reset occurs. Likewise, if the watchdog interrupt is enabled, the WDINT signal will be driven low for 512 OSCCLK cycles when an interrupt occurs. Watchdog is not functional and cannot generate a reset when OSCCLK is not present.

5.4.1 Servicing The Watchdog Timer

The WDCNTR is reset when the proper sequence is written to the WDKEY register before the 8-bit watchdog counter (WDCNTR) 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 26. Example Watchdog Key Sequences

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

Step 3 in Table 26 is the first action that enables the WDCNTR to be reset. The WDCNTR is not actually reset until step 6. Step 8 again re-enables the WDCNTR to be reset and step 9 resets the WDCNTR. Step 10 again re-enables the WDCNTR ro 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 (WDFLAG) in the WDCR register. After a reset, the program can read the state of this flag to determine the source of the reset. After reset, the WDFLAG should be cleared by software to allow the source of subsequent resets to be determined. Watchdog resets are not prevented when the flag is set.

5.4.2 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:**

  If the watchdog is configured to assert an interrupt, then the $\overline{\text{WDINT}}$ signal will be driven low for 512 OSCCLK cycles, causing the WAKEINT interrupt in the PIE to be taken if it is enabled in the PIE module. The watchdog interrupt is edge triggered on the falling edge of $\overline{\text{WDINT}}$. Thus, if the WAKEINT interrupt is re-enabled before $\overline{\text{WDINT}}$ goes inactive, you will not immediately get another interrupt. The next WAKEINT interrupt will occur at the next watchdog timeout.

  If the watchdog is re-configured from interrupt mode to reset mode while $\overline{\text{WDINT}}$ is still active low, then the device will reset immediately. The WDINTS bit in the SCSR register can be read to determine the current state of the $\overline{\text{WDINT}}$ signal before reconfiguring the watchdog to reset mode.

5.4.3 Watchdog Operation in Low Power Modes

In STANDBY mode, all of the clocks to the peripherals are turned off on the device. The only peripheral that remains functional is the watchdog since the watchdog module runs off the oscillator clock (OSCCLK). The $\overline{\text{WDINT}}$ signal is fed to the Low Power Modes (LPM) block so that it can be used to wake the device from STANDBY low power mode (if enabled). See the Low Power Modes Block section of the device data manual for details.

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. The watchdog is connected to the WAKEINT interrupt in the PIE.

NOTE: If the watchdog interrupt is used to wake-up from an IDLE or STANDBY low power mode condition, then make sure that the $\overline{\text{WDINT}}$ signal goes back high again before attempting to go back into the IDLE or STANDBY mode. The $\overline{\text{WDINT}}$ signal will be held low for 512 OSCCLK cycles when the watchdog interrupt is generated. You can determine the current state of $\overline{\text{WDINT}}$ 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, this feature cannot be used because the oscillator (and PLL) are turned off and, therefore, so is the watchdog.

5.4.4 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.

5.4.5 Watchdog Registers

The system control and status register (SCSR) contains the watchdog override bit and the watchdog interrupt enable/disable bit. Figure 27 describes the bit functions of the SCSR register.

Figure 27. System Control and Status Register (SCSR)

15	8
Reserved	
R-0	

7	3	2	1	0
Reserved		WDINTS	WDENINT	WDOVERRIDE
R-0		R-1	R/W-0	R/W1C-1

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 27. System Control and Status Register (SCSR) Field Descriptions

Bit	Field	Value	Description (1)
15-3	Reserved		
2	WDINTS		Watchdog interrupt status bit. WDINTS reflects the current state of the $\overline{\text{WDINT}}$ signal from the watchdog block. WDINTS follows the state of $\overline{\text{WDINT}}$ by two SYSCLKOUT cycles. If the watchdog interrupt is used to wake the device from IDLE or STANDBY low power mode, use this bit to make sure $\overline{\text{WDINT}}$ is not active before attempting to go back into IDLE or STANDBY mode.
		0	Watchdog interrupt signal ($\overline{\text{WDINT}}$) is active.
		1	Watchdog interrupt signal ($\overline{\text{WDINT}}$) is not active.
1	WDENINT		Watchdog interrupt enable.
		0	The watchdog reset ($\overline{\text{WDRST}}$) output signal is enabled and the watchdog interrupt ($\overline{\text{WDINT}}$) output signal is disabled. This is the default state on reset ($\overline{\text{XRS}}$). When the watchdog interrupt occurs the $\overline{\text{WDRST}}$ signal will stay low for 512 OSCCLK cycles. If the WDENINT bit is cleared while $\overline{\text{WDINT}}$ is low, a reset will immediately occur. The WDINTS bit can be read to determine the state of the $\overline{\text{WDINT}}$ signal.
		1	The $\overline{\text{WDRST}}$ output signal is disabled and the $\overline{\text{WDINT}}$ output signal is enabled. When the watchdog interrupt occurs, the $\overline{\text{WDINT}}$signal will stay low for 512 OSCCLK cycles. If the watchdog interrupt is used to wake the device from IDLE or STANDBY low power mode, use the WDINTS bit to make sure $\overline{\text{WDINT}}$ is not active before attempting to go back into IDLE or STANDBY mode.
0	WDOVERRIDE		Watchdog override
		0	Writing a 0 has no effect. If this bit is cleared, it remains in this state until a reset occurs. The current state of this bit is readable by the user.
		1	You can change the state of the watchdog disable (WDDIS) bit in the watchdog control (WDCR) register. If the WDOVERRIDE bit is cleared by writing a 1, you cannot modify the WDDIS bit.

(1) This register is EALLOW protected. See Section 7.2 for more information.

Figure 28. Watchdog Counter Register (WDCNTR)

15	8	7	0
Reserved		WDCNTR	
R-0		R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 28. Watchdog Counter Register (WDCNTR) Field Descriptions

Bits	Field	Description
15-8	Reserved	Reserved
7-0	WDCNTR	These bits contain the current value of the WD counter. The 8-bit counter continually increments at the watchdog clock (WDCLK), rate. If the counter overflows, then the watchdog initiates a reset. If the WDKEY register is written with a valid combination, then the counter is reset to zero. The watchdog clock rate is configured in the WDCR register.

Figure 29. Watchdog Reset Key Register (WDKEY)

15	8	7	0
Reserved		WDKEY	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 29. Watchdog Reset Key Register (WDKEY) Field Descriptions

Bits	Field	Value	Description (1)
15-8	Reserved		Reserved
7-0	WDKEY		Refer to Table 26 for examples of different WDKEY write sequences.
		0x55 + 0xAA	Writing 0x55 followed by 0xAA to WDKEY causes the WDCNTR bits to be cleared.
		Other value	Writing any value other than 0x55 or 0xAA causes no action to be generated. If any value other than 0xAA is written after 0x55, then the sequence must restart with 0x55.
			Reads from WDKEY return the value of the WDCR register.

(1) This register is EALLOW protected. See Section 7.2 for more information.

Figure 30. Watchdog Control Register (WDCR)

15	8
Reserved	

7	6	5	3	2	0
WDFLAG	WDDIS	WDCHK		WDPS	
R/W1C-0	R/W-0	R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 30. Watchdog Control Register (WDCR) Field Descriptions

Bits	Field	Value	Description (1)
15-8	Reserved		Reserved
7	WDFLAG		Watchdog reset status flag bit
		0	The reset was caused either by the $\overline{XRS}$ pin or because of power-up. The bit remains latched until you write a 1 to clear the condition. Writes of 0 are ignored.
		1	Indicates a watchdog reset ($\overline{WDRST}$) generated the reset condition. .
6	WDDIS		Watchdog disable. On reset, the watchdog module is enabled.
		0	Enables the watchdog module. WDDIS can be modified only if the WDOVERRIDE bit in the SCSR register is set to 1. (default)
		1	Disables the watchdog module.

(1) This register is EALLOW protected. See Section 7.2 for more information.

Table 30. Watchdog Control Register (WDCR) Field Descriptions (continued)

Bits	Field	Value	Description (1)
5-3	WDCHK		Watchdog check.
		0,0,0	You must ALWAYS write 1,0,1 to these bits whenever a write to this register is performed unless the intent is to reset the device via software.
		other	If the watchdog is enabled, then writing any other value causes an immediate device reset or watchdog interrupt to be taken. These three bits always read back as zero (0, 0, 0). This feature can be used to generate a software reset of the DSP.
2-0	WDPS		Watchdog pre-scale. These bits configure the watchdog counter clock (WDCLK) rate relative to OSCCLK/512:
		000	WDCLK = OSCCLK/512/1 (default)
		001	WDCLK = OSCCLK/512/1
		010	WDCLK = OSCCLK/512/2
		011	WDCLK = OSCCLK/512/4
		100	WDCLK = OSCCLK/512/8
		101	WDCLK = OSCCLK/512/16
		110	WDCLK = OSCCLK/512/32
		111	WDCLK = OSCCLK/512/64

When the $\overline{\text{XRS}}$ line is low, the WDFLAG bit is forced low. The WDFLAG bit is only set if a rising edge on $\overline{\text{WDRST}}$ signal is detected (after synch and an 8192 SYSCLKOUT cycle delay) and the $\overline{\text{XRS}}$ signal is high. If the $\overline{\text{XRS}}$ signal is low when $\overline{\text{WDRST}}$ goes high, then the WDFLAG bit remains at 0. In a typical application, the $\overline{\text{WDRST}}$ signal connects to the $\overline{\text{XRS}}$ input. Hence to distinguish between a watchdog reset and an external device reset, an external reset must be longer in duration then the watchdog pulse.

5.5 32-Bit CPU Timers 0/1/2

This section describes the three 32-bit CPU timers (Figure 31) (TIMER0/1/2).

CPU-Timer 0 and CPU-Timer 1 can be used in user applications. Timer 2 is reserved for DSP/BIOS. If the application is not using DSP/BIOS, then Timer 2 can be used in the application.

The CPU-timer interrupt signals (TINT0, TINT1, TINT2) are connected as shown in Figure 32.

Figure 31. CPU Timers

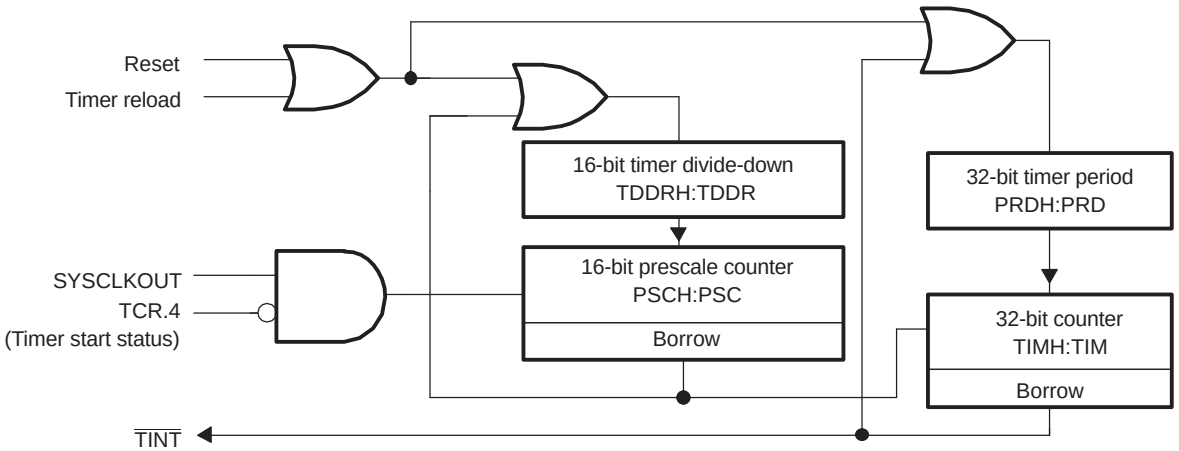
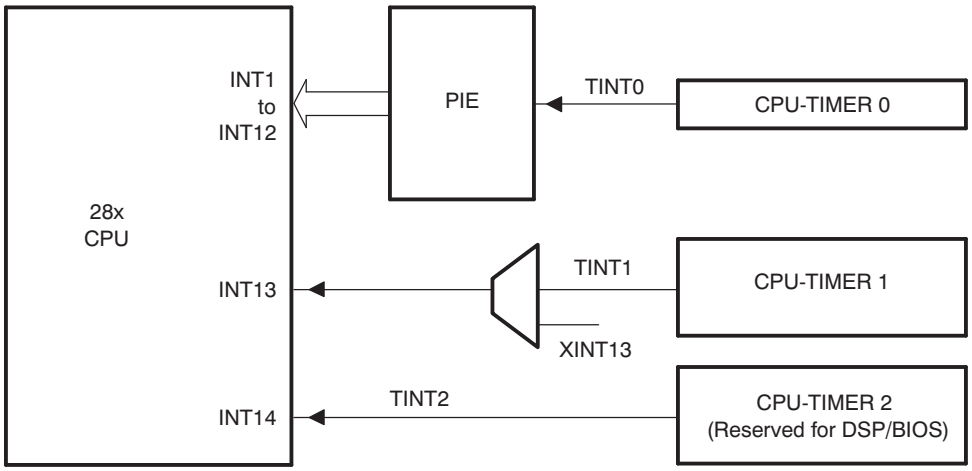


Figure 32. CPU-Timer Interrupt Signals and Output Signal



A The timer registers are connected to the Memory Bus of the 28x processor.

B The timing of the timers is synchronized to SYSCLKOUT of the processor clock.

The general operation of the CPU timer is as follows: The 32-bit counter register TIMH:TIM is loaded with the value in the period register PRDH:PRD. The counter decrements once every (TPR[TDDRH:TDDR]+1) SYSCLKOUT cycles where TDDRH:TDDR is the timer divider. When the counter reaches 0, a timer interrupt output signal generates an interrupt pulse. The registers listed in Table 31 are used to configure the timers.

Table 31. CPU Timers 0, 1, 2 Configuration and Control Registers

Name	Address	Size (x16)	Description	Bit Description
TIMER0TIM	0x0C00	1	CPU-Timer 0, Counter Register	Figure 33
TIMER0TIMH	0x0C01	1	CPU-Timer 0, Counter Register High	Figure 34
TIMER0PRD	0x0C02	1	CPU-Timer 0, Period Register	Figure 35
TIMER0PRDH	0x0C03	1	CPU-Timer 0, Period Register High	Figure 36
TIMER0TCR	0x0C04	1	CPU-Timer 0, Control Register	Figure 37
Reserved	0x0C05	1		
TIMER0TPR	0x0C06	1	CPU-Timer 0, Prescale Register	Figure 38
TIMER0TPRH	0x0C07	1	CPU-Timer 0, Prescale Register High	Figure 39
TIMER1TIM	0x0C08	1	CPU-Timer 1, Counter Register	Figure 33
TIMER1TIMH	0x0C09	1	CPU-Timer 1, Counter Register High	Figure 34
TIMER1PRD	0x0C0A	1	CPU-Timer 1, Period Register	Figure 35
TIMER1PRDH	0x0C0B	1	CPU-Timer 1, Period Register High	Figure 36
TIMER1TCR	0x0C0C	1	CPU-Timer 1, Control Register	Figure 37
Reserved	0x0C0D	1		
TIMER1TPR	0x0C0E	1	CPU-Timer 1, Prescale Register	Figure 38
TIMER1TPRH	0x0C0F	1	CPU-Timer 1, Prescale Register High	Figure 39
TIMER2TIM	0x0C10	1	CPU-Timer 2, Counter Register	Figure 33
TIMER2TIMH	0x0C11	1	CPU-Timer 2, Counter Register High	Figure 34
TIMER2PRD	0x0C12	1	CPU-Timer 2, Period Register	Figure 35
TIMER2PRDH	0x0C13	1	CPU-Timer 2, Period Register High	Figure 36
TIMER2TCR	0x0C14	1	CPU-Timer 2, Control Register	Figure 37
Reserved	0x0C15	1		
TIMER2TPR	0x0C16	1	CPU-Timer 2, Prescale Register	Figure 38
TIMER2TPRH	0x0C17	1	CPU-Timer 2, Prescale Register High	Figure 39

Figure 33. TIMERxTIM Register (x = 0, 1, 2)

15	0
TIM	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 32. TIMERxTIM Register Field Descriptions

Bits	Field	Description
15-0	TIM	CPU-Timer Counter Registers (TIMH:TIM): The TIM register holds the low 16 bits of the current 32-bit count of the timer. 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 divide-down 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.

Figure 34. TIMERxTIMH Register (x = 0, 1, 2)

15	0
TIMH	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 33. TIMERxTIMH Register Field Descriptions

Bits	Field	Description
15-0	TIMH	See description for TIMERxTIM.

Figure 35. TIMERxPRD Register (x = 0, 1, 2)

15	0
PRD	
R/W-0xFFFF	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 34. TIMERxPRD Register Field Descriptions

Bits	Field	Description
15-0	PRD	CPU-Timer Period Registers (PRDH:PRD): The PRD register holds the low 16 bits of the 32-bit period. 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).

Figure 36. TIMERxPRDH Register (x = 0, 1, 2)

15	0
PRDH	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 35. TIMERxPRDH Register Field Descriptions

Bits	Field	Description
15-0	PRDH	See description for TIMERxPRD

Figure 37. TIMERxTCR Register (x = 0, 1, 2)

15	14	13	12	11	10	9	8
TIF	TIE	Reserved		FREE	SOFT	Reserved	
R/W-0	R/W-0	R-0		R/W-0	R/W-0	R-0	

7	6	5	4	3	0
Reserved		TRB	TSS	Reserved	
R-0		R/W-0	R/W-0	R-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 36. TIMERxTCR Register Field Descriptions

Bits	Field	Value	Description
15	TIF		CPU-Timer Interrupt Flag.
		0	The CPU-Timer has not decremented to zero. Writes of 0 are ignored.
		1	This flag gets set when the CPU-timer decrements to zero. Writing a 1 to this bit clears the flag.
14	TIE		CPU-Timer Interrupt Enable.
		0	The CPU-Timer interrupt is disabled.
		1	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		Reserved
11-10	FREE SOFT		CPU-Timer Emulation Modes: These bits are special emulation bits that determine the state of the timer when a breakpoint is encountered in the high-level language debugger. If the FREE bit is set to 1, then, upon a software breakpoint, the timer continues to run (that is, free runs). In this case, SOFT is a *don't care*. But if FREE is 0, then SOFT takes effect. In this case, if SOFT = 0, the timer halts the next time the TIMH:TIM decrements. If the SOFT bit is 1, then the timer halts when the TIMH:TIM has decremented to zero.
		FREE SOFT	CPU-Timer Emulation Mode
		0 0	Stop after the next decrement of the TIMH:TIM (hard stop)
		0 1	Stop after the TIMH:TIM decrements to 0 (soft stop)
		1 0	Free run
		1 1	Free run In the SOFT STOP mode, the timer generates an interrupt before shutting down (since reaching 0 is the interrupt causing condition).
9-6	Reserved		Reserved
5	TRB		CPU-Timer Reload bit.
		0	The TRB bit is always read as zero. Writes of 0 are ignored.
		1	When you write a 1 to TRB, the TIMH:TIM is loaded with the value in the PRDH:PRD, and the prescaler counter (PSCH:PSC) is loaded with the value in the timer divide-down register (TDDRH:TDDR).
4	TSS		CPU-Timer stop status bit. TSS is a 1-bit flag that stops or starts the CPU-timer.
		0	Reads of 0 indicate the CPU-timer is running. To start or restart the CPU-timer, set TSS to 0. At reset, TSS is cleared to 0 and the CPU-timer immediately starts.
		1	Reads of 1 indicate that the CPU-timer is stopped. To stop the CPU-timer, set TSS to 1.
3-0	Reserved		Reserved

Figure 38. TIMERxTPR Register (x = 0, 1, 2)

15	8	7	0
PSC		TDDR	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 37. TIMERxTPR Register Field Descriptions

Bits	Field	Description
15-8	PSC	CPU-Timer Prescale Counter. These bits hold the current prescale count for the timer. For every timer clock source cycle that the PSCH:PSC value is greater than 0, the PSCH:PSC decrements by one. One timer clock (output of the timer prescaler) cycle after the PSCH:PSC reaches 0, the PSCH:PSC is loaded with the contents of the TDDRH:TDDR, and the timer counter register (TIMH:TIM) decrements by one. The PSCH:PSC is also reloaded whenever the timer reload bit (TRB) is set by software. The PSCH:PSC can be checked by reading the register, but it cannot be set directly. It must get its value from the timer divide-down register (TDDRH:TDDR). At reset, the PSCH:PSC is set to 0.
7-0	TDDR	CPU-Timer Divide-Down. Every (TDDRH:TDDR + 1) timer clock source cycles, the timer counter register (TIMH:TIM) decrements by one. At reset, the TDDRH:TDDR bits are cleared to 0. To increase the overall timer count by an integer factor, write this factor minus one to the TDDRH:TDDR bits. When the prescaler counter (PSCH:PSC) value is 0, one timer clock source cycle later, the contents of the TDDRH:TDDR reload the PSCH:PSC, and the TIMH:TIM decrements by one. TDDRH:TDDR also reloads the PSCH:PSC whenever the timer reload bit (TRB) is set by software.

Figure 39. TIMERxTPRH Register (x = 0, 1, 2)

15	8	7	0
PSCH		TDDRH	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 38. TIMERxTPRH Register Field Descriptions

Bits	Field	Description
15-8	PSCH	See description of TIMERxTPR.
7-0	TDDRH	See description of TIMERxTPR.

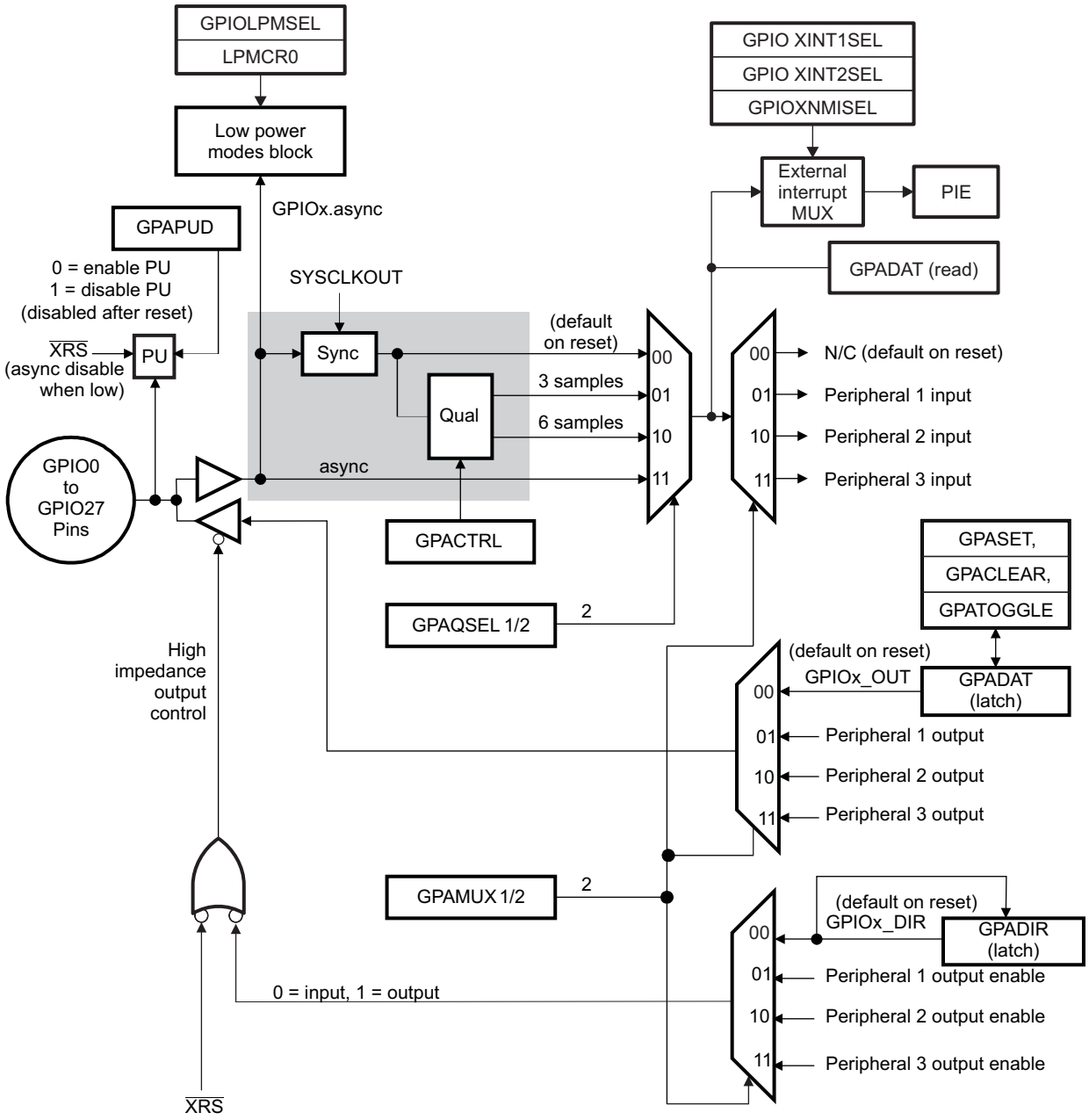
6 General-Purpose Input/Output (GPIO)

The GPIO multiplexing (MUX) registers are used to select the operation of shared pins. The pins are named by their general purpose I/O name (i.e., GPIO0 - GPIO87). These pins can be individually selected to operate as digital I/O, referred to as GPIO, or connected to one of up to three peripheral I/O signals (via the GPxMUXn registers). If selected for digital I/O mode, registers are provided to configure the pin direction (via the GPxDIR registers). You can also qualify the input signals to remove unwanted noise (via the GPxQSELn, GPACTRL, and GPBCTRL registers).

6.1 GPIO Module Overview

Up to three independent peripheral signals are multiplexed on a single GPIO-enabled pin in addition to individual pin bit-I/O capability. There are three 32-bit I/O ports. Port A consists of GPIO0-GPIO31, port B consists of GPIO32-GPIO63, and port C consists of GPIO64-87. Figure 40 shows the basic modes of operation for the GPIO module.

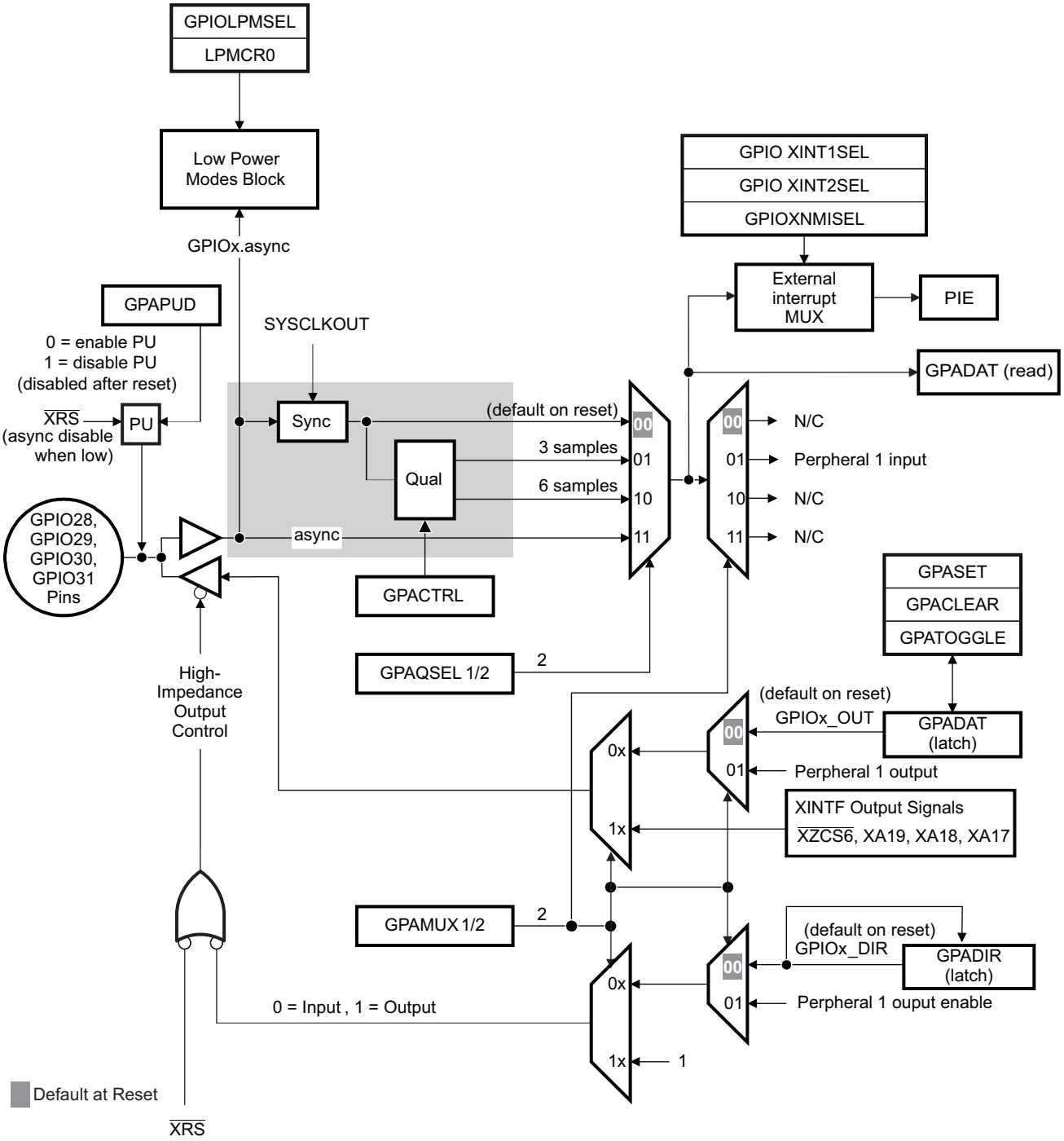
Figure 40. GPIO0 to GPIO27 Multiplexing Diagram



A The shaded area is disabled in the above GPIOs when the GPIOINENCLK bit is cleared to 0 in the PCLKCR3 register and the respective pin is configured as an output. This is to reduce power consumption when a pin is configured as an output. Clearing the GPIOINCLK bit will reset the sync and qualification logic so no residual value is left.

B GPxDAT latch/read are accessed at the same memory location.

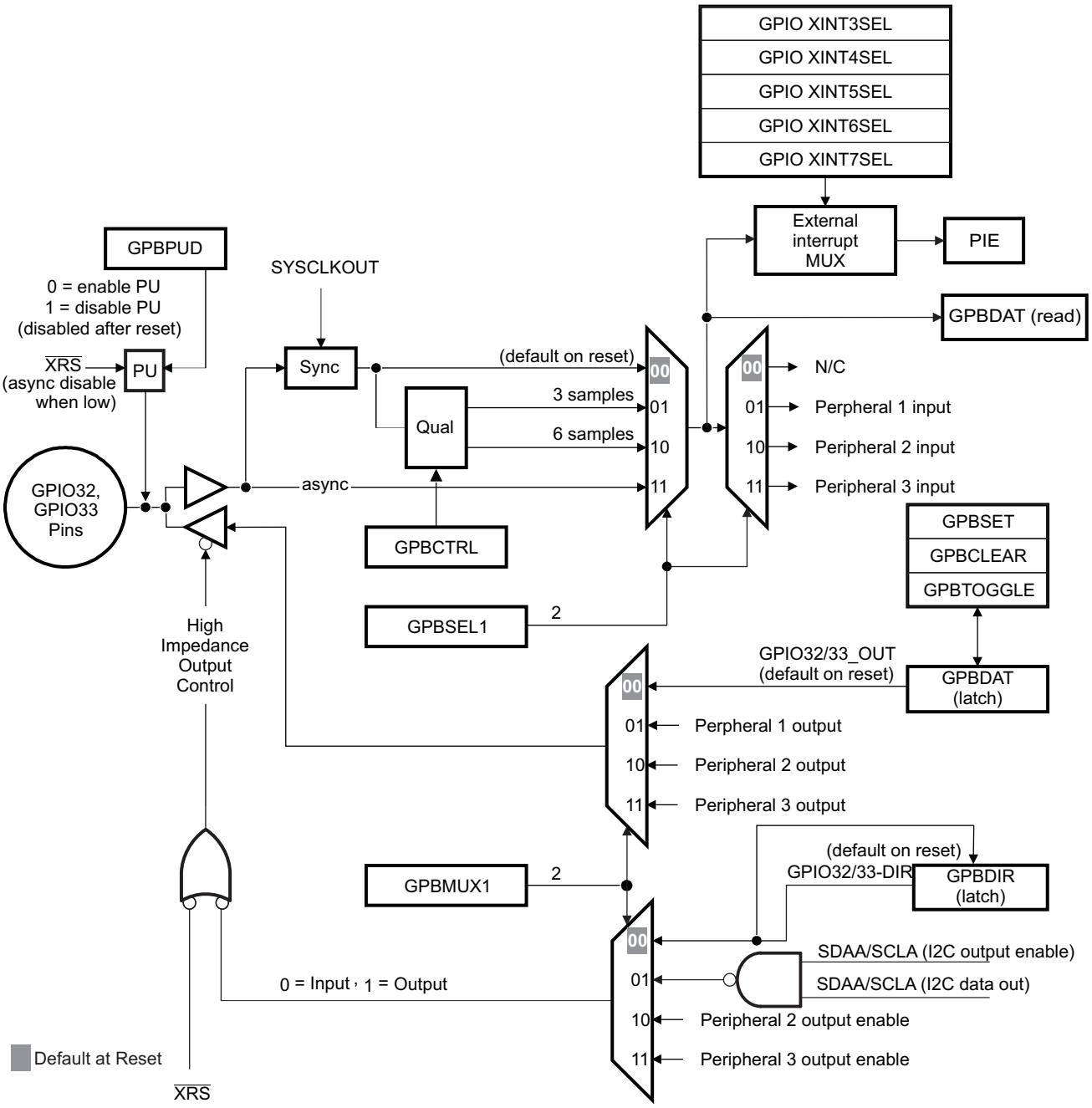
Figure 41. GPIO28 to GPIO31 Multiplexing Diagram (Peripheral 2 and Peripheral 3 Outputs Merged)



A The shaded area is disabled in the above GPIOs when the GPIOINENCLK bit is cleared to 0 in the PCLKCR3 register and the respective pin is configured as an output. This is to reduce power consumption when a pin is configured as an output. Clearing the GPIOINCLK bit will reset the sync and qualification logic so no residual value is left.

B The input qualification circuit is not reset when modes are changed (such as changing from output to input mode). Any state will get flushed by the circuit eventually.

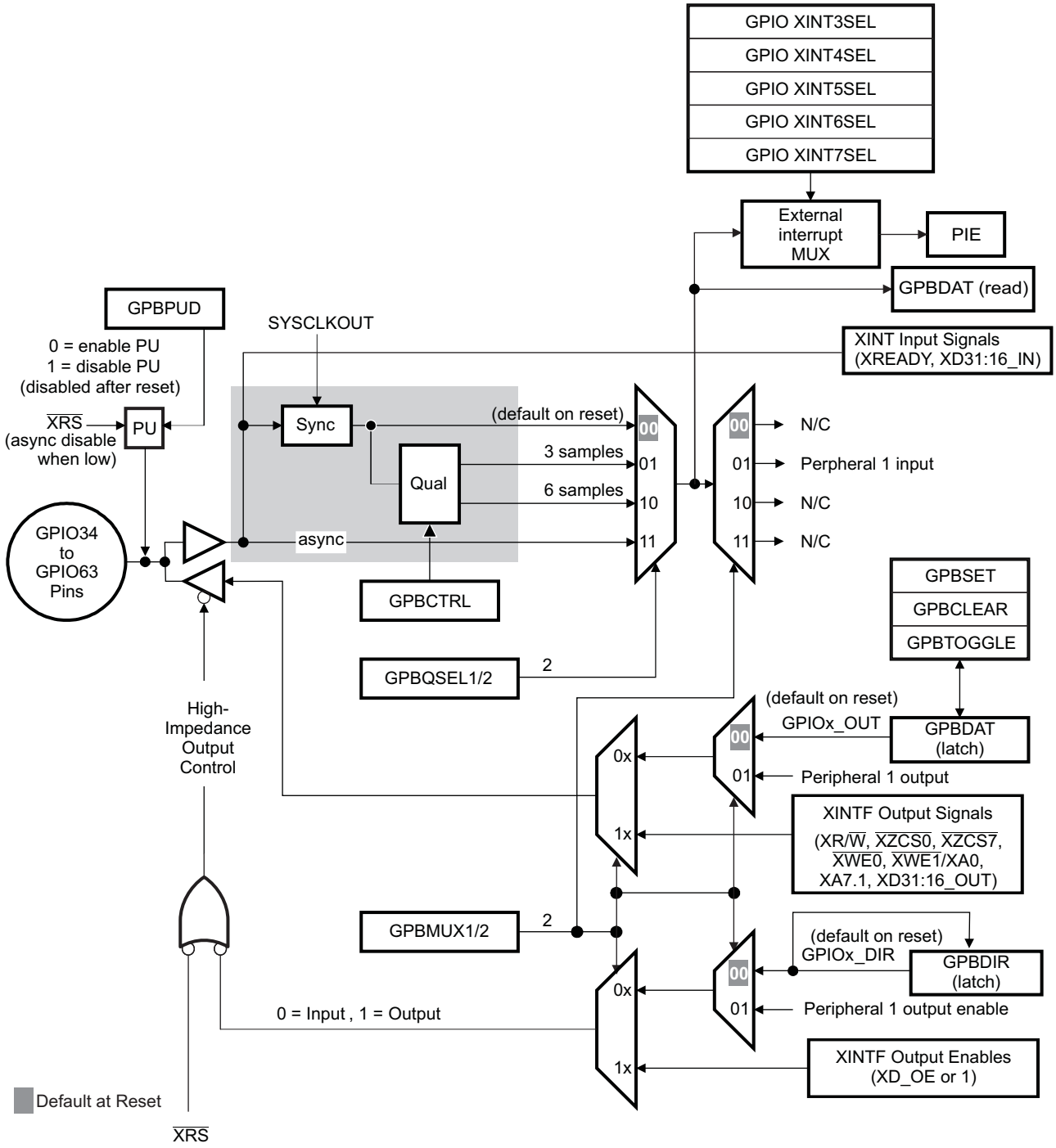
Figure 42. GPIO32, GPIO33 Multiplexing Diagram



A The GPIOINENCLK bit in the PCLKCR3 register does not affect the above GPIOs (I2C pins) since the pins are bi-directional.

B The input qualification circuit is not reset when modes are changed (such as changing from output to input mode). Any state will get flushed by the circuit eventually.

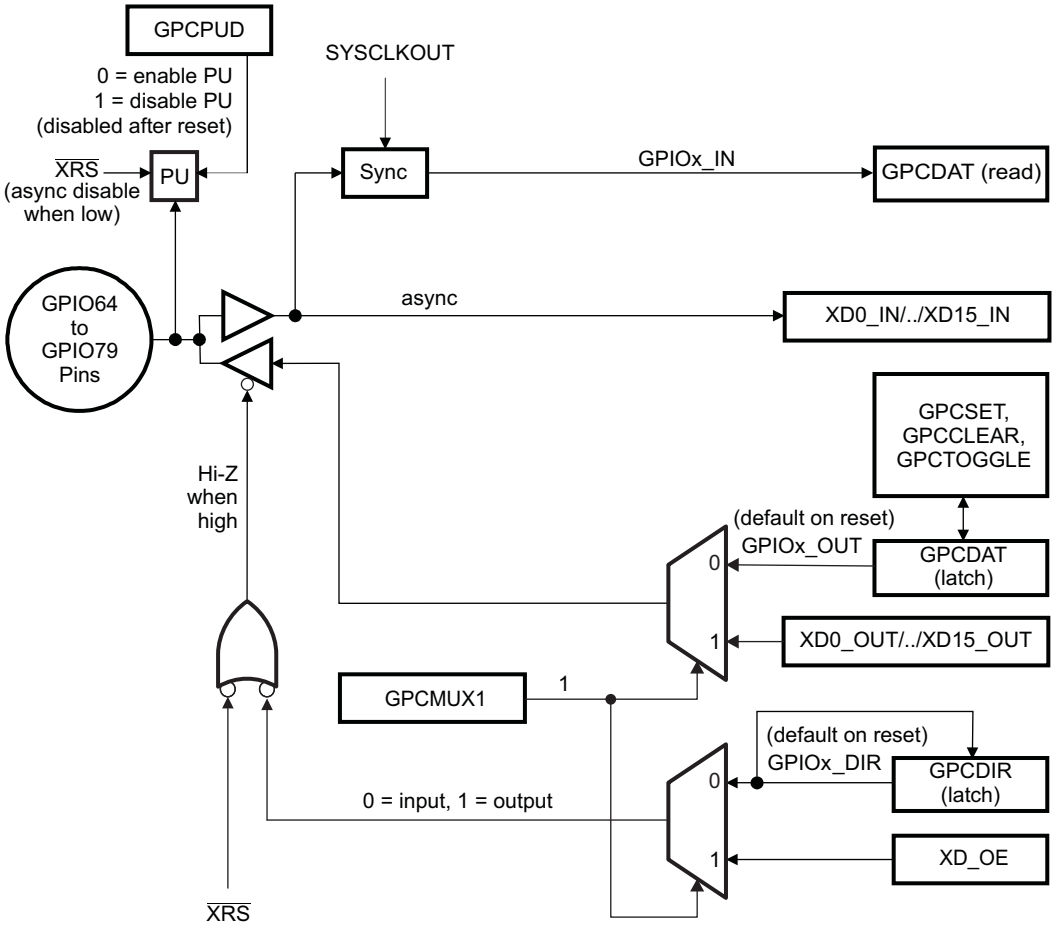
Figure 43. GPIO34 to GPIO63 Multiplexing Diagram (Peripheral 2 and Peripheral 3 Outputs Merged)



A The shaded area is disabled in the above GPIOs when the GPIOINENCLK bit is cleared to "0" in the PCLKCR3 register and the respective pin is configured as an output. This is to reduce power consumption when a pin is configured as an output. Clearing the GPIOINCLK bit will reset the sync and qualification logic so no residual value is left.

B The input qualification circuit is not reset when modes are changed (such as changing from output to input mode). Any state will get flushed by the circuit eventually.

Figure 44. GPIO64 to GPIO79 Multiplexing Diagram (Minimal GPIOs Without Qualification)



6.2 Configuration Overview

The pin function assignments, input qualification, and the external interrupt (XINT1 – XINT7, XNMI) sources are all controlled by the GPIO configuration control registers. In addition, you can assign pins to wake the device from the HALT and STANDBY low power modes and enable/disable internal pullup resistors. Table 39 and Table 40 list the registers that are used to configure the GPIO pins to match the system requirements.

Table 39. GPIO Control Registers

Name [1]	Address	Size (x16)	Register Description	Bit Description
GPACTRL	0x6F80	2	GPIO A Control Register (GPIO0-GPIO31)	Figure 53
GPAQSEL1	0x6F82	2	GPIO A Qualifier Select 1 Register (GPIO0-GPIO15)	Figure 55
GPAQSEL2	0x6F84	2	GPIO A Qualifier Select 2 Register (GPIO16-GPIO31)	Figure 56
GPAMUX1	0x6F86	2	GPIO A MUX 1 Register (GPIO0-GPIO15)	Figure 47
GPAMUX2	0x6F88	2	GPIO A MUX 2 Register (GPIO16-GPIO31)	Figure 48
GPADIR	0x6F8A	2	GPIO A Direction Register (GPIO0-GPIO31)	Figure 59
GPAPUD	0x6F8C	2	GPIO A Pull Up Disable Register (GPIO0-GPIO31)	Figure 62
GPBCTRL	0x6F90	2	GPIO B Control Register (GPIO32-GPIO63)	Figure 54
GPBQSEL1	0x6F92	2	GPIO B Qualifier Select 1 Register (GPIO32-GPIO47)	Figure 57
GPBQSEL2	0x6F94	2	GPIO B Qualifier Select 2 Register (GPIO48 - GPIO63)	Figure 58
GPBMUX1	0x6F96	2	GPIO B MUX 1 Register (GPIO32-GPIO47)	Figure 49
GPBMUX2	0x6F98	2	GPIO B MUX 2 Register (GPIO48-GPIO63)	Figure 50
GPBDIR	0x6F9A	2	GPIO B Direction Register (GPIO32-GPIO63)	Figure 60
GPBPUD	0x6F9C	2	GPIO B Pull Up Disable Register (GPIO32-GPIO63)	Figure 63
GPCMUX1	0x6FA6	2	GPIO C MUX 1 Register (GPIO64-GPIO79)	Figure 51
GPCMUX2	0x6FA8	2	GPIO C MUX 2 Register (GPIO80-GPIO87)	Figure 52
GPCDIR	0x6FAA	2	GPIO C Direction Register (GPIO64-GPIO87)	Figure 61
GPCPUD	0x6FAC	2	GPIO C Pull Up Disable Register (GPIO64-GPIO87)	Figure 64

(1) The registers in this table are EALLOW protected. See Section 7.2 for more information.

Table 40. GPIO Interrupt and Low Power Mode Select Registers

Name [1]	Address	Size (x16)	Register Description	Bit Description
GPIOXINT1SEL	0x6FE0	1	XINT1 Source Select Register (GPIO0-GPIO31)	Figure 71
GPIOXINT2SEL	0x6FE1	1	XINT2 Source Select Register (GPIO0-GPIO31)	Figure 71
GPIOXNMISEL	0x6FE2	1	XNMI Source Select Register (GPIO0-GPIO31)	Figure 71
GPIOXINT3SEL	0x6FE3	1	XINT3 Source Select Register (GPIO32 - GPIO63)	Table 82
GPIOXINT4SEL	0x6FE4	1	XINT4 Source Select Register (GPIO32 - GPIO63)	Table 82
GPIOXINT5SEL	0x6FE5	1	XINT5 Source Select Register (GPIO32 - GPIO63)	Table 82
GPIOXINT6SEL	0x6FE6	1	XINT6 Source Select Register (GPIO32 - GPIO63)	Table 82
GPIOXINT7SEL	0x6FE7	1	XINT7 Source Select Register (GPIO32 - GPIO63)	Table 82
GPIOLPMSEL	0x6FE8	1	LPM wakeup Source Select Register (GPIO0-GPIO31)	Figure 72

(1) The registers in this table are EALLOW protected. See Section 7.2 for more information.

To plan configuration of the GPIO module, consider the following steps:

Step 1. **Plan the device pin-out:**

Through a pin multiplexing scheme, a lot of flexibility is provided for assigning functionality to the GPIO-capable pins. Before getting started, look at the peripheral options available for each pin, and plan pin-out for your specific system. Will the pin be used as a general purpose input or output (GPIO) or as one of up to three available peripheral functions? Knowing this information will help determine how to further configure the pin.

Step 2. **Enable or disable internal pullup resistors:**

To enable or disable the internal pullup resistors, write to the respective bits in the GPIO pullup disable (GPAPUD, GPBPUD, and GPCPUD) registers. For pins that can function as ePWM output pins (GPIO0-GPIO11), the internal pullup resistors are disabled by default. All other GPIO-capable pins have the pullup enabled by default.

Step 3. **Select input qualification:**

If the pin will be used as an input, specify the required input qualification, if any. The input qualification is specified in the GPACTRL, GPBCTRL, GPAQSEL1, GPAQSEL2, GPBQSEL1, and GPBQSEL2 registers. By default, all of the input signals are synchronized to SYSCLKOUT only.

Step 4. **Select the pin function:**

Configure the GPxMUXn registers such that the pin is a GPIO or one of three available peripheral functions. By default, all GPIO-capable pins are configured at reset as general purpose input pins.

Step 5. **For digital general purpose I/O, select the direction of the pin:**

If the pin is configured as an GPIO, specify the direction of the pin as either input or output in the GPADIR, GPBDIR, and GPCDIR registers. By default, all GPIO pins are inputs. To change the pin from input to output, first load the output latch with the value to be driven by writing the appropriate value to the GPxCLEAR, GPxSET, or GPxTOGGLE registers. Once the output latch is loaded, change the pin direction from input to output via the GPxDIR registers. The output latch for all pins is cleared at reset.

Step 6. **Select low power mode wake-up sources:**

Specify which pins, if any, will be able to wake the device from HALT and STANDBY low power modes. The pins are specified in the GPIOLPMSEL register.

Step 7. **Select external interrupt sources:**

Specify the source for the XINT1 - XINT7, and XNMI interrupts. For each interrupt you can specify one of the port A signals (for XINT1/2/3) or port B signals (XINT4/5/6/7) as the source. This is done by specifying the source in the GPIOXINTnSEL, and GPIOXNMISEL registers. The polarity of the interrupts can be configured in the XINTnCR, and the XNMICR registers as described in Section 8.6.

NOTE: There is a 2-SYSCLKOUT cycle delay from when a write to configuration registers such as GPxMUXn and GPxQSELn occurs to when the action is valid

6.3 Digital General Purpose I/O Control

For pins that are configured as GPIO you can change the values on the pins by using the registers in Table 41.

Table 41. GPIO Data Registers

Name	Address	Size (x16)	Register Description	Bit Description
GPADAT	0x6FC0	2	GPIO A Data Register (GPIO0-GPIO31)	Figure 65
GPASET	0x6FC2	2	GPIO A Set Register (GPIO0-GPIO31)	Figure 68
GPACLEAR	0x6FC4	2	GPIO A Clear Register (GPIO0-GPIO31)	Figure 68
GPATOGGLE	0x6FC6	2	GPIO A Toggle Register (GPIO0-GPIO31)	Figure 68
GPBDAT	0x6FC8	2	GPIO B Data Register (GPIO32-GPIO63)	Figure 66
GPBSET	0x6FCA	2	GPIO B Set Register (GPIO32-GPIO63)	Figure 69
GPBCLEAR	0x6FCC	2	GPIO B Clear Register (GPIO32-GPIO63)	Figure 69
GPBTOGGLE	0x6FCE	2	GPIO B Toggle Register (GPIO32-GPIO63)	Figure 69

Table 41. GPIO Data Registers (continued)

Name	Address	Size (x16)	Register Description	Bit Description
GPCDAT	0x6FD0	2	GPIO C Data Register (GPIO64 - GPIO87)	Figure 67
GPCSET	0x6FD2	2	GPIO C Set Register (GPIO64 - GPIO87)	Figure 70
GPCCLEAR	0x6FD4	2	GPIO C Clear Register (GPIO64 - GPIO87)	Figure 70
GPCTOGGLE	0x6FD6	2	GPIO C Toggle Register (GPIO64 - GPIO87)	Figure 70

- **GPxDAT Registers**

  Each I/O port has one data register. Each bit in the data register corresponds to one GPIO pin. No matter how the pin is configured (GPIO or peripheral function), the corresponding bit in the data register reflects the current state of the pin after qualification. Writing to the GPxDAT register clears or sets the corresponding output latch and if the pin is enabled as a general purpose output (GPIO output) the pin will also be driven either low or high. If the pin is not configured as a GPIO output then the value will be latched, but the pin will not be driven. Only if the pin is later configured as a GPIO output, will the latched value be driven onto the pin.

  When using the GPxDAT register to change the level of an output pin, you should be cautious not to accidentally change the level of another pin. For example, if you mean to change the output latch level of GPIOA0 by writing to the GPADAT register bit 0, using a read-modify-write instruction. The problem can occur if another I/O port A signal changes level between the read and the write stage of the instruction. You can also change the state of that output latch. You can avoid this scenario by using the GPxSET, GPxCLEAR, and GPxTOGGLE registers to load the output latch instead.

- **GPxSET Registers**

  The set registers are used to drive specified GPIO pins high without disturbing other pins. Each I/O port has one set register and each bit corresponds to one GPIO pin. The set registers always read back 0. If the corresponding pin is configured as an output, then writing a 1 to that bit in the set register will set the output latch high and the corresponding pin will be driven high. If the pin is not configured as a GPIO output, then the value will be latched but the pin will not be driven. Only if the pin is later configured as a GPIO output will the latched value will be driven onto the pin. Writing a 0 to any bit in the set registers has no effect.

- **GPxCLEAR Registers**

  The clear registers are used to drive specified GPIO pins low without disturbing other pins. Each I/O port has one clear register. The clear registers always read back 0. If the corresponding pin is configured as a general purpose output, then writing a 1 to the corresponding bit in the clear register will clear the output latch and the pin will be driven low. If the pin is not configured as a GPIO output, then the value will be latched but the pin will not be driven. Only if the pin is later configured as a GPIO output will the latched value will be driven onto the pin. Writing a 0 to any bit in the clear registers has no effect.

- **GPxTOGGLE Registers**

  The toggle registers are used to drive specified GPIO pins to the opposite level without disturbing other pins. Each I/O port has one toggle register. The toggle registers always read back 0. If the corresponding pin is configured as an output, then writing a 1 to that bit in the toggle register flips the output latch and pulls the corresponding pin in the opposite direction. That is, if the output pin is driven low, then writing a 1 to the corresponding bit in the toggle register will pull the pin high. Likewise, if the output pin is high, then writing a 1 to the corresponding bit in the toggle register will pull the pin low. If the pin is not configured as a GPIO output, then the value will be latched but the pin will not be driven. Only if the pin is later configured as a GPIO output will the latched value will be driven onto the pin. Writing a 0 to any bit in the toggle registers has no effect.

6.4 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 GPAQSEL1, GPAQSEL2, GPBQSEL1 and GPBQSEL2 registers. In the case of a GPIO input pin, the qualification can be specified as only synchronize to SYSCLKOUT or qualification by a sampling window. For pins that are configured as peripheral inputs, the input can also be asynchronous in addition to synchronized to SYSCLKOUT or qualified by a sampling window. The remainder of this section describes the options available.

6.4.1 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, eCAN, and I2C. In addition, it may be desirable to have the ePWM trip zone ($\overline{TZ1}$-$\overline{TZ6}$) signals function independent of the presence of SYSCLKOUT.

The asynchronous option is not valid if the pin is used as a general purpose digital input pin (GPIO). If the pin is configured as a GPIO input and the asynchronous option is selected then the qualification defaults to synchronization to SYSCLKOUT as described in Section 6.4.2.

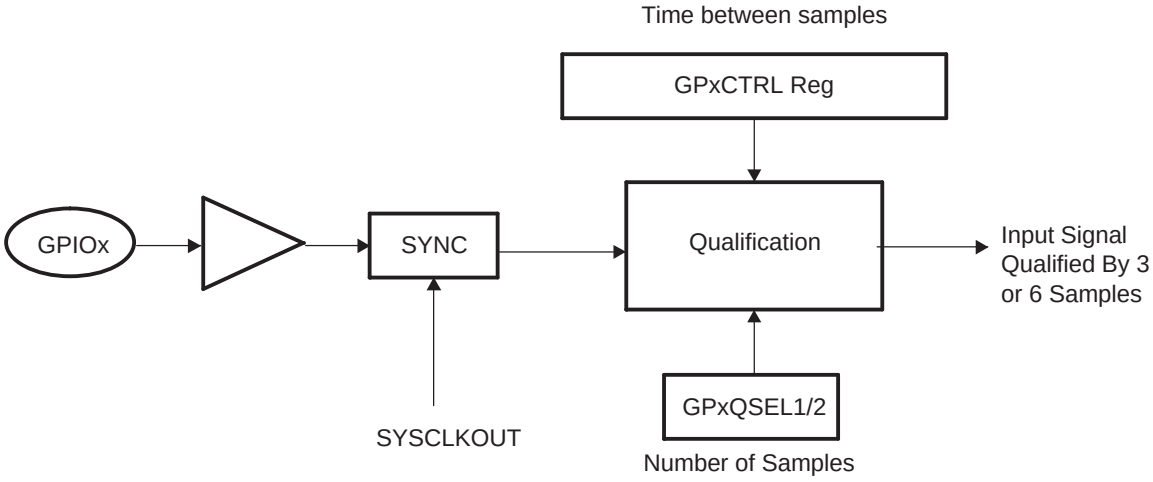
6.4.2 Synchronization to SYSCLKOUT Only

This is the default qualification mode of all the pins at reset. In this mode, the input signal is only synchronized to the system clock (SYSCLKOUT). Because the incoming signal is asynchronous, it can take up to a SYSCLKOUT period of delay in order for the input to the DSP to be changed. No further qualification is performed on the signal.

6.4.3 Qualification Using a Sampling Window

In this mode, the signal is first synchronized to the system clock (SYSCLKOUT) and then qualified by a specified number of cycles before the input is allowed to change. Figure 45 and Figure 46 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 45. Input Qualification Using a Sampling Window



Time between samples (sampling period):

To qualify the signal, the input signal is sampled at a regular period. The sampling period is specified by the user and determines the time duration between samples, or how often the signal will be sampled, relative to the CPU clock (SYSCLKOUT).

The sampling period is specified by the qualification period (QUALPRDn) bits in the GPxCTRL register. The sampling period is configurable in groups of 8 input signals. For example, GPIO0 to GPIO7 use GPACTRL[QUALPRD0] setting and GPIO8 to GPIO15 use GPACTRL[QUALPRD1]. Table 42 and Table 43 show the relationship between the sampling period or sampling frequency and the GPxCTRL[QUALPRDn] setting.

Table 42. Sampling Period

	Sampling Period
If GPxCTRL[QUALPRDn] = 0	$1 \times T_{SYSCLKOUT}$
If GPxCTRL[QUALPRDn] ≠ 0	$2 \times$ GPxCTRL[QUALPRDn] $\times T_{SYSCLKOUT}$
	Where $T_{SYSCLKOUT}$ is the period in time of SYSCLKOUT

Table 43. Sampling Frequency

	Sampling Frequency
If GPxCTRL[QUALPRDn] = 0	$f_{SYSCLKOUT}$
If GPxCTRL[QUALPRDn] ≠ 0	$f_{SYSCLKOUT} \times 1 \div (2 \times$ GPxCTRL[QUALPRDn])
	Where $f_{SYSCLKOUT}$ is the frequency of SYSCLKOUT

From these equations, the minimum and maximum time between samples can be calculated for a given SYSCLKOUT frequency:

Example: Maximum Sampling Frequency:

If GPxCTRL[QUALPRDn] = 0
then the sampling frequency is $f_{SYSCLKOUT}$
If, for example, $f_{SYSCLKOUT}$ = 150 MHz
then the signal will be sampled at 150 MHz or one sample every 6.67 ns.

Example: Minimum Sampling Frequency:

If GPxCTRL[QUALPRDn] = 0xFF (i.e. 255)
then the sampling frequency is $f_{SYSCLKOUT} \times 1 \div (2 \times$ GPxCTRL[QUALPRDn])
If, for example, $f_{SYSCLKOUT}$ = 150 MHz
then the signal will be sampled at 150 MHz $\times 1 \div (2 \times 255)$ or one sample every 3.4 μs.

Number of samples:

The number of times the signal is sampled is either 3 samples or 6 samples as specified in the qualification selection (GPAQSEL1, GPAQSEL2, GPBQSEL1, and GPBQSEL2) registers. When 3 or 6 consecutive cycles are the same, then the input change will be passed through to the DSP.

Total Sampling Window Width:

The sampling window is the time during which the input signal will be sampled as shown in Figure 46. 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 then the number of samples taken. For a thee-sample window, the sampling window width is 2 sampling periods wide where the sampling period is defined in Table 42. Likewise, for a six-sample window, the sampling window width is 5 sampling periods wide. Table 44 and Table 45 show the calculations that can be used to determine the total sampling window width based on GPxCTRL[QUALPRDn] and the number of samples taken.

Table 44. Case 1: Three-Sample Sampling Window Width

	Total Sampling Window Width
If GPxCTRL[QUALPRDn] = 0	$2 \times T_{SYSCLKOUT}$
If GPxCTRL[QUALPRDn] ≠ 0	$2 \times 2 \times$ GPxCTRL[QUALPRDn] $\times T_{SYSCLKOUT}$
	Where $T_{SYSCLKOUT}$ is the period in time of SYSCLKOUT

Table 45. Case 2: Six-Sample Sampling Window Width

	Total Sampling Window Width
If GPxCTRL[QUALPRDn] = 0	$5 \times T_{SYSCLKOUT}$
If GPxCTRL[QUALPRDn] ≠ 0	$5 \times 2 \times$ GPxCTRL[QUALPRDn] $\times T_{SYSCLKOUT}$
	Where $T_{SYSCLKOUT}$ is the period in time of SYSCLKOUT

NOTE: The external signal change is asynchronous with respect to both the sampling period and SYSCLKOUT. Due to the asynchronous nature of the external signal, the input should be held stable for a time greater than the sampling window width to make sure the logic detects a change in the signal. The extra time required can be up to an additional sampling period + $T_{SYSCLKOUT}$.

The required duration for an input signal to be stable for the qualification logic to detect a change is described in the device specific data manual.

Example Qualification Window:

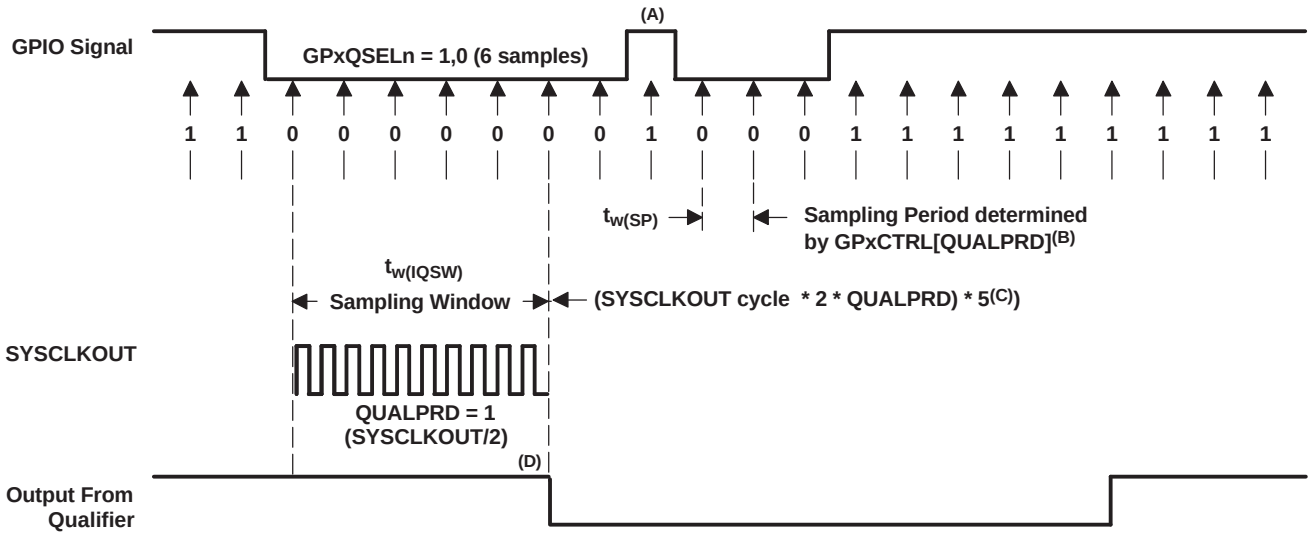
For the example shown in Figure 46, the input qualification has been configured as follows:

- GPxQSEL1/2 = 1,0. This indicates a six-sample qualification.
- GPxCTRL[QUALPRDn] = 1. The sampling period is $t_w(SP) = 2 \times GPxCTRL[QUALPRDn] \times T_{SYSCLKOUT}$.

This configuration results in the following:

- The width of the sampling window is: .
  $t_w(IQSW) = 5 \times t_w(SP) = 5 \times 2 \times GPxCTRL[QUALPRDn] \times T_{SYSCLKOUT}$ or $5 \times 2 \times T_{SYSCLKOUT}$
- If, for example, $T_{SYSCLKOUT}$ = 6.67 ns, then the duration of the sampling window is:
  $t_w(IQSW) = 5 \times 2 \times 6.67$ ns = 67 ns.
- To account for the asynchronous nature of the input relative to the sampling period and SYSCLKOUT, up to an additional sampling period, $t_w(SP)$, + $T_{SYSCLKOUT}$ may be required to detect a change in the input signal. For this example:
  $t_w(SP) + T_{SYSCLKOUT}$ = 13.34 ns + 6.67 ns = 20 ns
- In Figure 46, the glitch (A) is shorter then the qualification window and will be ignored by the input qualifier.

Figure 46. 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.

6.5 GPIO and Peripheral Multiplexing (MUX)

Up to three different peripheral functions are multiplexed along with a general input/output (GPIO) function per pin. This allows you to pick and choose a peripheral mix that will work best for the particular application.

Table 47, Table 48, and Table 49 show an overview of the possible multiplexing combinations sorted by GPIO pin. The second column indicates the I/O name of the pin on the device. Since the I/O name is unique, it is the best way to identify a particular pin. Therefore, the register descriptions in this section only refer to the GPIO name of a particular pin. The MUX register and particular bits that control the selection for each pin are indicated in the first column.

For example, the multiplexing for the GPIO7 pin is controlled by writing to GPAMUX[15:14]. By writing to these bits, the pin is configured as either GPIO7, or one of up to three peripheral functions. The GPIO7 pin can be configured as follows:

GPAMUX1[15:14] Bit Setting	Pin Functionality Selected
If GPAMUX1[15:14] = 0,0	Pin configured as GPIO7
If GPAMUX1[15:14] = 0,1	Pin configured as EPWM4B (O)
If GPAMUX1[15:14] = 1,0	Pin configured as MCLKRA (I/O)
If GPAMUX1[15:14] = 1,1	Pin configured as ECAP2 (I/O)

All devices in the 2833x, 2823x family have the same multiplexing scheme. The only difference is that if a peripheral is not available on a particular device, that MUX selection is reserved on that device and should not be used.

NOTE: If you should select a reserved GPIO MUX configuration that is not mapped to a peripheral, the state of the pin will be undefined and the pin may be driven. Reserved configurations are for future expansion and should not be selected. In the device MUX tables (Table 47, Table 48, and Table 49) these options are indicated as "Reserved".

Some peripherals can be assigned to more than one pin via the MUX registers. For example, the CAP1 function can be assigned to either the GPIO5 or GPIO24 pin, depending on individual system requirements as shown below:

Pin Assigned to CAP1		MUX Configuration
Choice 1	GPIO5	GPAMUX1[11:10] = 1,1
or Choice 2	GPIO24	GPAMUX2[17:16] = 0,1

If no pin is configured as an input to a peripheral, or if more than one pin is configured as an input for the same peripheral, then the input to the peripheral will either default to a 0 or a 1 as shown in Table 46. For example, if ECAP1 were assigned to both GPIO5 and GPIO24, the input to the eCAP1 peripheral would default to a high state as shown in Table 46 and the input would not be connected to GPIO5 or GPIO24.

Table 46. Default State of Peripheral Input

Peripheral Input	Description	Default Input (1)
$\overline{TZ1}$-$\overline{TZ6}$	Trip zone 1-6	1
EPWMSYNCI	ePWM Synch Input	0
ECAPn	eCAP input	1
EQEPnA	eQEP input	1
EQEPnI	eQEP index	1
EQEPnS	eQEP strobe	1
SPICLKx	SPI clock	1
$\overline{SPISTEx}$	SPI transmit enable	0
SPISIMOx	SPI Slave-in, master-out	1
SPISOMIx	SPI Slave-out, master-in	1
SCIRXDx	SCI receive	1
CANRXx	CAN receive	1
SDAA	I2C data	1
SCLA1	I2C clock	1

(1) This value will be assigned to the peripheral input if more then one pin has been assigned to the peripheral function in the GPxMUX1/2 registers or if no pin has been assigned.

Table 47. GPIOA MUX

	Default at Reset Primary I/O Function	Peripheral Selection	Peripheral Selection 2	Peripheral Selection 3
GPAMUX1 Register Bits	**(GPAMUX1 bits = 00)**	**(GPAMUX1 bits = 01)**	**(GPAMUX1 bits = 10)**	**(GPAMUX1 bits = 11)**
1-0	GPIO0	EPWM1A (O)	Reserved(1)	Reserved(1)
3-2	GPIO1	EPWM1B (O)	ECAP6 (I/O)	MFSRB (I/O)(1)
5-4	GPIO2	EPWM2A (O)	Reserved(1)	Reserved(1)
7-6	GPIO3	EPWM2B (O)	ECAP5 (I/O)	MCLKRB (I/O)(1)
9-8	GPIO4	EPWM3A (O)	Reserved(1)	Reserved(1)
11-10	GPIO5	EPWM3B (O)	MFSRA (I/O)	ECAP1 (I/O)
13-12	GPIO6	EPWM4A (O)	EPWMSYNCI (I)	EPWMSYNCO (O)
15-14	GPIO7	EPWM4B (O)	MCLKRA (I/O)	ECAP2 (I/O)
17-16	GPIO8	EPWM5A (O)	CANTXB (O)	$\overline{\text{ADCSOCAO}}$ (O)
19-18	GPIO9	EPWM5B (O)	SCITXDB (O)	ECAP3 (I/O)
21-20	GPIO10	EPWM6A (O)	CANRXB (I)	$\overline{\text{ADCSOCBO}}$ (O)
23-22	GPIO11	EPWM6B (O)	SCIRXDB (I)	ECAP4 (I/O)
25-24	GPIO12	$\overline{\text{TZ1}}$ (I)	CANTXB (O)	MDXB (O)
27-26	GPIO13	$\overline{\text{TZ2}}$ (I)	CANRXB (I)	MDRB (I)
29-28	GPIO14	$\overline{\text{TZ3}}$/$\overline{\text{XHOLD}}$ (I)	SCITXDB (O)	MCLKXB (I/O)
31-30	GPIO15	$\overline{\text{TZ4}}$/$\overline{\text{XHOLDA}}$ (O)	SCIRXDB (I)	MFSXB (I/O)
GPAMUX2 Register Bits	**(GPAMUX2 bits = 00)**	**(GPAMUX2 bits = 01)**	**(GPAMUX2 bits = 10)**	**(GPAMUX2 bits = 11)**
1-0	GPIO16	SPISIMOA (I/O)	CANTXB (O)	$\overline{\text{TZ5}}$ (I)
3-2	GPIO17	SPISOMIA (I/O)	CANRXB (I)	$\overline{\text{TZ6}}$ (I)
5-4	GPIO18	SPICLKA (I/O)	SCITXDB (O)	CANRXA (I)
7-6	GPIO19	$\overline{\text{SPISTEA}}$ (I/O)	SCIRXDB (I)	CANTXA (O)
9-8	GPIO20	EQEP1A (I)	MDXA (O)	CANTXB (O)
11-10	GPIO21	EQEP1B (I)	MDRA (I)	CANRXB (I)
13-12	GPIO22	EQEP1S (I/O)	MCLKXA (I/O)	SCITXDB (O)
15-14	GPIO23	EQEP1I (I/O)	MFSXA (I/O)	SCIRXDB (I)
17-16	GPIO24	ECAP1 (I/O)	EQEP2A (I)	MDXB (O)
19-18	GPIO25	ECAP2 (I/O)	EQEP2B (I)	MDRB (I)
21-20	GPIO26	ECAP3 (I/O)	EQEP2I (I/O)	MCLKXB (I/O)
23-22	GPIO27	ECAP4 (I/O)	EQEP2S (I/O)	MFSXB (I/O)
25-24	GPIO28	SCIRXDA (I)	$\overline{\text{XZCS6}}$ (O)	$\overline{\text{XZCS6}}$ (O)
27-26	GPIO29	SCITXDA (O)	XA19 (O)	XA19 (O)
29-28	GPIO30	CANRXA (I)	XA18 (O)	XA18 (O)
31-30	GPIO31	CANTXA (O)	XA17 (O)	XA17 (O)

(1) The word "Reserved" means that there is no peripheral assigned to this GPxMUX1/2 register setting. Should it be selected, the state of the pin will be undefined and the pin may be driven. This selection is a reserved configuration for future expansion.

Table 48. GPIOB MUX

	Default at Reset Primary I/O Function	Peripheral Selection 1	Peripheral Selection 2	Peripheral Selection 3
GPBMUX1 Register Bits	**(GPBMUX1 bits = 00)**	**(GPBMUX1 bits = 01)**	**(GPBMUX1 bits = 10)**	**(GPBMUX1 bits = 11)**
1,0	GPIO32 (I/O)	SDAA (I/OC)	EPWMSYNCI (I)	$\overline{\text{ADCSOCAO}}$ (O)
3,2	GPIO33 (I/O)	SCLA (I/OC)	EPWMSYNCO (O)	$\overline{\text{ADCSOCBO}}$ (O)
5,4	GPIO34 (I/O)	ECAP1 (I/O)	XREADY (I)	XREADY (I)
7,6	GPIO35 (I/O)	SCITXDA (O)	XR/$\overline{\text{W}}$ (O)	XR/$\overline{\text{W}}$ (O)
9,8	GPIO36 (I/O)	SCIRXDA (I)	$\overline{\text{XZCS0}}$ (O)	$\overline{\text{XZCS0}}$ (O)
11,10	GPIO37 (I/O)	ECAP2 (I/O)	$\overline{\text{XZCS7}}$ (O)	$\overline{\text{XZCS7}}$ (O)
13,12	GPIO38 (I/O)	Reserved	$\overline{\text{XWE0}}$ (O)	$\overline{\text{XWE0}}$ (O)
15,14	GPIO39 (I/O)	Reserved	XA16 (O)	XA16 (O)
17,16	GPIO40 (I/O)	Reserved	XA0/$\overline{\text{XWE1}}$ (O)	XA0/$\overline{\text{XWE1}}$ (O)
19,18	GPIO41 (I/O)	Reserved	XA1 (O)	XA1 (O)
21,20	GPIO42 (I/O)	Reserved	XA2 (O)	XA2 (O)
23,22	GPIO43 (I/O)	Reserved	XA3 (O)	XA3 (O)
25,24	GPIO44 (I/O)	Reserved	XA4 (O)	XA4 (O)
27,26	GPIO45 (I/O)	Reserved	XA5 (O)	XA5 (O)
29,28	GPIO46 (I/O)	Reserved	XA6 (O)	XA6 (O)
31,30	GPIO47 (I/O)	Reserved	XA7 (O)	XA7 (O)
GPBMUX2 Register Bits	**(GPBMUX2 bits = 00)**	**(GPBMUX2 bits = 01)**	**(GPBMUX2 bits = 10 or 11)**	
1,0	GPIO48 (I/O)	ECAP5 (I/O)	XD31 (I/O)	
3,2	GPIO49 (I/O)	ECAP6 (I/O)	XD30 (I/O)	
5,4	GPIO50 (I/O)	EQEP1A (I)	XD29 (I/O)	
7,6	GPIO51 (I/O)	EQEP1B (I)	XD28 (I/O)	
9,8	GPIO52 (I/O)	EQEP1S (I/O)	XD27 (I/O)	
11,10	GPIO53 (I/O)	EQEP1I (I/O)	XD26 (I/O)	
13,12	GPIO54 (I/O)	SPISIMOA (I/O)	XD25 (I/O)	
15,14	GPIO55 (I/O)	SPISOMIA (I/O)	XD24 (I/O)	
17,16	GPIO56 (I/O)	SPICLKA (I/O)	XD23 (I/O)	
19,18	GPIO57 (I/O)	$\overline{\text{SPISTEA}}$ (I/O)	XD22 (I/O)	
21,20	GPIO58 (I/O)	MCLKRA (I/O)	XD21 (I/O)	
23,22	GPIO59 (I/O)	MFSRA (I/O)	XD20 (I/O)	
25,24	GPIO60 (I/O)	MCLKRB (I/O)	XD19 (I/O)	
27,26	GPIO61 (I/O)	MFSRB (I/O)	XD18 (I/O)	
29,28	GPIO62 (I/O)	SCIRXDC (I)	XD17 (I/O)	
31,30	GPIO63 (I/O)	SCITXDC (O)	XD16 (I/O)	

Table 49. GPIOC MUX

	Default at Reset Primary I/O Function	Peripheral Selection 2 or 3
GPCMUX1 Register Bits	**(GPCMUX1 bits = 00 or 01)**	**(GPCMUX1 bits = 10 or 11)**
1,0	GPIO64 (I/O)	XD15 (I/O)
3,2	GPIO65 (I/O)	XD14 (I/O)
5,4	GPIO66 (I/O)	XD13 (I/O)
7,6	GPIO67 (I/O)	XD12 (I/O)
9,8	GPIO68 (I/O)	XD11 (I/O)
11,10	GPIO69 (I/O)	XD10 (I/O)
13,12	GPIO70 (I/O)	XD9 (I/O)
15,14	GPIO71 (I/O)	XD8 (I/O)
17,16	GPIO72 (I/O)	XD7 (I/O)
19,18	GPIO73 (I/O)	XD6 (I/O)
21,20	GPIO74 (I/O)	XD5 (I/O)
23,22	GPIO75 (I/O)	XD4 (I/O)
25,24	GPIO76 (I/O)	XD3 (I/O)
27,26	GPIO77 (I/O)	XD2 (I/O)
29,28	GPIO78 (I/O)	XD1 (I/O)
31,30	GPIO79 (I/O)	XD0 (I/O)
GPCMUX2 Register Bits	**GPCMUX2 bits = 00 or 01**	**GPCMUX2 bits = 10 or 11**
1,0	GPIO80 (I/O)	XA8 (O)
3,2	GPIO81 (I/O)	XA9 (O)
5,4	GPIO82 (I/O)	XA10 (O)
7,6	GPIO83 (I/O)	XA11 (O)
9,8	GPIO84 (I/O)	XA12 (O)
11,10	GPIO85 (I/O)	XA13 (O)
13,12	GPIO86 (I/O)	XA14 (O)
15,14	GPIO87 (I/O)	XA15 (O)
16 – 31	Reserved	Reserved

6.6 Register Bit Definitions

Figure 47. GPIO Port A MUX 1 (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-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
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-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND- R/W = Read/Write; R = Read only; -*n* = value after reset

Table 50. GPIO Port A Multiplexing 1 (GPAMUX1) Register Field Descriptions

Bits	Field	Value	Description [1]
31-30	GPIO15		Configure the GPIO15 pin as:
		00	GPIO15 - General purpose input/output 15 (default) (I/O)
		01	$\overline{TZ4}$ - Trip Zone 4 (I) or $\overline{XHOLDA}$ (O). The pin function for this option is based on the direction chosen in the GPADIR register. If the pin is configured as an input, then $\overline{TZ4}$ function is chosen. If the pin is configured as an output, then $\overline{XHOLDA}$ function is chosen. $\overline{XHOLDA}$ is driven active (low) when the XINTF has granted an $\overline{XHOLD}$ request. All XINTF buses and strobe signals will be in a high-impedance state. $\overline{XHOLDA}$ is released when the $\overline{XHOLD}$ signal is released. External devices should only drive the external bus when $\overline{XHOLDA}$ is active (low).
		10	SCIRXDB - SCI-B receive. (I)
		11	MFSXB - McBSP-B transmit frame synch (I/O) This option is reserved on devices that do not have a McBSP-B port. [2]
29-28	GPIO14		Configure the GPIO14 pin as:
		00	GPIO14 - General purpose I/O 14 (default) (I/O)
		01	$\overline{TZ3}$ - Trip zone 3 or $\overline{XHOLD}$ (I). $\overline{XHOLD}$, when active (low), requests the external memory interface (XINTF) to release the external bus and place all buses and strobes into a high-impedance state. To prevent this from happening when $\overline{TZ3}$ signal goes active, disable this function by writing XINTCNF2[HOLD] = 1. If this is not done, the XINTF bus will go into high impedance anytime $\overline{TZ3}$ goes low. On the ePWM side, $\overline{TZn}$ signals are ignored by default, unless they are enabled by the code. The XINTF will release the bus when any current access is complete and there are no pending accesses on the XINTF. (I)
		10	SCITXDB - SCI-B transmit (O)
		11	MCLKXB - McBSP-B transmit clock (I/O) This option is reserved on devices that do not have a McBSP-B port. [2]
27-26	GPIO13		Configure the GPIO13 pin as:
		00	GPIO13 - General purpose I/O 13 (default) (I/O)
		01	$\overline{TZ2}$ - Trip zone 2 (I)
		10	CANRXB - eCAN-B receive. (I)
		11	MDRB - McBSP-B Data Receive (I) This option is reserved on devices that do not have a McBSP-B port. [2]
25-24	GPIO12		Configure the GPIO12 pin as:
		00	GPIO12 - General purpose I/O 12 (default) (I/O)
		01	$\overline{TZ1}$ - Trip zone 1 (I)
		10	CANTXB - eCAN-B transmit. (O)
		11	MDXB - McBSP-B, Data transmit (O) This option is reserved on devices that do not have a McBSP-B port. [2]

[1] This register is EALLOW protected. See Section 7.2 for more information.

[2] If reserved configurations are selected, then the state of the pin will be undefined and the pin may be driven. These selections are reserved for future expansion and should not be used.

Table 50. GPIO Port A Multiplexing 1 (GPAMUX1) Register Field Descriptions (continued)

Bits	Field	Value	Description [1]
23-22	GPIO11		Configure the GPIO11 pin as:
		00	GPIO11 - General purpose I/O 11 (default) (I/O)
		01	EPWM6B - ePWM 6 output B (O)
		10	SCIRXDB - SCI-B receive (I)
		11	ECAP4 - eCAP4. (I/O)
21-20	GPIO10		Configure the GPIO10 pin as:
		00	GPIO10 - General purpose I/O 10 (default) (I/O)
		01	EPWM6A - ePWM6 output A (O)
		10	CANRXB - eCAN-B receive (I)
		11	$\overline{\text{ADCSOCBO}}$ - ADC Start of conversion B (O)
19-18	GPIO9		Configure the GPIO9 pin as:
		00	GPIO9 - General purpose I/O 9 (default) (I/O)
		01	EPWM5B - ePWM5 output B
		10	SCITXDB - SCI-B transmit (O)
		11	ECAP3 - eCAP3 (I/O)
17-16	GPIO8		Configure the GPIO8 pin as:
		00	GPIO8 - General purpose I/O 8 (default) (I/O)
		01	EPWM5A - ePWM5 output A (O)
		10	CANTXB - eCAN-B transmit (O)
		11	$\overline{\text{ADCSOCAO}}$ - ADC Start of conversion A
15-14	GPIO7		Configure the GPIO7 pin as:
		00	GPIO7 - General purpose I/O 7 (default) (I/O)
		01	EPWM4B - ePWM4 output B (O)
		10	MCLKRA - McBSP-A Receive clock (I/O)
		11	ECAP2 - eCAP2 (I/O)
13-12	GPIO6		Configure the GPIO6 pin as:
		00	GPIO6 - General purpose I/O 6 (default)
		01	EPWM4A - ePWM4 output A (O)
		10	EPWMSYNCI - ePWM Synch-in (I)
		11	EPWMSYNCO - ePWM Synch-out (O)
11-10	GPIO5		Configure the GPIO5 pin as:
		00	GPIO5 - General purpose I/O 5 (default) (I/O)
		01	EPWM3B - ePWM3 output B
		10	MFSRA - McBSP-A Receive frame synch (I/O)
		11	ECAP1 - eCAP1 (I/O)
9-8	GPIO4		Configure the GPIO4 pin as:
		00	GPIO4 - General purpose I/O 4 (default) (I/O)
		01	EPWM3A - ePWM3 output A (O)
		10	Reserved. [3]
		11	Reserved. [3]
7-6	GPIO3		Configure the GPIO3 pin as:
		00	GPIO3 - General purpose I/O 3 (default) (I/O)
		01	EPWM2B - ePWM2 output B (O)
		10	ECAP5 - eCAP5 (I/O)
		11	MCLKRB - McBSP-B receive clock (I/O)

[3] If reserved configurations are selected, then the state of the pin will be undefined and the pin may be driven. These selections are reserved for future expansion and should not be used.

Table 50. GPIO Port A Multiplexing 1 (GPAMUX1) Register Field Descriptions (continued)

Bits	Field	Value	Description [1]
5-4	GPIO2		Configure the GPIO2 pin as:
		00	GPIO2 (I/O) General purpose I/O 2 (default) (I/O)
		01	EPWM2A - ePWM2 output A (O)
		10	Reserved. [3]
		11	Reserved. [3]
3-2	GPIO1		Configure the GPIO1 pin as:
		00	GPIO1 - General purpose I/O 1 (default) (I/O)
		01	EPWM1B - ePWM1 output B (O)
		10	ECAP6 - eCAP6 (I/O)
		11	MFSRB - McBSP-B Receive Frame Synch (I/O)
1-0	GPIO0		Configure the GPIO0 pin as:
		00	GPIO0 - General purpose I/O 0 (default) (I/O)
		01	EPWM1A - ePWM1 output A (O)
		10	Reserved. [3]
		11	Reserved. [3]

Figure 48. GPIO Port A MUX 2 (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-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0	
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-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 51. GPIO Port A MUX 2 (GPAMUX2) Register Field Descriptions

Bits	Field	Value	Description [1]
31-30	GPIO31		Configure the GPIO31 pin as:
		00	GPIO31 - General purpose I/O 31 (default) (I/O)
		01	CANTXA - eCAN-A transmit (O)
		10 or 11	XA17 - External interface address line 17 (O)
29-28	GPIO30		Configure the GPIO30 pin as:
		00	GPIO30 (I/O) General purpose I/O 30 (default) (I/O)
		01	CANRXA - eCAN-A receive (I)
		10 or 11	XA18 - External interface address line 18
27-26	GPIO29		Configure the GPIO29 pin as:
		00	GPIO29 (I/O) General purpose I/O 29 (default) (I/O)
		01	SCITXDA - SCI-A transmit. (O)
		10 or 11	XA19 - External Interface address line 19 (O)
25-24	GPIO28		Configure the GPIO28 pin as:
		00	GPIO28 (I/O) General purpose I/O 28 (default) (I/O)
		01	SCIRXDA - SCI-A receive (I)
		10 or 11	$\overline{\text{XZCS6}}$ - External interface zone 6 chip select (O)

[1] This register is EALLOW protected. See Section 7.2 for more information.

Table 51. GPIO Port A MUX 2 (GPAMUX2) Register Field Descriptions (continued)

Bits	Field	Value	Description [1]
23-22	GPIO27		Configure the GPIO27 pin as:
		00	GPIO27 - General purpose I/O 27 (default) (I/O)
		01	ECAP4 - eCAP4. (I/O)
		10	EQEP2S - eQEP2 strobe (I/O)
		11	MFSXB - McBSP-B Transmit Frame Synch (I/O)
21-20	GPIO26		Configure the GPIO26 pin as:
		00	GPIO26 - General purpose I/O 26 (default) (I/O)
		01	ECAP3 - eCAP3. (I/O)
		10	EQEP2I - eQEP2 index. (I/O)
		11	MCLKXB - McBSP-B Transmit Clock (I/O)
19-18	GPIO25		Configure the GPIO25 pin as:
		00	GPIO25 - General purpose I/O 25 (default) (I/O)
		01	ECAP2 - eCAP2 (I/O)
		10	EQEP2B - eQEP2 input B (I)
		11	MDRB - McBSP-B data receive (O)
17-16	GPIO24		Configure the GPIO24 pin as:
		00	GPIO24 - General purpose I/O 24 (default) (I/O)
		01	ECAP1 - eCAP1 (I/O)
		10	EQEP2A - eQEP2 input A. (I)
		11	MDXB - McBSP-B data transmit (O)
15-14	GPIO23		Configure the GPIO23 pin as:
		00	GPIO23 - General purpose I/O 23 (default) (I/O)
		01	EQEP1I - eQEP1 index (I/O)
		10	MFSXA - McBSP-A transmit frame synch (I/O)
		11	SCIRXDB - SCI-B receive (I/O)
13-12	GPIO22		Configure the GPIO22 pin as:
		00	GPIO22 - General purpose I/O 22 (default) (I/O)
		01	EQEP1S - eQEP1 strobe (I/O)
		10	MCLKXA - McBSP-A transmit clock (I/O)
		11	SCITXDB - SCI-B transmit (O)
11-10	GPIO21		Configure the GPIO21 pin as:
		00	GPIO21 - General purpose I/O 21 (default) (I/O)
		01	EQEP1B - eQEP1 input B (I)
		10	MDRA - McBSP-A data receive (I)
		11	CANRXB - eCAN-B receive (I)
9-8	GPIO20		Configure the GPIO20 pin as:
		00	GPIO20 - General purpose I/O 22 (default) (I/O)
		01	EQEP1A - eQEP1 input A (I)
		10	MDXA - McBSP-A data transmit (O)
		11	CANTXB - eCAN-B transmit (O)
7-6	GPIO19		Configure the GPIO19 pin as:
		00	GPIO19 - General purpose I/O 19 (default) (I/O)
		01	$\overline{\text{SPISTEA}}$ - SPI-A slave transmit enable (I/O)
		10	SCIRXDB - SCI-B receive (I)
		11	CANTXA - eCAN-A Transmit (O)

Table 51. GPIO Port A MUX 2 (GPAMUX2) Register Field Descriptions (continued)

Bits	Field	Value	Description (1)
5-4	GPIO18		Configure the GPIO18 pin as:
		00	GPIO18 - General purpose I/O 18 (default) (I/O)
		01	SPICLKA - SPI-A clock (I/O)
		10	SCITXDB - SCI-B transmit. (O)
		11	CANRXA - eCAN-A Receive (I)
3-2	GPIO17		Configure the GPIO17 pin as:
		00	GPIO17 - General purpose I/O 17 (default) (I/O)
		01	SPISOMIA - SPI-A slave-out, master-in (I/O)
		10	CANRXB eCAN-B receive (I)
		11	$\overline{TZ6}$ - Trip zone 6 (I)
1-0	GPIO16		Configure the GPIO16 pin as:
		00	GPIO16 - General purpose I/O 16 (default) (I/O)
		01	SPISIMOA - SPI-A slave-in, master-out (I/O),
		10	CANTXB - eCAN-B transmit. (O)
		11	$\overline{TZ5}$ - Trip zone 5 (I)

Figure 49. GPIO Port B MUX 1 (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-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO39		GPIO38		GPIO37		GPIO36		GPIO35		GPIO34		GPIO33		GPIO32	
R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 52. GPIO Port B MUX 1 (GPBMUX1) Register Field Descriptions

Bit	Field	Value	Description
31:30	GPIO47		Configure this pin as:
		00	GPIO 47 - general purpose I/O 47 (default)
		01	Reserved
		10 or 11	XA7 - External interface (XINTF) address line 7 (O)
29:28	GPIO46		Configure this pin as:
		00	GPIO 46 - general purpose I/O 46 (default)
		01	Reserved
		10 or 11	XA6 - External interface (XINTF) address line 6 (O)
27:26	GPIO45		Configure this pin as:
		00	GPIO 45 - general purpose I/O 45 (default)
		01	Reserved
		10 or 11	XA5 - External interface (XINTF) address line 5 (O)
25:24	GPIO44		Configure this pin:
		00	GPIO 44 - general purpose I/O 44 (default)
		01	Reserved
		10 or 11	XA4 - External interface (XINTF) address line 4 (O)

Table 52. GPIO Port B MUX 1 (GPBMUX1) Register Field Descriptions (continued)

Bit	Field	Value	Description
23:22	GPIO43		Configure this pin as:
		00	GPIO 43 - general purpose I/O 43 (default)
		01	Reserved
		10 or 11	XA3 - External interface (XINTF) address line 3 (O)
21:20	GPIO42		Configure this pin as:
		00	GPIO 42 - general purpose I/O 42 (default)
		01	Reserved
		10 or 11	XA2 - External interface (XINTF) address line 2 (O)
19:18	GPIO41		Configure this pin as:
		00	GPIO 41 - general purpose I/O 41 (default)
		01	Reserved
		10 or 11	XA1 - External interface (XINTF) address line 1 (O)
17:16	GPIO40		Configure this pin as:
		00	GPIO 40 - general purpose I/O 40 (default)
		01	Reserved
		10 or 11	XA0/$\overline{\text{XWE1}}$ - External interface (XINTF) address line 1 or external interface write enable strobe 1 (O)
15:14	GPIO39		Configure this pin as:
		00	GPIO 39 - general purpose I/O 39 (default)
		01	Reserved
		10 or 11	XA16 - External interface (XINTF) address line 16 (O)
13:12	GPIO38		Configure this pin as:
		00	GPIO 38 - general purpose I/O 38 (default)
		01	Reserved
		10 or 11	$\overline{\text{XWE0}}$ - External interface write enable strobe 0
11:10	GPIO37		Configure this pin as:
		00	GPIO 37 - general purpose I/O 37 (default)
		01	ECAP2 - Enhanced capture input/output 2 (I/O)
		10 or 11	$\overline{\text{XZCS7}}$ - External interface zone 7 chip select (O)
9:8	GPIO36		Configure this pin as:
		00	GPIO 36 - general purpose I/O 36 (default)
		01	SCIRXDA - SCI-A receive data (I)
		10 or 11	$\overline{\text{XZCS0}}$ - External interface zone 0 chip select (O)
7:6	GPIO35		Configure this pin as:
		00	GPIO 35 - general purpose I/O 35 (default)
		01	SCITXDA - SCI-A transmit data (O)
		10 or 11	XR/$\overline{\text{W}}$ - Read Not Write Strobe. Normally held high. When low, XR/$\overline{\text{W}}$ indicates write cycle is active; when high, XR/$\overline{\text{W}}$ indicates read cycle is active.
5:4	GPIO34		Configure this pin as:
		00	GPIO 34 - general purpose I/O 34 (default)
		01	ECAPI - Enhanced capture input/output 1 (I/O)
		10 or 11	XREADY - External interface ready signal

Table 52. GPIO Port B MUX 1 (GPBMUX1) Register Field Descriptions (continued)

Bit	Field	Value	Description
3:2	GPIO33		Configure this pin as:
		00	GPIO 33 - general purpose I/O 33 (default)
		01	SCLA - I2C clock open drain bidirectional port (I/O)
		10	EPWMSYNCO - External ePWM sync pulse output (O)
		11	$\overline{\text{ADCSOCBO}}$ - ADC start-of-conversion B (O)
1:0	GPIO32		Configure this pin as:
		00	GPIO 32 - general purpose I/O 32 (default)
		01	SDAA - I2C data open drain bidirectional port (I/O)
		10	EPWMSYNCI - External ePWM sync pulse input (I)
		11	$\overline{\text{ADCSOCAO}}$ - ADC start-of-conversion A (O)

Figure 50. GPIO Port B MUX 2 (GPBMUX2) Register

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
GPIO63	GPIO62	GPIO61	GPIO60	GPIO59	GPIO58	GPIO57	GPIO56
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
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-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 53. GPIO Port B MUX 2 (GPBMUX2) Register Field Descriptions

Bit	Field	Value	Description
31:30	GPIO63		Configure this pin as:
		00	GPIO 63 - general purpose I/O 63 (default)
		01	SCITXDC - SCI-C transmit data (O)
		10 or 11	XD16 - External interface data line 16 (I/O)
29:28	GPIO62		Configure this pin as:
		00	GPIO 62 - general purpose I/O 62 (default)
		01	SCIRXDC - SCI-C receive data (I)
		10 or 11	XD17 - External interface data line 17 (I/O)
27:26	GPIO61		Configure this pin as:
		00	GPIO 61 - general purpose I/O 61 (default)
		01	MFSRB - McBSP-B receive frame synch (I/O)
		10 or 11	XD18 - External interface data line 18 (I/O)
25:24	GPIO60		Configure this pin as:
		00	GPIO 60 - general purpose I/O 60 (default)
		01	MCLKRB - McBSP-B receive clock (I/O)
		10 or 11	XD19 - External interface data line 19 (I/O)
23:22	GPIO59		Configure this pin as:
		00	GPIO 59 - general purpose I/O 59 (default)
		01	MFSRA - McBSP-A receive frame synch (I/O)
		10 or 11	XD20 - External interface data line 20 (I/O)

Table 53. GPIO Port B MUX 2 (GPBMUX2) Register Field Descriptions (continued)

Bit	Field	Value	Description
21:20	GPIO58		Configure this pin as:
		00	GPIO 58 - general purpose I/O 58 (default)
		01	MCLKRA - McBSP-A receive clock (I/O)
		10 or 11	XD21 - External interface data line 21 (I/O)
19:18	GPIO57		Configure this pin as:
		00	GPIO 57 - general purpose I/O 57 (default)
		01	$\overline{\text{SPISTEA}}$ - SPI-A slave transmit enable (I/O)
		10 or 11	XD22 - External interface data line 22 (I/O)
17:16	GPIO56		Configure this pin as:
		00	GPIO 56 - general purpose I/O 56 (default)
		01	SPICLKA - SPI-A clock input/output (I/O)
		10 or 11	XD23 - External interface data line 23 (I/O)
15:14	GPIO55		Configure this pin as:
		00	GPIO 55 - general purpose I/O 55 (default)
		01	SPISOMIA - SPI-A slave out, master in (I/O)
		10 or 11	XD24 - External interface data line 24 (I/O)
13:12	GPIO54		Configure this pin as:
		00	GPIO 54 - general purpose I/O 54 (default)
		01	SPISIMOA - SPI slave in, master out (I/O)
		10 or 11	XD25 - External interface data line 25 (I/O)
11:10	GPIO53		Configure this pin as:
		00	GPIO 53 - general purpose I/O 53 (default)
		01	EQEP1I - Enhanced QEP1 index (I/O)
		10 or 11	XD26 - External interface data line 26 (I/O)
9:8	GPIO52		Configure this pin as:
		00	GPIO 52 - general purpose I/O 52 (default)
		01	EQEP1S - Enhanced QEP1 strobe (I/O)
		10 or 11	XD27External interface data line 27 (I/O)
7:6	GPIO51		Configure this pin as:
		00	GPIO 51 - general purpose I/O 51 (default)
		01	EQEP1B - Enhanced QEP1 input B (I)
		10 or 11	XD28 - External interface data line 28 (I/O)
5:4	GPIO50		Configure this pin as:
		00	GPIO 50 - general purpose I/O 50 (default)
		01	EQEP1A - Enhanced QEP1 input A (I)
		10 or 11	XD29 - External interface data line 29 (I/O)
3:2	GPIO49		Configure this pin as:
		00	GPIO 49 - general purpose I/O 49 (default)
		01	ECAP6 - Enhanced Capture input/output 6 (I/O)
		10 or 11	XD30 - External interface data line 30 (I/O)

Table 53. GPIO Port B MUX 2 (GPBMUX2) Register Field Descriptions (continued)

Bit	Field	Value	Description
1:0	GPIO48		Configure this pin as:
		00	GPIO 48 - general purpose I/O 48 (default)
		01	ECAP5 - Enhanced Capture input/output 5 (I/O)
		10 or 11	XD31 - External interface data line 31 (I/O)

Figure 51. GPIO Port C MUX 1 (GPCMUX1) Register

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
GPIO79	GPIO78	GPIO77	GPIO76	GPIO75	GPIO74	GPIO73	GPIO72
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
GPIO71	GPIO70	GPIO69	GPIO68	GPIO67	GPIO66	GPIO65	GPIO64
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 54. GPIO Port C MUX 1 (GPCMUX1) Register Field Descriptions

Bit	Field	Value	Description
31:30	GPIO79		Configure this pin as:
		00 or 01	GPIO 79 - general purpose I/O 79 (default)
		10 or 11	XD0 - External interface data line 0 (O)
29:28	GPIO78		Configure this pin as:
		00 or 01	GPIO 78 - general purpose I/O 78 (default)
		10 or 11	XD1 - External interface data line 1 (O)
27:26	GPIO77		Configure this pin as:
		00 or 01	GPIO 77 - general purpose I/O 77 (default)
		10 or 11	XD2 - External interface data line 2 (O)
25:24	GPIO76		Configure this pin as:
		00 or 01	GPIO 76 - general purpose I/O 76 (default)
		10 or 11	XD3 - External interface data line 3 (O)
23:22	GPIO75		Configure this pin as:
		00 or 01	GPIO 75 - general purpose I/O 75 (default)
		10 or 11	XD4 - External interface data line 4 (O)
21:20	GPIO74		Configure this pin as:
		00 or 01	GPIO 74 - general purpose I/O 74 (default)
		10 or 11	XD5 - External interface data line 5(O)
19:18	GPIO73		Configure this pin as:
		00 or 01	GPIO 73 - general purpose I/O 73 (default)
		10 or 11	XD6 - External interface data line 6 (O)
17:16	GPIO72		Configure this pin as:
		00 or 01	GPIO 72 - general purpose I/O 72 (default)
		10 or 11	XD7 - External interface data line 7 (O)
15:14	GPIO71		Configure this pin as:
		00 or 01	GPIO 71 - general purpose I/O 71 (default)
		10 or 11	XD8 - External interface data line 8 (O)
13:12	GPIO70		Configure this pin as:
		00 or 01	GPIO 70 - general purpose I/O 70 (default)
		10 or 11	XD9 - External interface data line 9 (O)

Table 54. GPIO Port C MUX 1 (GPCMUX1) Register Field Descriptions (continued)

Bit	Field	Value	Description
11:10	GPIO69		Configure this pin as:
		00 or 01	GPIO 69 - general purpose I/O 69 (default)
		10 or 11	XD10 - External interface data line 10 (O)
9:8	GPIO68		Configure this pin as:
		00 or 01	GPIO 68 - general purpose I/O 68 (default)
		10 or 11	XD11 - External interface data line 11 (O)
7:6	GPIO67		Configure this pin as:
		00 or 01	GPIO 67 - general purpose I/O 67 (default)
		10 or 11	XD12 - External interface data line 12 (O)
5:4	GPIO66		Configure this pin as:
		00 or 01	GPIO 66 - general purpose I/O 66 (default)
		10 or 11	XD13 - External interface data line 13 (O)
3:2	GPIO65		Configure this pin as:
		00 or 01	GPIO 65 - general purpose I/O 65 (default)
		10 or 11	XD14 - External interface data line 14 (O)
1:0	GPIO64		Configure this pin as:
		00 or 01	GPIO 64 - general purpose I/O 64 (default)
		10 or 11	XD15 - External interface data line 15 (O)

Figure 52. GPIO Port C MUX 2 (GPCMUX2) Register

31	16
Reserved	
R-0	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIO87		GPIO86		GPIO85		GPIO84		GPIO83		GPIO82		GPIO81		GPIO80	
R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 55. GPIO Port C MUX 2 (GPCMUX2) Register Field Descriptions

Bit	Field	Value	Description
31:16	Reserved		
15:14	GPIO87		Configure this pin as:
		00 or 01	GPIO 87 - general purpose I/O 87 (default)
		10 or 11	XA15 - External interface address line 15 (O)
13:12	GPIO86		Configure this pin as:
		00 or 01	GPIO 86 - general purpose I/O 86 (default)
		10 or 11	XA14 - External interface address line 14 (O)
11:10	GPIO85		Configure this pin as:
		00 or 01	GPIO 85 - general purpose I/O 85 (default)
		10 or 11	XA13 - External interface address line 13 (O)

Table 55. GPIO Port C MUX 2 (GPCMUX2) Register Field Descriptions (continued)

Bit	Field	Value	Description
9:8	GPIO84		Configure this pin as:
		00 or 01	GPIO 84 - general purpose I/O 84 (default)
		10 or 11	XA12 - External interface address line 12 (O)
7:6	GPIO83		Configure this pin as:
		00 or 01	GPIO 83 - general purpose I/O 83 (default)
		10 or 11	XA11 - External interface address line 11 (O)
5:4	GPIO82		Configure this pin as:
		00 or 01	GPIO 82 - general purpose I/O 82 (default)
		10 or 11	XA10 - External interface address line 10 (O)
3:2	GPIO81		Configure this pin as:
		00 or 01	GPIO 81 - general purpose I/O 81 (default)
		10 or 11	XA9 - External interface address line 9 (O)
1:0	GPIO80		Configure this pin as:
		00 or 01	GPIO 80 - general purpose I/O 80(default)
		10 or 11	XA8 - External interface address line 8 (O)

Figure 53. GPIO Port A Qualification Control (GPACTRL) Register

31	24	23	16
QUALPRD3		QUALPRD2	
R/W-0		R/W-0	
15	8	7	0
QUALPRD1		QUALPRD0	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

The GPxCTRL registers specify the sampling period for input pins when configured for input qualification using a window of 3 or 6 samples. The sampling period is the amount of time between qualification samples relative to the period of SYSCLKOUT. The number of samples is specified in the GPxQSELn registers.

Table 56. GPIO Port A Qualification Control (GPACTRL) Register Field Descriptions

Bits	Field	Value	Description (1)
31-24	QUALPRD3		Specifies the sampling period for pins GPIO24 to GPIO31.
		0x00	Sampling Period = $T_{SYSCLKOUT}$ (2)
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		. . .	. . .
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$
23-16	QUALPRD2		Specifies the sampling period for pins GPIO16 to GPIO23.
		0x00	Sampling Period = $T_{SYSCLKOUT}$ (2)
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		. . .	. . .
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$
15-8	QUALPRD1		Specifies the sampling period for pins GPIO8 to GPIO15.
		0x00	Sampling Period = $T_{SYSCLKOUT}$ (2)
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		. . .	. . .
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$
7-0	QUALPRD0		Specifies the sampling period for pins GPIO0 to GPIO7.
		0x00	Sampling Period = $T_{SYSCLKOUT}$ (2)
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		. . .	. . .
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$

(1) This register is EALLOW protected. See Section 7.2 for more information.

(2) $T_{SYSCLKOUT}$ indicates the period of SYSCLKOUT.

Figure 54. GPIO Port B Qualification Control (GPBCTRL) Register

31	24	23	16
QUALPRD3		QUALPRD2	
R/W-0		R/W-0	
15	8	7	0
QUALPRD1		QUALPRD0	
R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 57. GPIO Port B Qualification Control (GPBCTRL) Register Field Descriptions

Bits	Field	Value	Description (1)
31-24	QUALPRD3		Specifies the sampling period for pins GPIO56 to GPIO63
		0x00	Sampling Period = $T_{SYSCLKOUT}$ (2)
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		. . .	. . .
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$
23-16	QUALPRD2		Specifies the sampling period for pins GPIO48 to GPIO55
		0x00	Sampling Period = $T_{SYSCLKOUT}$ (2)
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		. . .	. . .
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$
15-8	QUALPRD1		Specifies the sampling period for pins GPIO40 to GPIO47
		0x00	Sampling Period = $T_{SYSCLKOUT}$ (2)
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		. . .	. . .
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$
7-0	QUALPRD0		Specifies the sampling period for pins GPIO32 to GPIO39
		0x00	Sampling Period = $T_{SYSCLKOUT}$ (2)
		0x01	Sampling Period = 2 × $T_{SYSCLKOUT}$
		0x02	Sampling Period = 4 × $T_{SYSCLKOUT}$
		. . .	. . .
		0xFF	Sampling Period = 510 × $T_{SYSCLKOUT}$

(1) This register is EALLOW protected. See Section 7.2 for more information.
(2) $T_{SYSCLKOUT}$ indicates the period of SYSCLKOUT.

Figure 55. GPIO Port A Qualification Select 1 (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-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
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-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 58. GPIO Port A Qualification Select 1 (GPAQSEL1) Register Field Descriptions

Bits	Field	Value	Description (1)
31-0	GPIO15-GPIO0		Select input qualification type for GPIO0 to GPIO15. The input qualification of each GPIO input is controlled by two bits as shown in Figure 55.
		00	Synchronize to SYSCLKOUT only. Valid for both peripheral and GPIO pins.
		01	Qualification using 3 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPACTRL register.
		10	Qualification using 6 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPACTRL register.
		11	Asynchronous. (no synchronization or qualification). This option applies to pins configured as peripherals only. If the pin is configured as a GPIO input, then this option is the same as 0,0 or synchronize to SYSCLKOUT.

(1) This register is EALLOW protected. See Section 7.2 for more information.

Figure 56. GPIO Port A Qualification Select 2 (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-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
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-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 59. GPIO Port A Qualification Select 2 (GPAQSEL2) Register Field Descriptions

Bits	Field	Value	Description (1)
31-0	GPIO31-GPIO16		Select input qualification type for GPIO16 to GPIO31. The input qualification of each GPIO input is controlled by two bits as shown in Figure 56.
		00	Synchronize to SYSCLKOUT only. Valid for both peripheral and GPIO pins.
		01	Qualification using 3 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPACTRL register.
		10	Qualification using 6 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPACTRL register.
		11	Asynchronous. (no synchronization or qualification). This option applies to pins configured as peripherals only. If the pin is configured as a GPIO input, then this option is the same as 0,0 or synchronize to SYSCLKOUT.

(1) This register is EALLOW protected. See Section 7.2 for more information.

Figure 57. GPIO Port B Qualification Select 1 (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-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
GPIO39	GPIO38	GPIO37	GPIO36	GPIO35	GPIO34	GPIO33	GPIO32
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 60. GPIO Port B Qualification Select 1 (GPBQSEL1) Register Field Descriptions

Bits	Field	Value	Description [1]
31-0	GPIO47-GPIO32		Select input qualification type for GPIO32 to GPIO47. The input qualification of each GPIO input is controlled by two bits as shown in Figure 55.
		00	Synchronize to SYSCLKOUT only. Valid for both peripheral and GPIO pins.
		01	Qualification using 3 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPACTRL register.
		10	Qualification using 6 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPACTRL register.
		11	Asynchronous. (no synchronization or qualification). This option applies to pins configured as peripherals only. If the pin is configured as a GPIO input, then this option is the same as 0,0 or synchronize to SYSCLKOUT.

(1) This register is EALLOW protected. See Section 7.2 for more information.

Figure 58. GPIO Port B Qualification Select 2 (GPBQSEL2) Register

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
GPIO63	GPIO62	GPIO61	GPIO60	GPIO59	GPIO58	GPIO57	GPIO56
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
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-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 61. GPIO Port B Qualification Select 2 (GPBQSEL2) Register Field Descriptions

Bits	Field	Value	Description [1]
31-0	GPIO63-GPIO48		Select input qualification type for GPIO48 to GPIO63. The input qualification of each GPIO input is controlled by two bits as shown in Figure 56.
		00	Synchronize to SYSCLKOUT only. Valid for both peripheral and GPIO pins.
		01	Qualification using 3 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPACTRL register.
		10	Qualification using 6 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPACTRL register.
		11	Asynchronous. (no synchronization or qualification). This option applies to pins configured as peripherals only. If the pin is configured as a GPIO input, then this option is the same as 0,0 or synchronize to SYSCLKOUT.

(1) This register is EALLOW protected. See Section 7.2 for more information.

The GPADIR and GPBDIR registers control the direction of the pins when they are configured as a GPIO in the appropriate MUX register. The direction register has no effect on pins configured as peripheral functions.

Figure 59. GPIO Port A Direction (GPADIR) Register

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

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 62. GPIO Port A Direction (GPADIR) Register Field Descriptions

Bits	Field	Value	Description [1]
31-0	GPIO31-GPIO0		Controls direction of GPIO Port A pins when the specified pin is configured as a GPIO in the appropriate GPAMUX1 or GPAMUX2 register.
		0	Configures the GPIO pin as an input. (default)
		1	Configures the GPIO pin as an output
			The value currently in the GPADAT output latch is driven on the pin. To initialize the GPADAT latch prior to changing the pin from an input to an output, use the GPASET, GPACLEAR, and GPATOGGLE registers.

(1) This register is EALLOW protected. See Section 7.2 for more information.

Figure 60. GPIO Port B Direction (GPBDIR) Register

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

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 63. GPIO Port B Direction (GPBDIR) Register Field Descriptions

Bits	Field	Value	Description [1]
31-0	GPIO63-GPIO32		Controls direction of GPIO pin when GPIO mode is selected. Reading the register returns the current value of the register setting
		0	Configures the GPIO pin as an input. (default)
		1	Configures the GPIO pin as an output

(1) This register is EALLOW protected. See Section 7.2 for more information.

Figure 61. GPIO Port C Direction (GPCDIR) Register

31							24
Reserved							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
GPIO87	GPIO86	GPIO85	GPIO84	GPIO83	GPIO82	GPIO81	GPIO80
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
GPIO79	GPIO78	GPIO77	GPIO76	GPIO75	GPIO74	GPIO73	GPIO72
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
GPIO71	GPIO70	GPIO69	GPIO68	GPIO67	GPIO66	GPIO65	GPIO64
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 64. GPIO Port C Direction (GPCDIR) Register Field Descriptions

Bits	Field	Value	Description [1]
31-0	GPIO87-GPIO64		Controls direction of GPIO pin when GPIO mode is selected. Reading the register returns the current value of the register setting
		0	Configures the GPIO pin as an input. (default)
		1	Configures the GPIO pin as an output

(1) This register is EALLOW protected. See Section 7.2 for more information.

The pullup disable (GPxPUD) registers allow you to specify which pins should have an internal pullup resister enabled. The internal pullups on the pins that can be configured as ePWM outputs(GPIO0-GPIO11) are all disabled asynchronously when the external reset signal ($\overline{\text{XRS}}$) is low. The internal pullups on all other pins are enabled on reset. When coming out of reset, the pullups remain in their default state until you enable or disable them selectively in software by writing to this register. The pullup configuration applies both to pins configured as I/O and those configured as peripheral functions.

Figure 62. GPIO Port A Pullup Disable (GPAPUD) Registers

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

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 65. GPIO Port A Internal Pullup Disable (GPAPUD) Register Field Descriptions

Bits	Field	Value	Description [1]
31-0	GPIO31-GPIO0		Configure the internal pullup resister on the selected GPIO Port A pin. Each GPIO pin corresponds to one bit in this register.
		0	Enable the internal pullup on the specified pin. (default for GPIO12-GPIO31)
		1	Disable the internal pullup on the specified pin. (default for GPIO0-GPIO11)

(1) This register is EALLOW protected. See Section 7.2 for more information.

Figure 63. GPIO Port B Pullup Disable (GPBPUD) Registers

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

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 66. GPIO Port B Internal Pullup Disable (GPBPUD) Register Field Descriptions

Bits	Field	Value	Description [1]
31-0	GPIO63-GPIO32		Configure the internal pullup resister on the selected GPIO Port B pin. Each GPIO pin corresponds to one bit in this register.
		0	Enable the internal pullup on the specified pin. (default)
		1	Disable the internal pullup on the specified pin.

(1) This register is EALLOW protected. See Section 7.2 for more information.

Figure 64. GPIO Port C Pullup Disable (GPCPUD) Registers

31							24
Reserved							
R/W-0							
23	22	21	20	19	18	17	16
GPIO87	GPIO86	GPIO85	GPIO84	GPIO83	GPIO82	GPIO81	GPIO80
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
GPIO79	GPIO78	GPIO77	GPIO76	GPIO75	GPIO74	GPIO73	GPIO72
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
GPIO71	GPIO70	GPIO69	GPIO68	GPIO67	GPIO66	GPIO65	GPIO64
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 67. GPIO Port C Internal Pullup Disable (GPCPUD) Register Field Descriptions

Bits	Field	Value	Description [1]
31-0	GPIO87-GPIO64		Configure the internal pullup resister on the selected GPIO Port C pin. Each GPIO pin corresponds to one bit in this register.
		0	Enable the internal pullup on the specified pin.
		1	Disable the internal pullup on the specified pin.

(1) This register is EALLOW protected. See Section 7.2 for more information.

The GPIO data registers indicate the current status of the GPIO pin, irrespective of which mode the pin is in. Writing to this register will set the respective GPIO pin high or low if the pin is enabled as a GPIO output, otherwise the value written is latched but ignored. The state of the output register latch will remain in its current state until the next write operation. A reset will clear all bits and latched values to zero. The value read from the GPxDAT registers reflect the state of the pin (after qualification), not the state of the output latch of the GPxDAT register.

Typically the DAT registers are used for reading the current state of the pins. To easily modify the output level of the pin refer to the SET, CLEAR and TOGGLE registers.

Figure 65. GPIO Port A Data (GPADAT) Register

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

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset(1)

(1) x = The state of the GPADAT register is unknown after reset. It depends on the level of the pin after reset.

Table 68. GPIO Port A Data (GPADAT) Register Field Descriptions

Bits	Field	Value	Description
31-0	GPIO31-GPIO0		Each bit corresponds to one GPIO port A pin (GPIO0-GPIO31) as shown in Figure 65.
		0	Reading a 0 indicates that the state of the pin is currently low, irrespective of the mode the pin is configured for.
			Writing a 0 will force an output of 0 if the pin is configured as a GPIO output in the appropriate GPAMUX1/2 and GPADIR registers; otherwise, the value is latched but not used to drive the pin.
		1	Reading a 1 indicates that the state of the pin is currently high irrespective of the mode the pin is configured for.
			Writing a 1will force an output of 1if the pin is configured as a GPIO output in the appropriate GPAMUX1/2 and GPADIR registers; otherwise, the value is latched but not used to drive the pin.

Figure 66. GPIO Port B Data (GPBDAT) Register

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

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset(1)

(1) x = The state of the GPADAT register is unknown after reset. It depends on the level of the pin after reset.

Table 69. GPIO Port B Data (GPBDAT) Register Field Descriptions

Bit	Field	Value	Description
31-0	GPIO63-GPIO32		Each bit corresponds to one GPIO port B pin (GPIO32-GPIO63) as shown in Figure 66.
		0	Reading a 0 indicates that the state of the pin is currently low, irrespective of the mode the pin is configured for.
			Writing a 0 will force an output of 0 if the pin is configured as a GPIO output in the appropriate GPBMUX1 and GPBDIR registers; otherwise, the value is latched but not used to drive the pin.
		1	Reading a 1 indicates that the state of the pin is currently high irrespective of the mode the pin is configured for.
			Writing a 1 will force an output of 1 if the pin is configured as a GPIO output in the GPBMUX1 and GPBDIR registers; otherwise, the value is latched but not used to drive the pin.

Figure 67. GPIO Port C Data (GPCDAT) Register

31							24
Reserved							
R-0							
23	22	21	20	19	18	17	16
GPIO87	GPIO86	GPIO85	GPIO84	GPIO83	GPIO82	GPIO81	GPIO80
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
15	14	13	12	11	10	9	8
GPIO79	GPIO78	GPIO77	GPIO76	GPIO75	GPIO74	GPIO73	GPIO72
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
7	6	5	4	3	2	1	0
GPIO71	GPIO70	GPIO69	GPIO68	GPIO67	GPIO66	GPIO65	GPIO64
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset(1)

(1) x = The state of the GPADAT register is unknown after reset. It depends on the level of the pin after reset.

Table 70. GPIO Port C Data (GPCDAT) Register Field Descriptions

Bit	Field	Value	Description
31-3	Reserved		Reserved
2-0	GPIO87-GPIO64		Each bit corresponds to one GPIO port B pin (GPIO64-GPIO87) as shown in Figure 67
		0	Reading a 0 indicates that the state of the pin is currently low, irrespective of the mode the pin is configured for.
			Writing a 0 will force an output of 0 if the pin is configured as a GPIO output in the appropriate GPCMUX1 and GPCDIR registers; otherwise, the value is latched but not used to drive the pin.
		1	Reading a 1 indicates that the state of the pin is currently high irrespective of the mode the pin is configured for.
			Writing a 1 will force an output of 1 if the pin is configured as a GPIO output in the GPCMUX1 and GPCDIR registers; otherwise, the value is latched but not used to drive the pin.

Figure 68. GPIO Port A Set, Clear and Toggle (GPASET, GPACLEAR, GPATOGGLE) Registers

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

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 71. GPIO Port A Set (GPASET) Register Field Descriptions

Bits	Field	Value	Description
31-0	GPIO31-GPIO0		Each GPIO port A pin (GPIO0-GPIO31) corresponds to one bit in this register as shown in Figure 68.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to high. If the pin is configured as a GPIO output then it will be driven high. If the pin is not configured as a GPIO output then the latch is set high but the pin is not driven.

Table 72. GPIO Port A Clear (GPACLEAR) Register Field Descriptions

Bits	Field	Value	Description
31-0	GPIO31 - GPIO0		Each GPIO port A pin (GPIO0-GPIO31) corresponds to one bit in this register as shown in Figure 68.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to low. If the pin is configured as a GPIO output then it will be driven low. If the pin is not configured as a GPIO output then the latch is cleared but the pin is not driven.

Table 73. GPIO Port A Toggle (GPATOGGLE) Register Field Descriptions

Bits	Field	Value	Description
31-0	GPIO31-GPIO0		Each GPIO port A pin (GPIO0-GPIO31) corresponds to one bit in this register as shown in Figure 68.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to toggle from its current state. If the pin is configured as a GPIO output then it will be driven in the opposite direction of its current state. If the pin is not configured as a GPIO output then the latch is toggled but the pin is not driven.

Figure 69. GPIO Port B Set, Clear and Toggle (GPBSET, GPBCLEAR, GPBTOGGLE) Registers

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

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 74. GPIO Port B Set (GPBSET) Register Field Descriptions

Bits	Field	Value	Description
31-0	GPIO63-GPIO32		Each GPIO port B pin (GPIO32-GPIO63) corresponds to one bit in this register as shown in Figure 69.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to high. If the pin is configured as a GPIO output then it will be driven high. If the pin is not configured as a GPIO output then the latch is set but the pin is not driven.

Table 75. GPIO Port B Clear (GPBCLEAR) Register Field Descriptions

Bits	Field	Value	Description
31-0	GPIO63-GPIO32		Each GPIO port B pin (GPIO32-GPIO63) corresponds to one bit in this register as shown in Figure 69.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to low. If the pin is configured as a GPIO output then it will be driven low. If the pin is not configured as a GPIO output then the latch is cleared but the pin is not driven.

Table 76. GPIO Port B Toggle (GPBTOGGLE) Register Field Descriptions

Bits	Field	Value	Description
31-0	GPIO63-GPIO32		Each GPIO port B pin (GPIO32-GPIO63) corresponds to one bit in this register as shown in Figure 69.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to toggle from its current state. If the pin is configured as a GPIO output then it will be driven in the opposite direction of its current state. If the pin is not configured as a GPIO output then the latch is cleared but the pin is not driven.

Figure 70. GPIO Port C Set, Clear and Toggle (GPCSET, GPCCLEAR, GPCTOGGLE) Registers

31							24
Reserved							
R-0							
23	22	21	20	19	18	17	16
GPIO87	GPIO86	GPIO85	GPIO84	GPIO83	GPIO82	GPIO81	GPIO80
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
GPIO79	GPIO78	GPIO77	GPIO76	GPIO75	GPIO74	GPIO73	GPIO72
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
GPIO71	GPIO70	GPIO69	GPIO68	GPIO67	GPIO66	GPIO65	GPIO64
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 77. GPIO Port C Set (GPCSET) Register Field Descriptions

Bits	Field	Value	Description
31-24	Reserved		Reserved
23-0	GPIO87-GPIO64		Each GPIO port C pin (GPIO64-GPIO87) corresponds to one bit in this register as shown in Figure 70.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to high. If the pin is configured as a GPIO output then it will be driven high. If the pin is not configured as a GPIO output then the latch is set but the pin is not driven.

Table 78. GPIO Port C Clear (GPCCLEAR) Register Field Descriptions

Bits	Field	Value	Description
31-24	Reserved		Reserved
23-0	GPIO87-GPIO64		Each GPIO port C pin (GPIO64-GPIO87) corresponds to one bit in this register as shown in Figure 70.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to low. If the pin is configured as a GPIO output then it will be driven low. If the pin is not configured as a GPIO output then the latch is cleared but the pin is not driven.

Table 79. GPIO Port C Toggle (GPCTOGGLE) Register Field Descriptions

Bits	Field	Value	Description
31-24	Reserved		Reserved
23-0	GPIO87-GPIO64		Each GPIO port C pin (GPIO64-GPIO87) corresponds to one bit in this register as shown in Figure 70.
		0	Writes of 0 are ignored. This register always reads back a 0.
		1	Writing a 1 forces the respective output data latch to toggle from its current state. If the pin is configured as a GPIO output then it will be driven in the opposite direction of its current state. If the pin is not configured as a GPIO output then the latch is cleared but the pin is not driven.

Figure 71. GPIO XINTn, XNMI Interrupt Select (GPIOXINTnSEL, GPIOXNMISEL) Registers

15	5	4	0
Reserved		GPIOXINTnSEL	
R-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 80. GPIO XINTn Interrupt Select (GPIOXINTnSEL)[1] Register Field Descriptions

Bits	Field	Value	Description [2]
15-5	Reserved		Reserved
4-0	GPIOXINTnSEL		Select the port A GPIO signal (GPIO0 - GPIO31) that will be used as the XINT1 or XINT2 interrupt source. In addition, you can configure the interrupt in the XINT1CR or XINT2CR registers described in Section 8.6. To use XINT2 as ADC start of conversion, enable it in the ADCTRL2 register. The $\overline{\text{ADCSOC}}$ signal is always rising edge sensitive.
		00000	Select the GPIO0 pin as the XINTn interrupt source (default)
		00001	Select the GPIO1 pin as the XINTn interrupt source
		. . .	. . .
		11110	Select the GPIO30 pin as the XINTn interrupt source
		11111	Select the GPIO31 pin as the XINTn interrupt source

(1) n = 1 or 2

(2) This register is EALLOW protected. See Section 7.2 for more information.

Table 81. XINT1/XINT2 Interrupt Select and Configuration Registers

n	Interrupt	Interrupt Select Register	Configuration Register
1	XINT1	GPIOXINT1SEL	XINT1CR
2	XINT2	GPIOXINT2SEL	XINT2CR

Table 82. GPIO XINT3 - XINT7 Interrupt Select (GPIOXINTnSEL) Register Field Descriptions[1]

Bits	Field	Value	Description [2]
15-5	Reserved		Reserved
4-0	GPIOXINTnSEL		Select the port B GPIO signal (GPIO32 - GPIO63) that will be used as the XINTn interrupt source. In addition, you can configure the interrupt in the XINTnCR register described in Section 8.6.
		00000	Select the GPIO32 pin as the XINTn interrupt source (default)
		00001	Select the GPIO33 pin as the XINTn interrupt source
		. . .	. . .
		11110	Select the GPIO62 pin as the XINTn interrupt source
		11111	Select the GPIO63 pin as the XINTn interrupt source

(1) n = 3, 4, 5, 6, or 7

(2) This register is EALLOW protected. See Section 7.2 for more information.

Table 83. XINT3 - XINT7 Interrupt Select and Configuration Registers

n	Interrupt	Interrupt Select Register	Configuration Register
3	XINT3	GPIOXINT3SEL	XINT3CR
4	XINT4	GPIOXINT4SEL	XINT4CR
5	XINT5	GPIOXINT5SEL	XINT5CR
6	XINT6	GPIOXINT6SEL	XINT6CR
7	XINT7	GPIOXINT7SEL	XINT7CR

Table 84. GPIO XNMI Interrupt Select (GPIOXNMISEL) Register Field Descriptions

Bits	Field	Value	Description [1]
15-5	Reserved		Reserved
4-0	GPIOSEL		Select which port A GPIO signal (GPIO0 - GPIO31) will be used as the XNMI interrupt source. In addition you can configure the interrupt in the XNMICR register described in Section 8.6.
		00000	Select the GPIO0 pin as the XNMI interrupt source (default)
		00001	Select the GPIO1 pin as the XNMI interrupt source
		. . .	. . .
		11110	Select the GPIO30 pin as the XNMI interrupt source
		11111	Select the GPIO31 pin as the XNMI interrupt source

(1) This register is EALLOW protected. See Section 7.2 for more information.

Figure 72. GPIO Low Power Mode Wakeup Select (GPIOLPMSEL) Register

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

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 85. GPIO Low Power Mode Wakeup Select (GPIOLPMSEL) Register Field Descriptions

Bits	Field	Value	Description [1]
31-0	GPIO31 - GPIO0		Low Power Mode Wakeup Selection. Each bit in this register corresponds to one GPIO port A pin (GPIO0 - GPIO31) as shown in Figure 72.
		0	If the bit is cleared, the signal on the corresponding pin will have no effect on the HALT and STANDBY low power modes.
		1	If the respective bit is set to 1, the signal on the corresponding pin is able to wake the device from both HALT and STANDBY low power modes.

(1) This register is EALLOW protected. See Section 7.2 for more information.

7 Peripheral Frames

This chapter describes the peripheral frames. It also describes the device emulation registers.

7.1 Peripheral Frame Registers

The 2833x, 2823x devices contain four peripheral register spaces. The spaces are categorized as follows:

- Peripheral Frame 0: These are peripherals that are mapped directly to the CPU memory bus. See Table 86.
- Peripheral Frame 1: These are peripherals that are mapped to the 32-bit peripheral bus. See Table 87.
- Peripheral Frame 2: These are peripherals that are mapped to the 16-bit peripheral bus. See Table 88.
- Peripheral Frame 3: McBSP registers are mapped to this. See Table 89.

Table 86. Peripheral Frame 0 Registers(1)

NAME	ADDRESS RANGE	SIZE (×16)	ACCESS TYPE (2)
Device Emulation Registers	0x00 0880 - 0x00 09FF	384	EALLOW protected
FLASH Registers(3)	0x00 0A80 - 0x00 0ADF	96	EALLOW protected
Code Security Module Registers	0x00 0AE0 - 0x00 0AEF	16	EALLOW protected
ADC registers (dual-mapped) (0 wait, read only)	0x00 0B00 - 0x00 0B1F	32	Not EALLOW protected
XINTF Registers	0x00 0B20 - 0x00 0B3F	32	Not EALLOW protected
CPU–TIMER0/1/2 Registers	0x00 0C00 - 0x00 0C3F	64	Not EALLOW protected
PIE Registers	0x00 0CE0 - 0x00 0CFF	32	Not EALLOW protected
PIE Vector Table	0x00 0D00 - 0x00 0DFF	256	EALLOW protected
DMA Registers	0x00 1000 - 0x00 11FF	512	EALLOW protected

(1) Registers in Frame 0 support 16-bit and 32-bit accesses.
(2) If registers are EALLOW protected, then writes cannot be performed until the EALLOW instruction is executed. The EDIS instruction disables writes to prevent stray code or pointers from corrupting register contents.
(3) The Flash Registers are also protected by the Code Security Module (CSM).

Table 87. Peripheral Frame 1 Registers

Name	Address Range	Size (x16)	Access Type (1)
eCANA Registers	0x6000 - 0x60FF	256	Some eCAN control registers (and selected bits in other eCAN control registers) are EALLOW-protected. eCAN control registers require 32-bit access. See *TMS320x2833x, 2823x DSP Enhanced Controller Area Network (eCAN) User's Guide* (SPRUEU1) for more information.
eCANB Registers	0x6200 - 0x62FF	256	Some eCAN control registers (and selected bits in other eCAN control registers) are EALLOW-protected. eCAN control registers require 32-bit access. See *TMS320x2833x, 2823x DSP Enhanced Controller Area Network (eCAN) User's Guide* (SPRUEU1) for more information.
eCANB Mailbox RAM	0x6300 - 0x63FF	256	Not EALLOW-protected
ePWM1 Registers	0x6800 - 0x683F	64	Some ePWM registers are EALLOW-protected. See Section 7.2.
ePWM2 Registers	0x6840 - 0x687F	64	
ePWM3 Registers	0x6880 - 0x68BF	64	
ePWM4 Registers	0x68C0 - 0x68FF	64	
ePWM5 Registers	0x6900 - 0x693F	64	
ePWM6 Registers	0x6940 - 0x697F	64	

(1) Peripheral Frame 1 allows 16-bit and 32-bit accesses. All 32-bit accesses are aligned to even address boundaries.

Table 87. Peripheral Frame 1 Registers (continued)

Name	Address Range	Size (x16)	Access Type (1)
eCAP1 Registers	0x6A00 - 0x6A1F	32	Not EALLOW-protected
eCAP2 Registers	0x6A20 - 0x6A3F	32	
eCAP3 Registers	0x6A40 - 0x6A5F	32	
eCAP4 Registers	0x6A60 - 0x6A7F	32	
eCAP5 Registers	0x6A80 - 0x6A9F	32	
eCAP6 Registers	0x6AA0 - 0x6ABF	32	
eQEP1 Registers	0x6B00 - 0x6B3F	64	
eQEP2 Registers	0x6B40 - 0x6B7F	64	
GPIO Control Registers	0x6F80 - 0x6FBF	128	EALLOW-protected
GPIO Data Registers	0x6FC0 - 0x6FDF	32	Not EALLOW-protected
GPIO Interrupt and LPM Select Registers	0x6FE0 - 0x6FFF	32	EALLOW-protected

Table 88. Peripheral Frame 2 Registers

Name	Address Range	Size (x16)	Access Type (1)
System Control Registers	0x7010 - 0x702F	32	EALLOW-protected
SPI-A Registers	0x7040 - 0x704F	16	Not EALLOW-protected
SCI-A Registers	0x7050 - 0x705F	16	Not EALLOW-protected
External Interrupt Registers	0x7070 - 0x707F	32	Not EALLOW-protected
ADC Registers	0x7100 - 0x711F	32	Not EALLOW-protected
SCI-B Registers	0x7750 - 0x775F	16	Not EALLOW-protected
SCI-C Registers	0x7770 - 0x777F	16	Not EALLOW-protected
I2C Registers	0x7900 - 0x793F	64	Not EALLOW-protected

(1) Peripheral Frame 2 only allows 16-bit accesses. All 32-bit accesses are ignored (invalid data can be returned or written).

Table 89. Peripheral Frame 3 Registers

NAME	ADDRESS RANGE	SIZE (×16)	Access Type
McBSP-A Registers	0x5000 - 0x503F	64	Not EALLOW-protected
McBSP-B Registers	0x5040 - 0x507F	64	Not EALLOW-protected
EPWM1 + HRPWM1 (DMA) (1)	0x5800 - 0x583F	64	Some registers are EALLOW-protected. Refer to the device datasheet for details.
EPWM2 + HRPWM2 (DMA)	0x5840 - 0x587F	64	
EPWM3 + HRPWM3 (DMA)	0x5880 - 0x58BF	64	
EPWM4 + HRPWM4 (DMA)	0x58C0 - 0x58FF	64	
EPWM5 + HRPWM5 (DMA)	0x5900 - 0x593F	64	
EPWM6 + HRPWM6 (DMA)	0x5940 - 0x597F	64	

(1) The ePWM/HRPWM modules can be re-mapped to Peripheral Frame 3 where they can be accessed by the DMA module. To achieve this, bit 0 (MAPEPWM) of MAPCNF register (address 0x702E) must be set to 1. This register is EALLOW protected. When this bit is 0, the ePWM/HRPWM modules are mapped to Peripheral Frame 1.

Figure 73. MAPCNF Register (0x702E)

31	1	0
Reserved		MAPEPWM
		R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

7.2 EALLOW-Protected Registers

Several control registers are protected from spurious CPU writes by the EALLOW protection mechanism. The EALLOW bit in status register 1 (ST1) indicates if the state of protection as shown in Table 90.

Table 90. Access to EALLOW-Protected Registers

EALLOW Bit	CPU Writes	CPU Reads	JTAG Writes	JTAG Reads
0	Ignored	Allowed	Allowed[1]	Allowed
1	Allowed	Allowed	Allowed	Allowed

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

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

The following registers are EALLOW-protected:

- Device Emulation Registers
- Flash Registers
- CSM Registers
- PIE Vector Table
- System Control Registers
- GPIO MUX Registers
- Certain eCAN Registers
- XINTF Registers

Table 91. EALLOW-Protected Device Emulation Registers

Name	Address	Size (x16)	Description
DEVICECNF	0x0880 0x0881	2	Device Configuration Register
PROTSTART	0x0884	1	Block Protection Start Address Register
PROTRANGE	0x0885	1	Block Protection Range Address Register

Table 92. EALLOW-Protected Flash/OTP Configuration Registers

Name	Address	Size (x16)	Description
FOPT	0x0A80	1	Flash Option Register
FPWR	0x0A82	1	Flash Power Modes Register
FSTATUS	0x0A83	1	Status Register
FSTDBYWAIT	0x0A84	1	Flash Sleep To Standby Wait State Register
FACTIVEWAIT	0x0A85	1	Flash Standby To Active Wait State Register
FBANKWAIT	0x0A86	1	Flash Read Access Wait State Register
FOTPWAIT	0x0A87	1	OTP Read Access Wait State Register

Table 93. EALLOW-Protected Code Security Module (CSM) Registers

Register Name	Address	Size (x16)	Register Description
KEY0	0x0AE0	1	Low word of the 128-bit KEY register
KEY1	0x0AE1	1	Second word of the 128-bit KEY register
KEY2	0x0AE2	1	Third word of the 128-bit KEY register

Table 93. EALLOW-Protected Code Security Module (CSM) Registers (continued)

Register Name	Address	Size (x16)	Register Description
KEY3	0x0AE3	1	Fourth word of the 128-bit KEY register
KEY4	0x0AE4	1	Fifth word of the 128-bit KEY register
KEY5	0x0AE5	1	Sixth word of the 128-bit KEY register
KEY6	0x0AE6	1	Seventh word of the 128-bit KEY register
KEY7	0x0AE7	1	High word of the 128-bit KEY register
CSMSCR	0x0AEF	1	CSM status and control register

Table 94. EALLOW-Protected PIE Vector Table

Name	Address	Size (x16)	Description
Not used	0x0D00	2	Reserved
	0x0D02		
	0x0D04		
	0x0D06		
	0x0D08		
	0x0D0A		
	0x0D0C		
	0x0D0E		
	0x0D10		
	0x0D12		
	0x0D14		
	0x0D16		
	0x0D18		
INT13	0x0D1A	2	External Interrupt 13 (XINT13) or CPU-Timer 1 (for RTOS use)
INT14	0x0D1C	2	CPU-Timer 2 (for RTOS use)
DATALOG	0x0D1E	2	CPU Data Logging Interrupt
RTOSINT	0x0D20	2	CPU Real-Time OS Interrupt
EMUINT	0x0D22	2	CPU Emulation Interrupt
NMI	0x0D24	2	External Non-Maskable Interrupt
ILLEGAL	0x0D26	2	Illegal Operation
USER1	0x0D28	2	User-Defined Trap
. .		.	.
USER12	0x0D3E	2	User-Defined Trap
INT1.1	0x0D40	2	Group 1 Interrupt Vectors
.	.	.	
INT1.8	0x0D4E	2	
.	.	.	Group 2 Interrupt Vectors
.	.	.	to Group 11 Interrupt Vectors
.	.	.	
INT12.1	0x0DF0	2	Group 12 Interrupt Vectors
.	.	.	
INT12.8	0x0DFE	2	

Table 95. EALLOW-Protected PLL, Clocking, Watchdog, and Low-Power Mode Registers

Name	Address	Size (x16)	Description
PLLSTS	0x7011	1	PLL Status Register
HISPCP	0x701A	1	High-Speed Peripheral Clock Prescaler Register for HSPCLK Clock
LOSPCP	0x701B	1	Low-Speed Peripheral Clock Prescaler Register for HSPCLK Clock
PCLKCR0	0x701C	1	Peripheral Clock Control Register 0
PCLKCR1	0x701D	1	Peripheral Clock Control Register 1
LPMCR0	0x701E	1	Low Power Mode Control Register 0
PCLKCR3	0x7020	1	Peripheral Clock Control Register 3
PLLCR	0x7021	1	PLL Control Register
SCSR	0x7022	1	System Control and Status Register
WDCNTR	0x7023	1	Watchdog Counter Register
WDKEY	0x7025	1	Watchdog Reset Key Register
WDCR	0x7029	1	Watchdog Control Register

Table 96. EALLOW-Protected GPIO MUX Registers

Name	Address	Size (x16)	Description
GPACTRL	0x6F80	2	GPIO A Control Register (GPIO0 to GPIO31)
GPAQSEL1	0x6F82	2	GPIO A Qualifier Select 1 Register (GPIO0 to GPIO15)
GPAQSEL2	0x6F84	2	GPIO A Qualifier Select 2 Register (GPIO16 to GPIO31)
GPAMUX1	0x6F86	2	GPIO A Mux 1 Register (GPIO0 to GPIO15)
GPAMUX2	0x6F88	2	GPIO A Mux 2 Register (GPIO16 to GPIO31)
GPADIR	0x6F8A	2	GPIO A Direction Register (GPIO0 to GPIO31)
GPAPUD	0x6F8C	2	GPIO A Pull Up Disable Register (GPIO0 to GPIO31)
GPBCTRL	0x6F90	2	GPIO B Control Register (GPIO32 to GPIO35)
GPBQSEL1	0x6F92	2	GPIO B Qualifier Select 1 Register (GPIO32 to GPIO35)
GPBQSEL2	0x6F94	2	Reserved
GPBMUX1	0x6F96	2	GPIO B Mux 1 Register (GPIO32 to GPIO35)
GPBMUX2	0x6F98	2	Reserved
GPBDIR	0x6F9A	2	GPIO B Direction Register (GPIO32 to GPIO35)
GPBPUD	0x6F9C	2	GPIO B Pull Up Disable Register (GPIO32 to GPIO35)
GPCMUX1	0x6FA6	2	GPIO C Mux 1 Register (GPIO64 to 79)
GPCMUX2	0x6FA8	2	GPIO C Mux 2 Register (GPIO80 to 87)
GPCDIR	0x6FAA	2	GPIO C Direction Register (GPIO64 to 87)
GPCPUD	0x6FAC	2	GPIO C Pull Up Disable Register (GPIO64 to 87)
GPIOXINT1SEL	0x6FE0	1	XINT1 GPIO Input Select Register (GPIO0 to GPIO31)
GPIOXINT2SEL	0x6FE1	1	XINT2 GPIO Input Select Register (GPIO0 to GPIO31)
GPIOXNMISEL	0x6FE2	1	XNMI GPIO Input Select Register (GPIO0 to GPIO31)
GPIOXINT3SEL	0x6FE3	1	XINT3 GPIO Input Select Register (GPIO32 to GPIO63)
GPIOXINT4SEL	0x6FE4	1	XINT4 GPIO Input Select Register (GPIO32 to GPIO63)
GPIOXINT5SEL	0x6FE5	1	XINT5 GPIO Input Select Register (GPIO32 to GPIO63)
GPIOXINT6SEL	0x6FE6	1	XINT6 GPIO Input Select Register (GPIO32 to GPIO63)
GPIOXINT7SEL	0x6FE7	1	XINT7 GPIO Input Select Register (GPIO32 to GPIO63)
GPIOLPMSEL	0x6FE8	2	LPM GPIO Select Register (GPIO0 to GPIO31)

Table 97. EALLOW-Protected eCAN Registers

Name	eCAN-A Address	eCAN-B Address	Size (x16)	Description
CANMC	0x6014	0x6214	2	Master Control Register(1)
CANBTC	0x6016	0x6216	2	Bit Timing Configuration Register(2)
CANGIM	0x6020	0x6220	2	Global Interrupt Mask Register(3)
CANMIM	0x6024	0x6224	2	Mailbox Interrupt Mask Register
CANTSC	0x602E	0x622E	2	Time Stamp Counter
CANTIOC	0x602A	0x622A	1	I/O Control Register for CANTXA Pin(4)
CANRIOC	0x602C	0x622C	1	I/O Control Register for CANRXA Pin(5)

(1) Only bits CANMC[15-9] and [7-6] are protected
(2) Only bits BCR[23-16] and [10-0] are protected
(3) Only bits CANGIM[17-16] , [14-8], and [2-0] are protected
(4) Only IOCONT1[3] is protected
(5) Only IOCONT2[3] is protected

Table 98 shows addresses for the following ePWM EALLOW-protected registers:

- Trip Zone Select Register (TZSEL)
- Trip Zone Control Register (TZCTL)
- Trip Zone Enable Interrupt Register (TZEINT)
- Trip Zone Clear Register (TZCLR)
- Trip Zone Force Register (TZFRC)
- HRPWM Configuration Register (HRCNFG)

Table 98. EALLOW-Protected ePWM1 - ePWM6 Registers

	TZSEL	TZCTL	TZEINT	TZCLR	TZFRC	HRCNFG	Size x16
ePWM1	0x6812	0x6814	0x6815	0x6817	0x6818	0x6820	1
ePWM2	0x6852	0x6854	0x6855	0x6857	0x6858	0x6860	1
ePWM3	0x6892	0x6894	0x6895	0x6897	0x6898	0x68A0	1
ePWM4	0x68D2	0x68D4	0x68D5	0x68D7	0x68D8	0x68E0	1
ePWM5	0x6912	0x6914	0x6915	0x6917	0x6918	0x6920	1
ePWM6	0x6952	0x6954	0x6955	0x6957	0x6958	0x6960	1

Table 99. XINTF Registers

Name	Address	Size (x16)	Description (1)
XTIMING0	0x0000−0B20	2	XINTF Timing Register, Zone 0
XTIMING6(2)	0x0000−0B2C	2	XINTF Timing Register, Zone 6
XTIMING7	0x0000−0B2E	2	XINTF Timing Register, Zone 7
XINTCNF2(3)	0x0000−0B34	2	XINTF Configuration Register
XBANK	0x0000−0B38	1	XINTF Bank Control Register
XREVISION	0x0000−0B3A	1	XINTF Revision Register
XRESET	0x0000 083D	1	XINTF Reset Register

(1) All XINTF registers are EALLOW protected.
(2) XTIMING1 - XTIMING5 are reserved for future expansion and are not currently used.
(3) XINTCNF1 is reserved and not currently used.

7.3 Device Emulation Registers

These registers are used to control the protection mode of the C28x CPU and to monitor some critical device signals. The registers are defined in Table 100.

Table 100. Device Emulation Registers

Name	Address	Size (x16)	Description
DEVICECNF	0x0880 0x0881	2	Device Configuration Register
PARTID	0x380090	1	Part ID Register
CLASSID	0x0882	1	Class ID Register
REVID	0x0883	1	Revision ID Register
PROTSTART	0x0884	1	Block Protection Start Address Register
PROTRANGE	0x0885	1	Block Protection Range Address Register

Figure 74. Device Configuration (DEVICECNF) Register

31	27	26	20	19	18	16
Reserved	TRST	Reserved		ENPROT	Reserved	
R-0	R-0	R-0		R/W-1	R-111	

15	5	4	3	2	0
Reserved	XRS	Res	VMAPS	Reserved	
R-0	R-P	R-0	R-1	R-011	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 101. DEVICECNF Register Field Descriptions

Bits	Field	Value	Description
31-28	Reserved		Reserved
27	TRST		Read status of $\overline{\text{TRST}}$ signal. Reading this bit gives the current satus of the $\overline{\text{TRST}}$ signal.
		0	No emulator is connected.
		1	An emulator is connected.
26:20	Reserved		
19	ENPROT		Enable Write-Read Protection Mode Bit.
		0	Disables write-read protection mode
		1	Enables write-read protection as specified by the PROTSTART and PROTRANGE registers
18-6	Reserved		Reserved
5	XRS		Reset Input Signal Status. This is connected directly to the $\overline{\text{XRS}}$ input pin.
4	Reserved		Reserved
3	VMAPS		VMAP Configure Status. This indicates the status of VMAP.
2-0	Reserved		Reserved

Figure 75. Part ID Register

15	8	7	0
PARTTYPE		PARTNO	
R		R	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 102. PARTID Register Field Descriptions

Bit	Field	Value [1]	Description
15:8	PARTTYPE		These 8 bits specify the type of device such as flash-based.
		0x00	Flash-based device
			All other values are reserved.
7:0	PARTNO		These 8 bits specify the feature set of the device as follows:
		0x00EF	F28335
		0x00EE	F28334
		0x00ED	F28332
		0x00E8	F28235
		0x00E7	F28234
		0x00E6	F28232

(1) The reset value depends on the device as indicated in the register description.

Figure 76. CLASSID Register

15	8	7	0
PARTTYPE		CLASSNO	
R		R	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 103. CLASSID Register Description

Bits	Field	Value	Description
15-8	PARTTYPE		These 8 bits specify the type of device such as flash-based.
		0x00	Flash-based device
			All other values are reserved.
7-0	CLASSNO		These 16 bits specify the class for the particular part.
		0x00EF	F28335, F28334, F28332 TMS320F2833x Floating Point Class device
		0x00E8	F28235, F28234, F28232 TMS320F2823x Fixed Point Class device

Figure 77. REVID Register

15	0
REVID	
R	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 104. REVID Register Field Descriptions

Bits	Field	Value	Description
15-0	REVID		[1]These 16 bits specify the silicon revision number for the particular part. This number always starts with 0x0000 on the first revision of the silicon and is incremented on any subsequent revisions.
		0x0000	Silicon Revision 0 - TMX
		0x0001	Silicon Revision 1 - TMS

(1) The reset value depends on the silicon revision as described in the register field description.

7.4 Write-Followed-by-Read Protection

The PROTSTART and PROTRANGE registers set the memory address range for which CPU "write" followed by "read" operations are protected (operations occur in sequence rather then in their natural pipeline order). This is necessary protection for certain peripheral operations.

Example: The following lines of code perform a write to register 1 (REG1) location and then the next instruction performs a read from Register 2 (REG2) location. On the processor memory bus, with block protection disabled, the read operation is issued before the write as shown.

```
MOV @REG1,AL    ---------+ TBIT
@REG2,#BIT_X    ---------|-------> Read
                         +-------> Write
```

If block protection is enabled, then the read is stalled until the write occurs as shown:

```
MOV @REG1,AL    ---------+ TBIT
@REG2,#BIT_X    ---------|-----+
                         +-----|---> Write
                               +---> Read
```

Table 105. PROTSTART and PROTRANGE Registers

Name	Address	Size	Type	Reset	Description
PROTSTART	0x0884	16	R/W	0x0100[1]	The PROTSTART register sets the starting address relative to the 16 most significant bits of the processors lower 22-bit address reach. Hence, the smallest resolution is 64 words.
PROTRANGE	0x0885	16	R/W	0x00FF[1]	The PROTRANGE register sets the block size (from the starting address), starting with 64 words and incrementing by binary multiples (64, 128, 256, 512, 1K, 2K, 4K, 8K, 16K, ..., 2M).

(1) The default values of these registers on reset are selected to cover the Peripheral Frame 1, Peripheral Frame 2, Peripheral Frame 3, and XINTF Zone 1 areas of the memory map (address range 0x4000 to 0x8000).

Table 106. PROTSTART Valid Values

		Register Bits (1)															
Start Address	Register Value	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00 0000	0x0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00 0040	0x0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x00 0080	0x0002	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x00 00C0	0x0003	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
0x3F FF00	0xFFFC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
0x3F FF40	0xFFFD	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0x3F FF80	0xFFFE	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0x3F FFC0	0xFFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

(1) The quickest way to calculate register value is to divide the desired block starting address by 64.

Table 107. PROTRANGE Valid Values

		Register Bits (1)															
Block Size	Register Value	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
64	0x0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
128	0x0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
256	0x0003	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
512	0x0007	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
1K	0x000F	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

(1) Not all register values are valid. The PROTSTART address value must be a multiple of the range value. For example: if the block size is set to 4K, then the start address can only be at any 4K boundary.

Table 107. PROTRANGE Valid Values (continued)

Block Size	Register Value	Register Bits (1)															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
256K	0x0FFF	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
512K	0x1FFF	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
1M	0x3FFF	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2M	0x7FFF	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4M	0xFFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

8 Peripheral Interrupt Expansion (PIE)

The peripheral interrupt expansion (PIE) block multiplexes numerous interrupt sources into a smaller set of interrupt inputs. The PIE block can support 96 individual interrupts that are grouped into blocks of eight. Each group is fed into one of 12 core interrupt lines (INT1 to INT12). Each of the 96 interrupts is supported by its own vector stored in a dedicated RAM block that you can modify. The CPU, upon servicing the interrupt, automatically fetches the appropriate interrupt vector. It takes nine CPU clock cycles to fetch the vector and save critical CPU registers. Therefore, the CPU can respond quickly to interrupt events. Prioritization of interrupts is controlled in hardware and software. Each individual interrupt can be enabled/disabled within the PIE block.

8.1 Overview of the PIE Controller

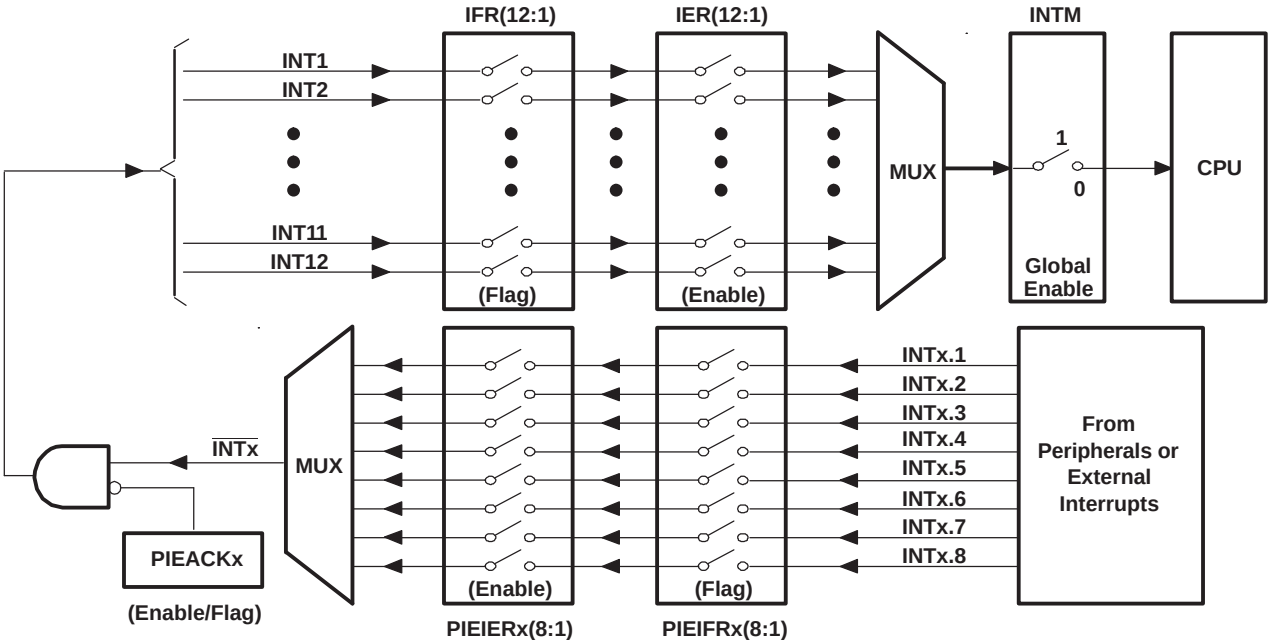
The 28x CPU supports one nonmaskable interrupt (NMI) and 16 maskable prioritized interrupt requests (INT1-INT14, RTOSINT, and DLOGINT) at the CPU level. The 28x devices have many peripherals and each peripheral is capable of generating one or more interrupts in response to many events at the peripheral level. Because the CPU does not have sufficient capacity to handle all peripheral interrupt requests at the CPU level, a centralized peripheral interrupt expansion (PIE) controller is required to arbitrate the interrupt requests from various sources such as peripherals and other external pins.

The PIE vector table is used to store the address (vector) of each interrupt service routine (ISR) within the system. There is one vector per interrupt source including all MUXed and nonMUXed interrupts. You populate the vector table during device initialization and you can update it during operation.

8.1.1 Interrupt Operation Sequence

Figure 78 shows an overview of the interrupt operation sequence for all multiplexed PIE interrupts. Interrupt sources that are not multiplexed are fed directly to the CPU.

Figure 78. Overview: Multiplexing of Interrupts Using the PIE Block



- **Peripheral Level**

  An interrupt-generating event occurs in a peripheral. The interrupt flag (IF) bit corresponding to that event is set in a register for that particular peripheral.

  If the corresponding interrupt enable (IE) bit is set, the peripheral generates an interrupt request to the PIE controller. If the interrupt is not enabled at the peripheral level, then the IF remains set until cleared by software. If the interrupt is enabled at a later time, and the interrupt flag is still set, the interrupt request is asserted to the PIE.

  Interrupt flags within the peripheral registers must be manually cleared. See the peripheral reference guide for a specific peripheral for more information.

- **PIE Level**

  The PIE block multiplexes eight peripheral and external pin interrupts into one CPU interrupt. These interrupts are divided into 12 groups: PIE group 1 - PIE group 12. The interrupts within a group are multiplexed into one CPU interrupt. For example, PIE group 1 is multiplexed into CPU interrupt 1 (INT1) while PIE group 12 is multiplexed into CPU interrupt 12 (INT12). Interrupt sources connected to the remaining CPU interrupts are not multiplexed. For the nonmultiplexed interrupts, the PIE passes the request directly to the CPU.

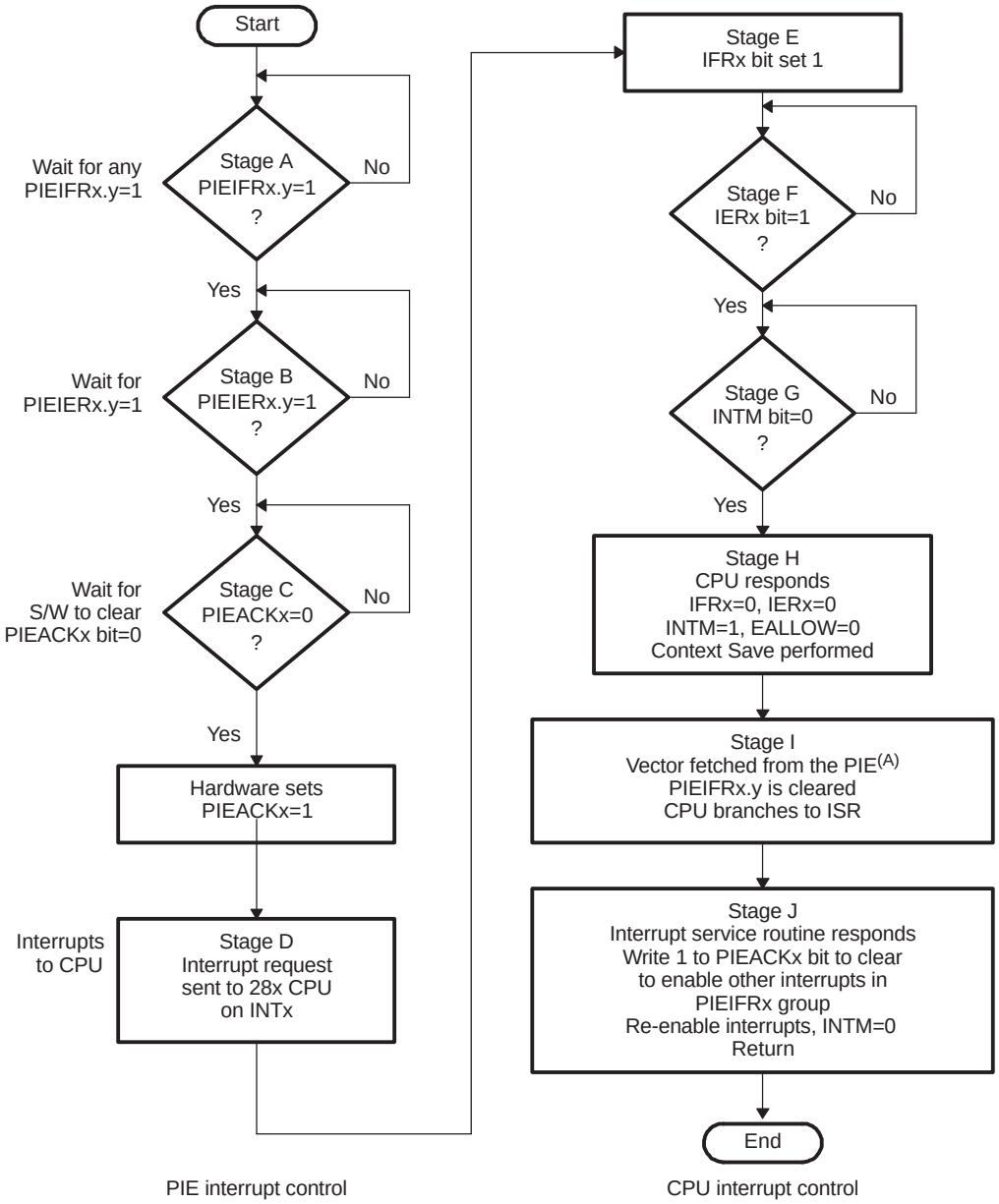
  For multiplexed interrupt sources, each interrupt group in the PIE block has an associated flag register (PIEIFRx) and enable (PIEIERx) register (x = PIE group 1 - PIE group 12). Each bit, referred to as y, corresponds to one of the 8 MUXed interrupts within the group. Thus PIEIFRx.y and PIEIERx.y correspond to interrupt y (y = 1-8) in PIE group x (x = 1-12). In addition, there is one acknowledge bit (PIEACK) for every PIE interrupt group referred to as PIEACKx (x = 1-12). Figure 79 illustrates the behavior of the PIE hardware under various PIEIFR and PIEIER register conditions.

  Once the request is made to the PIE controller, the corresponding PIE interrupt flag (PIEIFRx.y) bit is set. If the PIE interrupt enable (PIEIERx.y) bit is also set for the given interrupt then the PIE checks the corresponding PIEACKx bit to determine if the CPU is ready for an interrupt from that group. If the PIEACKx bit is clear for that group, then the PIE sends the interrupt request to the CPU. If PIEACKx is set, then the PIE waits until it is cleared to send the request for INTx. See Section 6.3 for details.

- **CPU Level**

  Once the request is sent to the CPU, the CPU level interrupt flag (IFR) bit corresponding to INTx is set. After a flag has been latched in the IFR, the corresponding interrupt is not serviced until it is appropriately enabled in the CPU interrupt enable (IER) register or the debug interrupt enable register (DBGIER) and the global interrupt mask (INTM) bit.

Figure 79. Typical PIE/CPU Interrupt Response - INTx.y



A For multiplexed interrupts, the PIE responds with the highest priority interrupt that is both flagged and enabled. If there is no interrupt both flagged and enabled, then the highest priority interrupt within the group (INTx.1 where x is the PIE group) is used. See Section Section 8.3.3 for details.

As shown in Table 108, the requirements for enabling the maskable interrupt at the CPU level depends on the interrupt handling process being used. In the standard process, which happens most of the time, the DBGIER register is not used. When the 28x is in real-time emulation mode and the CPU is halted, a different process is used. In this special case, the DBGIER is used and the INTM bit is ignored. If the DSP is in real-time mode and the CPU is running, the standard interrupt-handling process applies.

Table 108. Enabling Interrupt

Interrupt Handling Process	Interrupt Enabled If...
Standard	INTM = 0 and bit in IER is 1
DSP in real-time mode and halted	Bit in IER is 1 and DBGIER is 1

The CPU then prepares to service the interrupt. This preparation process is described in detail in *TMS320C28x DSP CPU and Instruction Set Reference Guide* (literature number SPRU430). In preparation, the corresponding CPU IFR and IER bits are cleared, EALLOW and LOOP are cleared, INTM and DBGM are set, the pipeline is flushed and the return address is stored, and the automatic context save is performed. The vector of the ISR is then fetched from the PIE module. If the interrupt request comes from a multiplexed interrupt, the PIE module uses the group PIEIERx and PIEIFRx registers to decode which interrupt needs to be serviced. This decode process is described in detail in Section Section 8.3.3.

The address for the interrupt service routine that is executed is fetched directly from the PIE interrupt vector table. There is one 32-bit vector for each of the possible 96 interrupts within the PIE. Interrupt flags within the PIE module (PIEIFRx.y) are automatically cleared when the interrupt vector is fetched. The PIE acknowledge bit for a given interrupt group, however, must be cleared manually when ready to receive more interrupts from the PIE group.

8.2 Vector Table Mapping

On 28xx devices, the interrupt vector table can be mapped to four distinct locations in memory. In practice only the PIE vector table mapping is used.

This vector mapping is controlled by the following mode bits/signals:

VMAP: VMAP is found in Status Register 1 ST1 (bit 3). A device reset sets this bit to 1. The state of this bit can be modified by writing to ST1 or by SETC/CLRC VMAP instructions. For normal operation leave this bit set.

M0M1MAP: M0M1MAP is found in Status Register 1 ST1 (bit 11). A device reset sets this bit to 1. The state of this bit can be modified by writing to ST1 or by SETC/CLRC M0M1MAP instructions. For normal 28xx device operation, this bit should remain set. M0M1MAP = 0 is reserved for TI testing only.

ENPIE: ENPIE is found in PIECTRL Register (bit 0). The default value of this bit, on reset, is set to 0 (PIE disabled). The state of this bit can be modified after reset by writing to the PIECTRL register (address 0x0000 0CE0).

Using these bits and signals the possible vector table mappings are shown in Table 109.

Table 109. Interrupt Vector Table Mapping

Vector MAPS	Vectors Fetched From	Address Range	VMAP	M0M1MAP	ENPIE
M1 Vector(1)	M1 SARAM Block	0x000000 - 0x00003F	0	0	X
M0 Vector(1)	M0 SARAM Block	0x000000 - 0x00003F	0	1	X
BROM Vector	Boot ROM Block	0x3FFFC0 - 0x3FFFFF	1	X	0
PIE Vector	PIE Block	0x000D00 - 0x000DFF	1	X	1

(1) Vector map M0 and M1 Vector is a reserved mode only. On the 28x devices these are used as SARAM.

The M1 and M0 vector table mapping are reserved for TI testing only. When using other vector mappings, the M0 and M1 memory blocks are treated as SARAM blocks and can be used freely without any restrictions.

After a device reset operation, the vector table is mapped as shown in Table 110.

Table 110. Vector Table Mapping After Reset Operation

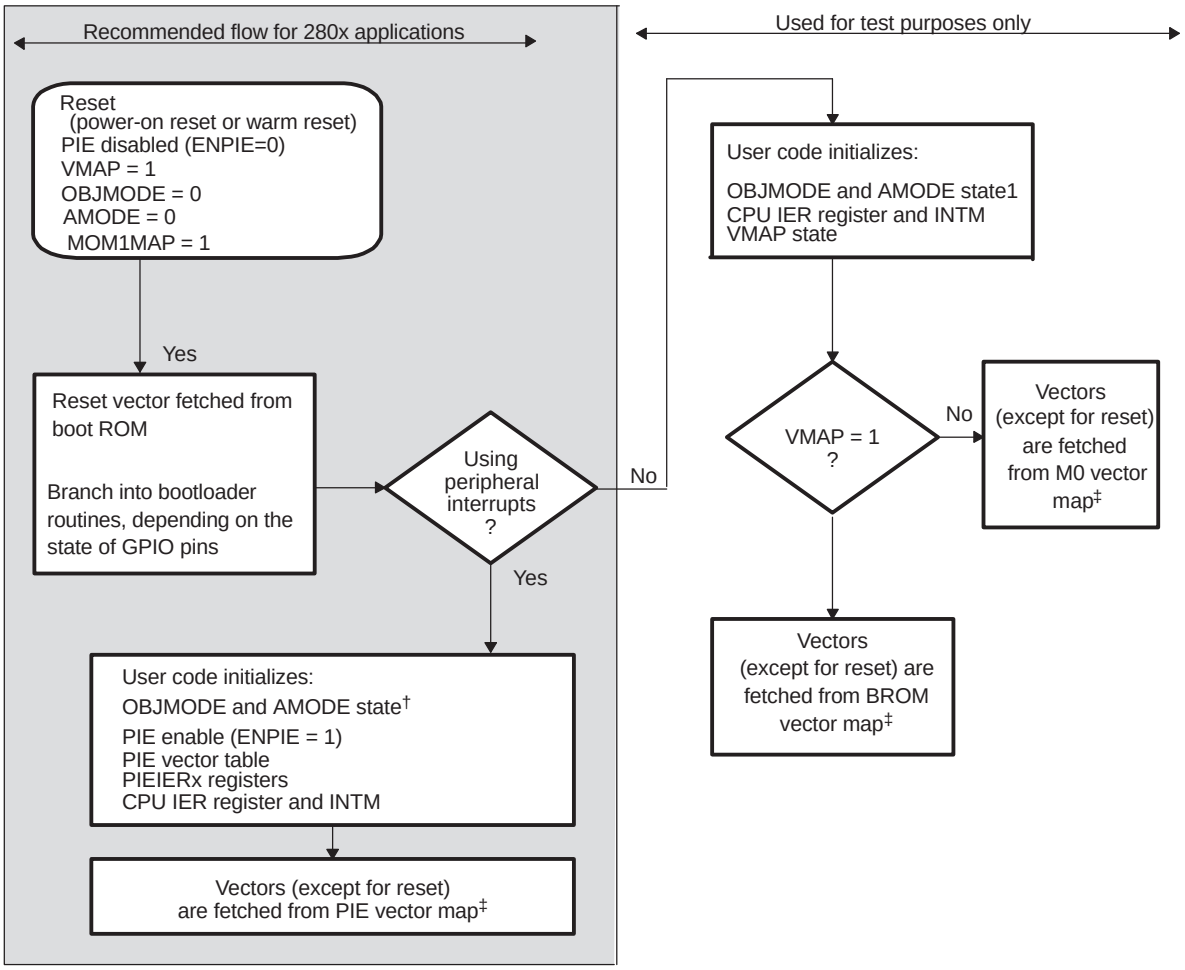
Vector MAPS	Reset Fetched From	Address Range	VMAP (1)	M0M1MAP (1)	ENPIE (1)
BROM Vector (2)	Boot ROM Block	0x3FFFC0 - 0x3FFFFF	1	1	0

(1) On the 28x devices, the VMAP and M0M1MAP modes are set to 1 on reset. The ENPIE mode is forced to 0 on reset.
(2) The reset vector is always fetched from the boot ROM.

After the reset and boot is complete, the PIE vector table should be initialized by the user's code. Then the application enables the PIE vector table. From that point on the interrupt vectors are fetched from the PIE vector table. Note: when a reset occurs, the reset vector is always fetched from the vector table as shown in Table 110. After a reset the PIE vector table is always disabled.

Figure 80 illustrates the process by which the vector table mapping is selected.

Figure 80. Reset Flow Diagram



A The compatibility operating mode of the 28x CPU is determined by a combination of the OBJMODE and AMODE bits in Status Register 1 (ST1):

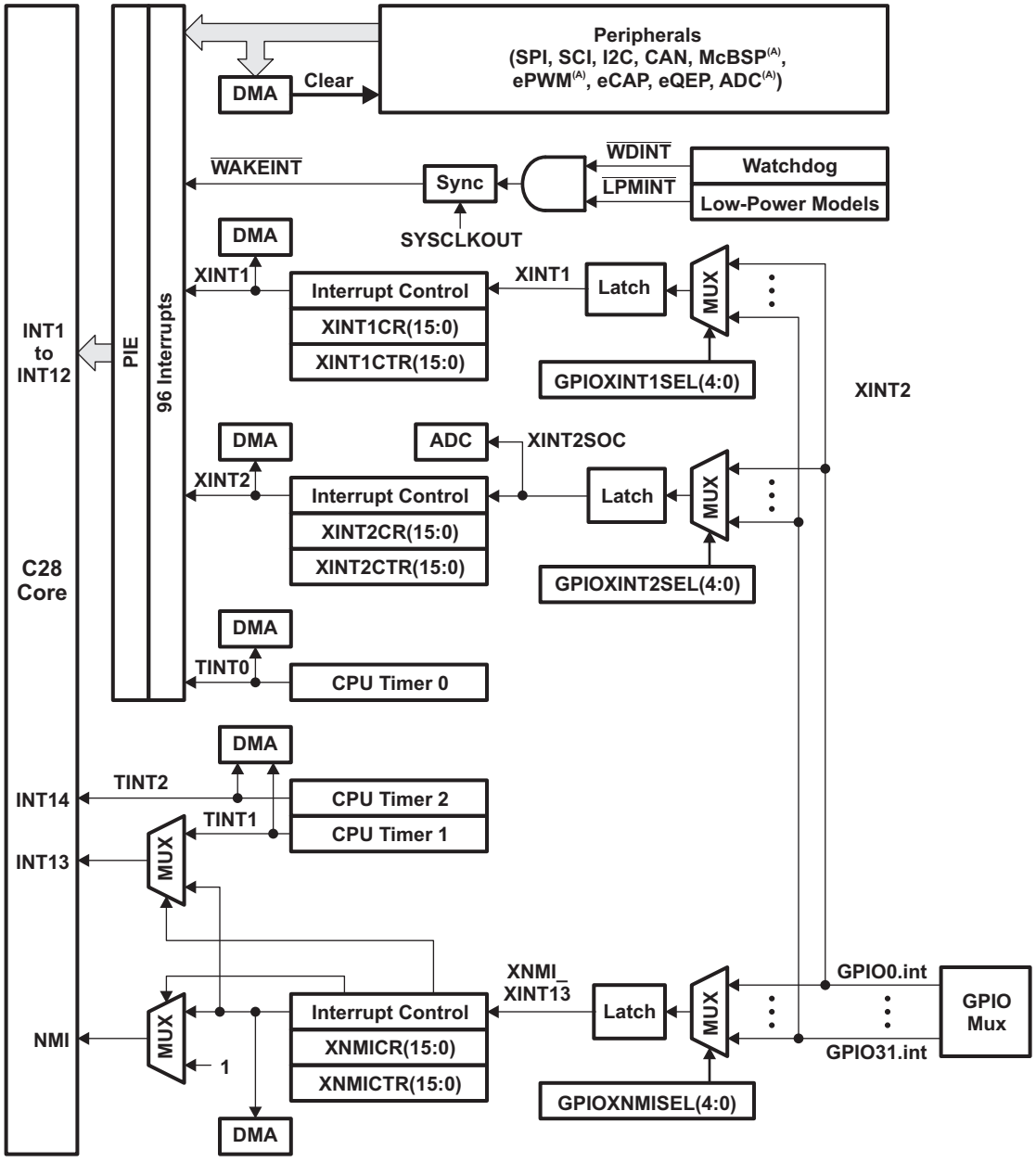
Operating Mode	OBJMODE	AMODE	
C28x Mode	1	0	
24x/240x Source-Compatible	1	1	
C27x Object-Compatible	0	0	(Default at reset)

B The reset vector is always fetched from the boot ROM.

8.3 Interrupt Sources

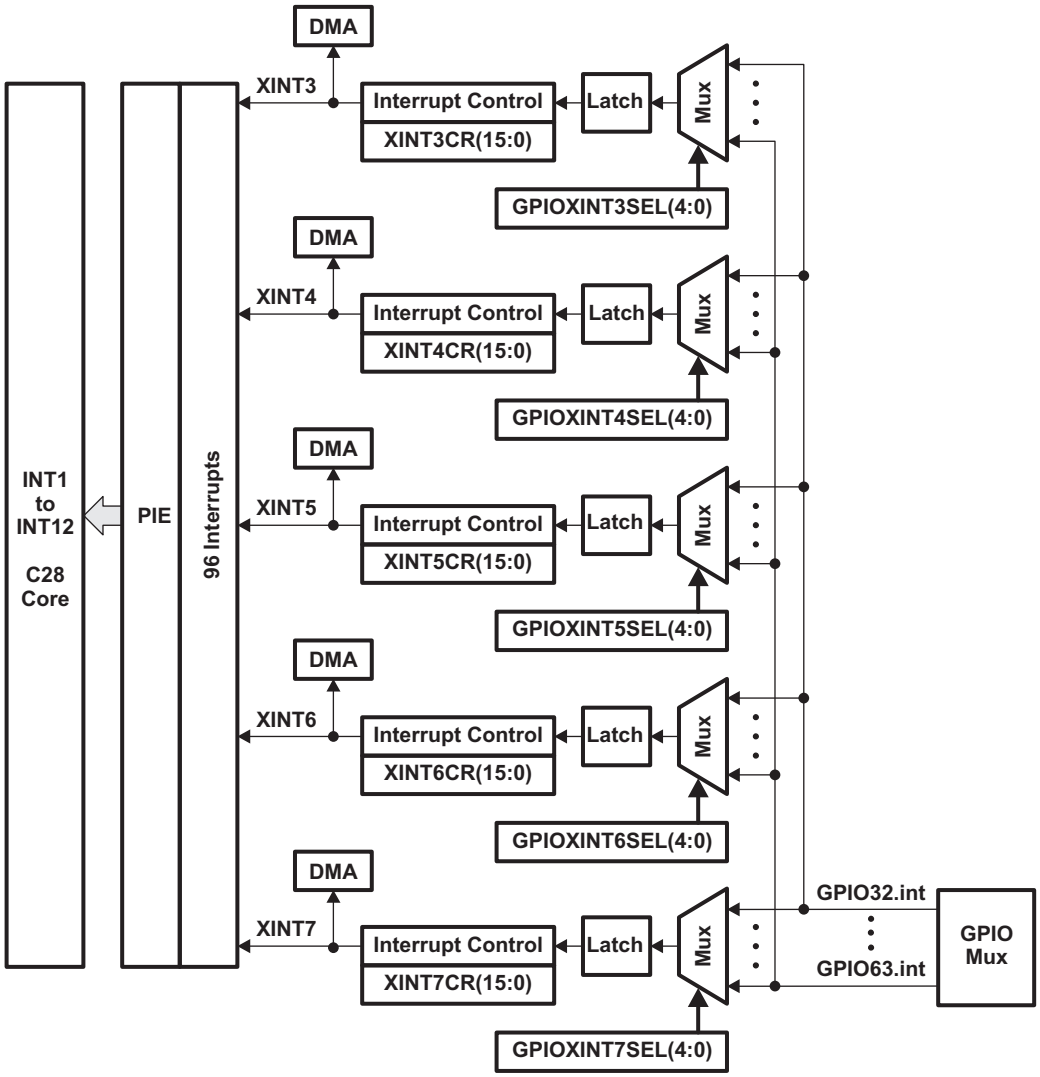
Figure 81 shows how the various interrupt sources are multiplexed within the devices. This multiplexing (MUX) scheme may not be exactly the same on all 28x devices. See the data manual of your particular device for details.

Figure 81. PIE Interrupt Sources and External Interrupts XINT1/XINT2



A DMA accessible

Figure 82. PIE Interrupt Sources and External Interrupts (XINT3 – XINT7)



8.3.1 Procedure for Handling Multiplexed Interrupts

The PIE module multiplexes eight peripheral and external pin interrupts into one CPU interrupt. These interrupts are divided into 12 groups: PIE group 1 - PIE group 12. Each group has an associated enable PIEIER and flag PIEIFR register. These registers are used to control the flow of interrupts to the CPU. The PIE module also uses the PIEIER and PIEIFR registers to decode to which interrupt service routine the CPU should branch.

There are three main rules that should be followed when clearing bits within the PIEIFR and the PIEIER registers:

Rule 1: Never clear a PIEIFR bit by software

An incoming interrupt may be lost while a write or a read-modify-write operation to the PIEIFR register takes place. To clear a PIEIFR bit, the pending interrupt must be serviced. If you want to clear the PIEIFR bit without executing the normal service routine, then use the following procedure:

1. Set the EALLOW bit to allow modification to the PIE vector table.
2. Modify the PIE vector table so that the vector for the peripheral's service routine points to a temporary ISR. This temporary ISR will only perform a return from interrupt (IRET) operation.
3. Enable the interrupt so that the interrupt will be serviced by the temporary ISR.
4. After the temporary interrupt routine is serviced, the PIEIFR bit will be clear
5. Modify the PIE vector table to re-map the peripheral's service routine to the proper service routine.
6. Clear the EALLOW bit.

Rule 2: Procedure for software-prioritizing interrupts

Use the method found in the *C2833x C/C++ Header Files and Peripheral Examples in C* (literature number SPRC530) .

(a) Use the CPU IER register as a global priority and the individual PIEIER registers for group priorities. In this case the PIEIER register is only modified within an interrupt. In addition, only the PIEIER for the same group as the interrupt being serviced is modified. This modification is done while the PIEACK bit holds additional interrupts back from the CPU.

(b) Never disable a PIEIER bit for a group when servicing an interrupt from an unrelated group.

Rule 3: Disabling interrupts using PIEIER

If the PIEIER registers are used to enable and then later disable an interrupt then the procedure described in Section 8.3.2 must be followed.

8.3.2 Procedures for Enabling And Disabling Multiplexed Peripheral Interrupts

The proper procedure for enabling or disabling an interrupt is by using the peripheral interrupt enable/disable flags. The primary purpose of the PIEIER and CPU IER registers is for software prioritization of interrupts within the same PIE interrupt group. The software package *C280x C/C++ Header Files and Peripheral Examples in C* (literature number SPRC530) includes an example that illustrates this method of software prioritizing interrupts.

Should bits within the PIEIER registers need to be cleared outside of this context, one of the following two procedures should be followed. The first method preserves the associated PIE flag register so that interrupts are not lost. The second method clears the associated PIE flag register.

Method 1: Use the PIEIERx register to disable the interrupt and preserve the associated PIEIFRx flags.

To clear bits within a PIEIERx register while preserving the associated flags in the PIEIFRx register, the following procedure should be followed:

Step a. Disable global interrupts (INTM = 1).
Step b. Clear the PIEIERx.y bit to disable the interrupt for a given peripheral. This can be done for one or more peripherals within the same group.
Step c. Wait 5 cycles. This delay is required to be sure that any interrupt that was incoming to the CPU has been flagged within the CPU IFR register.
Step d. Clear the CPU IFRx bit for the peripheral group. This is a safe operation on the CPU IFR register.
Step e. Clear the PIEACKx bit for the peripheral group.
Step f. Enable global interrupts (INTM = 0).

Method 2: Use the PIEIERx register to disable the interrupt and clear the associated PIEIFRx flags.

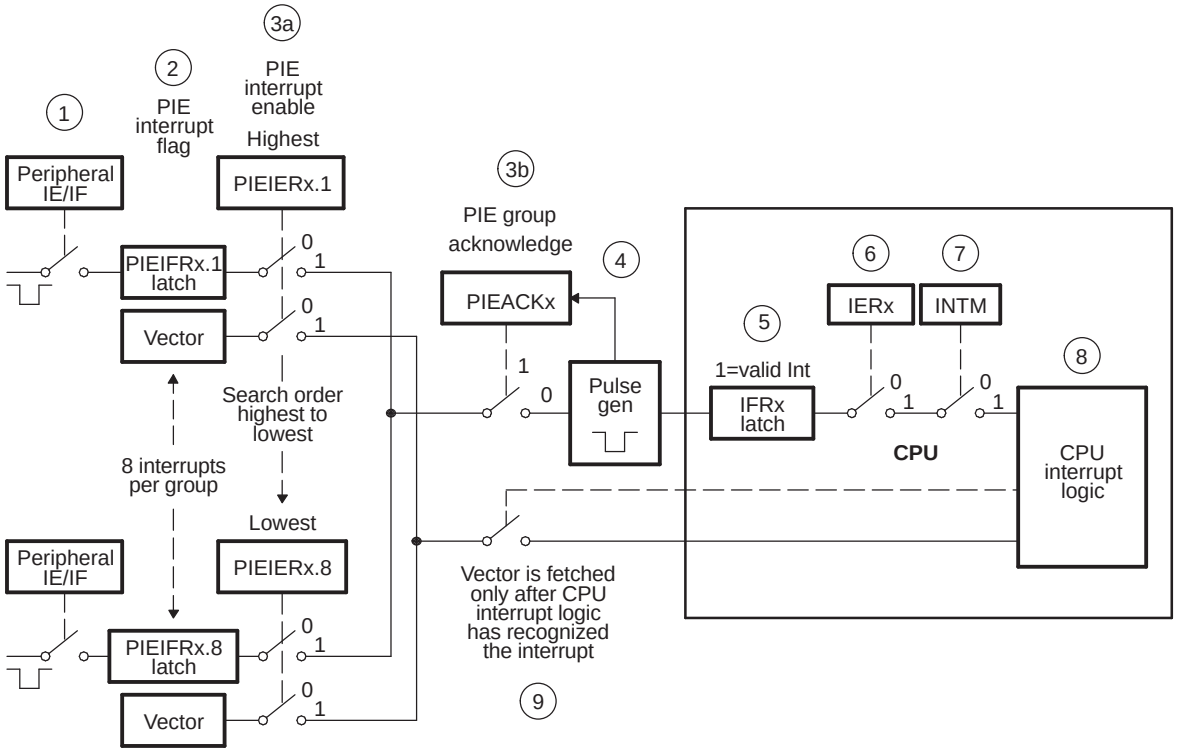
To perform a software reset of a peripheral interrupt and clear the associated flag in the PIEIFRx register and CPU IFR register, the following procedure should be followed:

Step 1. Disable global interrupts (INTM = 1).
Step 2. Set the EALLOW bit.
Step 3. Modify the PIE vector table to temporarily map the vector of the specific peripheral interrupt to a empty interrupt service routine (ISR). This empty ISR will only perform a return from interrupt (IRET) instruction. This is the safe way to clear a single PIEIFRx.y bit without losing any interrupts from other peripherals within the group.
Step 4. Disable the peripheral interrupt at the peripheral register.
Step 5. Enable global interrupts (INTM = 0).
Step 6. Wait for any pending interrupt from the peripheral to be serviced by the empty ISR routine.
Step 7. Disable global interrupts (INTM = 1).
Step 8. Modify the PIE vector table to map the peripheral vector back to its original ISR.
Step 9. Clear the EALLOW bit.
Step 10. Disable the PIEIER bit for given peripheral.
Step 11. Clear the IFR bit for given peripheral group (this is safe operation on CPU IFR register).
Step 12. Clear the PIEACK bit for the PIE group.
Step 13. Enable global interrupts.

8.3.3 Flow of a Multiplexed Interrupt Request From a Peripheral to the CPU

Figure 83 shows the flow with the steps shown in circled numbers. Following the diagram, the steps are described.

Figure 83. Multiplexed Interrupt Request Flow Diagram



Step 1. Any peripheral or external interrupt within the PIE group generates an interrupt. If interrupts are enabled within the peripheral module then the interrupt request is sent to the PIE module.

Step 2. The PIE module recognizes that interrupt y within PIE group x (INTx.y) has asserted an interrupt and the appropriate PIE interrupt flag bit is latched: PIEIFRx.y = 1.

Step 3. For the interrupt request to be sent from the PIE to the CPU, both of the following conditions must be true:

(a) The proper enable bit must be set (PIEIERx.y = 1) and
(b) The PIEACKx bit for the group must be clear.

Step 4. If both conditions in 3a and 3b are true, then an interrupt request is sent to the CPU and the acknowledge bit is again set (PIEACKx = 1). The PIEACKx bit will remain set until you clear it to indicate that additional interrupts from the group can be sent from the PIE to the CPU.

Step 5. The CPU interrupt flag bit is set (CPU IFRx = 1) to indicate a pending interrupt x at the CPU level.

Step 6. If the CPU interrupt is enabled (CPU IER bit x = 1, or DBGIER bit x = 1) AND the global interrupt mask is clear (INTM = 0) then the CPU will service the INTx.

Step 7. The CPU recognizes the interrupt and performs the automatic context save, clears the IER bit, sets INTM, and clears EALLOW. All of the steps that the CPU takes in order to prepare to service the interrupt are documented in the *TM S320C28x DSP CPU and Instruction Set Reference Guide* (literature number SPRU430).

Step 8. The CPU will then request the appropriate vector from the PIE.

Step 9. For multiplexed interrupts, the PIE module uses the current value in the PIEIERx and PIEIFRx registers to decode which vector address should be used. There are two possible cases:

(a) The vector for the highest priority interrupt within the group that is both enabled in the

PIEIERx register, and flagged as pending in the PIEIFRx is fetched and used as the branch address. In this manner if an even higher priority enabled interrupt was flagged after Step 7, it will be serviced first.

(b) If no flagged interrupts within the group are enabled, then the PIE will respond with the vector for the highest priority interrupt within that group. That is the branch address used for INTx.1. This behavior corresponds to the 28x TRAP or INT instructions.

NOTE: Because the PIEIERx register is used to determine which vector will be used for the branch, you must take care when clearing bits within the PIEIERx register. The proper procedure for clearing bits within a PIEIERx register is described in Section 8.3.2. Failure to follow these steps can result in changes occurring to the PIEIERx register after an interrupt has been passed to the CPU at Step 5 in Figure 6-5. In this case, the PIE will respond as if a TRAP or INT instruction was executed unless there are other interrupts both pending and enabled.

At this point, the PIEIFRx.y bit is cleared and the CPU branches to the vector of the interrupt fetched from the PIE.

8.3.4 The PIE Vector Table

The PIE vector table (see Table 112) consists of a 256 x 16 SARAM block that can also be used as RAM (in data space only) if the PIE block is not in use. The PIE vector table contents are undefined on reset. The CPU fixes interrupt priority for INT1 to INT12. The PIE controls priority for each group of eight interrupts. For example, if INT1.1 should occur simultaneously with INT8.1, both interrupts are presented to the CPU simultaneously by the PIE block, and the CPU services INT1.1 first. If INT1.1 should occur simultaneously with INT1.8, then INT1.1 is sent to the CPU first and then INT1.8 follows. Interrupt prioritization is performed during the vector fetch portion of the interrupt processing.

When the PIE is enabled, a TRAP #1 through TRAP #12 or an INTR INT1 to INTR INT12 instruction transfers program control to the interrupt service routine corresponding to the first vector within the PIE group. For example: TRAP #1 fetches the vector from INT1.1, TRAP #2 fetches the vector from INT2.1 and so forth. Similarly an OR IFR, #16-bit operation causes the vector to be fetched from INTR1.1 to INTR12.1 locations, if the respective interrupt flag is set. All other TRAP, INTR, OR IFR,#16-bit operations fetch the vector from the respective table location. The vector table is EALLOW protected.

Out of the 96 possible MUXed interrupts in Table 111, 43 interrupts are currently used. The remaining interrupts are reserved for future devices. These reserved interrupts can be used as software interrupts if they are enabled at the PIEIFRx level, provided none of the interrupts within the group is being used by a peripheral. Otherwise, interrupts coming from peripherals may be lost by accidentally clearing their flags when modifying the PIEIFR.

To summarize, there are two safe cases when the reserved interrupts can be used as software interrupts:

1. No peripheral within the group is asserting interrupts.
2. No peripheral interrupts are assigned to the group. For example, PIE group 11 and 12 do not have any peripherals attached to them.

The interrupt grouping for peripherals and external interrupts connected to the PIE module is shown in Table 111. Each row in the table shows the 8 interrupts multiplexed into a particular CPU interrupt. The entire PIE vector table, including both MUXed and non-MUXed interrupts, is shown in Table 112.

Table 111. PIE MUXed Peripheral Interrupt Vector Table

	INTx.8	INTx.7	INTx.6	INTx.5	INTx.4	INTx.3	INTx.2	INTx.1
INT1.y	*WAKEINT*	*TINT0*	*ADCINT*	*XINT2*	*XINT1*	Reserved	*SEQ2INT*	*SEQ1INT*
	(LPM/WD)	(TIMER 0)	(ADC)	Ext. int. 2	Ext. int. 1	-	(ADC)	(ADC)
	0xD4E	0xD4C	0xD4A	0xD48	0xD46	0xD44	0xD42	0xD40
INT2.y	Reserved	Reserved	*EPWM6_ TZINT*	*EPWM5_TZINT*	*EPWM4_TZINT*	*EPWM3_TZINT*	*EPWM2_TZINT*	*EPWM1_TZINT*
	-	-	(ePWM6)	(ePWM5)	(ePWM4)	(ePWM3)	(ePWM2)	(ePWM1)
	0xD5E	0xD5C	0xD5A	0xD58	0xD56	0xD54	0xD52	0xD50
INT3.y	Reserved	Reserved	*EPWM6_ INT*	*EPWM5_INT*	*EPWM4_INT*	*EPWM3_INT*	*EPWM2_INT*	*EPWM1_INT*
	-	-	(ePWM6)	(ePWM5)	(ePWM4)	(ePWM3)	(ePWM2)	(ePWM1)
	0xD6E	0xD6C	0xD6A	0xD68	0xD66	0xD64	0xD62	0xD60
INT4.y	Reserved	Reserved	*ECAP6_INT*	*ECAP5_INT*	*ECAP4_INT*	*ECAP3_INT*	*ECAP2_INT*	*ECAP1_INT*
	-	-	(eCAP6)	(eCAP5)	(eCAP4)	(eCAP3)	(eCAP2)	(eCAP1)
	0xD7E	0xD7C	0xD7A	0xD78	0xD76	0xD74	0xD72	0xD70
INT5.y	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	*EQEP2_INT*	*EQEP1_INT*
	-	-	-	-	-	-	(eQEP2)	(eQEP1)
	0xD8E	0xD8C	0xD8A	0xD88	0xD86	0xD84	0xD82	0xD80
INT6.y	Reserved	Reserved	*MXINTA*	*MRINTA*	*MXINTB*	*MRINTB*	*SPITXINTA*	*SPIRXINTA*
			(McBSP-A)	(McBSP-A)	(McBSP-B)	(McBSP-B)	(SPI-A)	(SPI-A)
	0xD9E	0xD9C	0xD9A	0xD98	0xD96	0xD94	0xD92	0xD90
INT7.y	Reserved	Reserved	*DINTCH6*	*DINTCH5*	*DINTCH4*	*DINTCH3*	*DINTCH2*	*DINTCH1*
			(DMA6)	(DMA5)	(DMA4)	(DMA3)	(DMA2)	(DMA1)
	0xDAE	0xDAC	0xDAA	0xDA8	0xDA6	0xDA4	0xDA2	0xDA0
INT8.y	Reserved	Reserved	*SCITXINTC*	*SCIRXINTC*	Reserved	Reserved	*I2CINT2A*	*I2CINT1A*
	-	-	(SCI-C)	(SCI-C)	-	-	(I2C-A)	(I2C-A)
	0xDBE	0xDBC	0xDBA	0xDB8	0xDB6	0xDB4	0xDB2	0xDB0
INT9.y	*ECAN1INTB*	*ECAN0INTB*	*ECAN1INTA*	*ECAN0INTA*	*SCITXINTB*	*SCIRXINTB*	*SCITXINTA*	*SCIRXINTA*
	(CAN-B)	(CAN-B)	(CAN-A)	(CAN-A)	(SCI-B)	(SCI-B)	(SCI-A)	(SCI-A)
	0xDCE	0xDCC	0xDCA	0xDC8	0xDC6	0xDC4	0xDC2	0xDC0
INT10.y	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
	-	-	-	-	-	-	-	-
	0xDDE	0xDDC	0xDDA	0xDD8	0xDD6	0xDD4	0xDD2	0xDD0
INT11.y	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	
	-	-	-	-	-	-	-	-
INT12.y	*LUF*	*LVF*	Reserved	*XINT7*	*XINT6*	*XINT5*	*XINT4*	*XINT3*
	(FPU)	(FPU)	-	Ext. Int. 7	Ext. Int. 6	Ext. Int. 5	Ext. Int. 4	Ext. Int. 3
	0xDFE	0xDFC	0xDFA	0xDF8	0xDF6	0xDF4	0xDF2	0xDF0

Table 112. PIE Vector Table

Name	VECTOR ID [1]	Address [2]	Size (x16)	Description [3]		CPU Priority	PIE 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	External Interrupt 13 (XINT13) or CPU-Timer1		17	-
INT14	14	0x0000 0D1C	2	CPU-Timer2 (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	External Non-Maskable Interrupt		3	-
ILLEGAL	19	0x0000 0D26	2	Illegal Operation		-	-
USER1	20	0x0000 0D28	2	User-Defined Trap		-	-
USER2	21	0x0000 0D2A	2	User Defined Trap		-	-
USER3	22	0x0000 0D2C	2	User Defined Trap		-	-
USER4	23	0x0000 0D2E	2	User Defined Trap		-	-
USER5	24	0x0000 0D30	2	User Defined Trap		-	-
USER6	25	0x0000 0D32	2	User Defined Trap		-	-
USER7	26	0x0000 0D34	2	User Defined Trap		-	-
USER8	27	0x0000 0D36	2	User Defined Trap		-	-
USER9	28	0x0000 0D38	2	User Defined Trap		-	-
USER10	29	0x0000 0D3A	2	User Defined Trap		-	-
USER11	30	0x0000 0D3C	2	User Defined Trap		-	-
USER12	31	0x0000 0D3E	2	User Defined Trap		-	-
PIE Group 1 Vectors - MUXed into CPU INT1							
INT1.1	32	0x0000 0D40	2	SEQ1INT	(ADC)	5	1 (highest)
INT1.2	33	0x0000 0D42	2	SEQ2INT	(ADC)	5	2
INT1.3	34	0x0000 0D44	2	Reserved		5	3
INT1.4	35	0x0000 0D46	2	XINT1		5	4
INT1.5	36	0x0000 0D48	2	XINT2		5	5
INT1.6	37	0x0000 0D4A	2	ADCINT	(ADC)	5	6
INT1.7	38	0x0000 0D4C	2	TINT0	(CPU-Timer0)	5	7
INT1.8	39	0x0000 0D4E	2	WAKEINT	(LPM/WD)	5	8 (lowest)

(1) The VECTOR ID is used by DSP/BIOS.
(2) Reset is always fetched from location 0x003F FFC0 in Boot ROM.
(3) All the locations within the PIE vector table are EALLOW protected.

Table 112. PIE Vector Table (continued)

Name	VECTOR ID [1]	Address [2]	Size (x16)	Description [3]		CPU Priority	PIE Group Priority
PIE Group 2 Vectors - MUXed into CPU INT2							
INT2.1	40	0x0000 0D50	2	EPWM1_TZINT	(EPWM1)	6	1 (highest)
INT2.2	41	0x0000 0D52	2	EPWM2_TZINT	(EPWM2)	6	2
INT2.3	42	0x0000 0D54	2	EPWM3_TZINT	(EPWM3)	6	3
INT2.4	43	0x0000 0D56	2	EPWM4_TZINT	(EPWM4)	6	4
INT2.5	44	0x0000 0D58	2	EPWM5_TZINT	(EPWM5)	6	5
INT2.6	45	0x0000 0D5A	2	EPWM6_TZINT	(EPWM6)	6	6
INT2.7	46	0x0000 0D5C	2	Reserved		6	7
INT2.8	47	0x0000 0D5E	2			6	8 (lowest)
PIE Group 3 Vectors - MUXed into CPU INT3							
INT3.1	48	0x0000 0D60	2	EPWM1_INT	(EPWM1)	7	1 (highest)
INT3.2	49	0x0000 0D62	2	EPWM2_INT	(EPWM2)	7	2
INT3.3	50	0x0000 0D64	2	EPWM3_INT	(EPWM3)	7	3
INT3.4	51	0x0000 0D66	2	EPWM4_INT	(EPWM4)	7	4
INT3.5	52	0x0000 0D68	2	EPWM5_INT	(EPWM5)	7	5
INT3.6	53	0x0000 0D6A	2	EPWM6_INT	(EPWM6)	7	6
INT3.7	54	0x0000 0D6C	2	Reserved		7	7
INT3.8	55	0x0000 0D6E	2	Reserved		7	8 (lowest)
PIE Group 4 Vectors - MUXed into CPU INT4							
INT4.1	56	0x0000 0D70	2	ECAP1_INT	(ECAP1)	8	1 (highest)
INT4.2	57	0x0000 0D72	2	ECAP2_INT	(ECAP2)	8	2
INT4.3	58	0x0000 0D74	2	ECAP3_INT	(ECAP3)	8	3
INT4.4	59	0x0000 0D76	2	ECAP4_INT	(ECAP4)	8	4
INT4.5	60	0x0000 0D78	2	ECAP5_INT	(ECAP5)	8	5
INT4.6	61	0x0000 0D7A	2	ECAP6_INT	(ECAP6)	8	6
INT4.7	62	0x0000 0D7C	2	Reserved		8	7
INT4.8	63	0x0000 0D7E	2	Reserved		8	8 (lowest)
PIE Group 5 Vectors - MUXed into CPU INT5							
INT5.1	64	0x0000 0D80	2	EQEP1_INT	(EQEP1)	9	1 (highest)
INT5.2	65	0x0000 0D82	2	EQEP2_INT	(EQEP2)	9	2
INT5.3	66	0x0000 0D84	2	Reserved		9	3
INT5.4	67	0x0000 0D86	2	Reserved		9	4
INT5.5	68	0x0000 0D88	2	Reserved		9	5
INT5.6	69	0x0000 0D8A	2	Reserved		9	6
INT5.7	70	0x0000 0D8C	2	Reserved		9	7
INT5.8	71	0x0000 0D8E	2	Reserved		9	8 (lowest)
PIE Group 6 Vectors - MUXed into CPU INT6							
INT6.1	72	0x0000 0D90	2	SPIRXINTA	(SPI-A)	10	1 (highest)
INT6.2	73	0x0000 0D92	2	SPITXINTA	(SPI-A)	10	2
INT6.3	74	0x0000 0D94	2	MRINTB	(McBSP-B)	10	3
INT6.4	75	0x0000 0D96	2	MXINTB	(McBSP-B)(SPI-B)	10	4
INT6.5	76	0x0000 0D98	2	MRINTA	(McBSP-A)	10	5
INT6.6	77	0x0000 0D9A	2	MXINTA	(McBSP-A)	10	6
INT6.7	78	0x0000 0D9C	2	Reserved		10	7
INT6.8	79	0x0000 0D9E	2	Reserved		10	8 (lowest)

Table 112. PIE Vector Table (continued)

Name	VECTOR ID [1]	Address [2]	Size (x16)	Description [3]		CPU Priority	PIE Group Priority
PIE Group 7 Vectors - MUXed into CPU INT7							
INT7.1	80	0x0000 0DA0	2	DINTCH1	DMA Channel 1	11	1 (highest)
INT7.2	81	0x0000 0DA2	2	DINTCH2	DMA Channel 2	11	2
INT7.3	82	0x0000 0DA4	2	DINTCH3	DMA Channel 3	11	3
INT7.4	83	0x0000 0DA6	2	DINTCH4	DMA Channel 4	11	4
INT7.5	84	0x0000 0DA8	2	DINTCH5	DMA Channel 5	11	5
INT7.6	85	0x0000 0DAA	2	DINTCH6	DMA Channel 6	11	6
INT7.7	86	0x0000 0DAC	2	Reserved	-	11	7
INT7.8	87	0x0000 0DAE	2	Reserved	-	11	8 (lowest)
PIE Group 8 Vectors - MUXed into CPU INT8							
INT8.1	88	0x0000 0DB0	2	I2CINT1A	(I2C-A)	12	1 (highest)
INT8.2	89	0x0000 0DB2	2	I2CINT2A	(I2C-A)	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	SCIRXINTC	(SCI-C)	12	5
INT8.6	93	0x0000 0DBA	2	SCITXINTC	(SCI-C)	12	6
INT8.7	94	0x0000 0DBC	2	Reserved	-	12	7
INT8.8	95	0x0000 0DBE	2	Reserved	-	12	8 (lowest)
PIE Group 9 Vectors - MUXed into CPU INT9							
INT9.1	96	0x0000 0DC0	2	SCIRXINTA	(SCI-A)	13	1 (highest)
INT9.2	97	0x0000 0DC2	2	SCITXINTA	(SCI-A)	13	2
INT9.3	98	0x0000 0DC4	2	SCIRXINTB	(SCI-B)	13	3
INT9.4	99	0x0000 0DC6	2	SCITXINTB	(SCI-B)	13	4
INT9.5	100	0x0000 0DC8	2	ECAN0INTA	(eCAN-A)	13	5
INT9.6	101	0x0000 0DCA	2	ECAN1INTA	(eCAN-A)	13	6
INT9.7	102	0x0000 0DCC	2	ECAN0INTB	(eCAN-B)	13	7
INT9.8	103	0x0000 0DCE	2	ECAN1INTB	(eCAN-B)	13	8 (lowest)
PIE Group 10 Vectors - MUXed into CPU INT10							
INT10.1	104	0x0000 0DD0	2	Reserved	-	14	1 (highest)
INT10.2	105	0x0000 0DD2	2	Reserved		14	2
INT10.3	106	0x0000 0DD4	2	Reserved		14	3
INT10.4	107	0x0000 0DD6	2	Reserved		14	4
INT10.5	108	0x0000 0DD8	2	Reserved		14	5
INT10.6	109	0x0000 0DDA	2	Reserved		14	6
INT10.7	110	0x0000 0DDC	2	Reserved		14	7
INT10.8	111	0x0000 0DDE	2	Reserved		14	8 (lowest)
PIE Group 11 Vectors - MUXed into CPU INT11							
INT11.1	112	0x0000 0DE0	2	Reserved		15	1 (highest)
INT11.2	113	0x0000 0DE2	2	Reserved		15	2
INT11.3	114	0x0000 0DE4	2	Reserved		15	3
INT11.4	115	0x0000 0DE6	2	Reserved		15	4
INT11.5	116	0x0000 0DE8	2	Reserved		15	5

Table 112. PIE Vector Table (continued)

Name	VECTOR ID [1]	Address [2]	Size (x16)	Description [3]		CPU Priority	PIE Group Priority
INT11.6	117	0x0000 0DEA	2	Reserved		15	6
INT11.7	118	0x0000 0DEC	2	Reserved		15	7
INT11.8	119	0x0000 0DEE	2	Reserved		15	8 (lowest)
PIE Group 12 Vectors - Muxed into CPU INT12							
INT12.1	120	0x0000 0DF0	2	XINT3		16	1 (highest)
INT12.2	121	0x0000 0DF2	2	XINT4		16	2
INT12.3	122	0x0000 0DF4	2	XINT5		16	3
INT12.4	123	0x0000 0DF6	2	XINT6		16	4
INT12.5	124	0x0000 0DF8	2	XINT7		16	5
INT12.6	125	0x0000 0DFA	2	Reserved		16	6
INT12.7	126	0x0000 0DFC	2	LVF	FPU	16	7
INT12.8	127	0x0000 0DFE	2	LUF	FPU	16	8 (lowest)

8.4 PIE Configuration Registers

The registers controlling the functionality of the PIE block are shown in Table 113.

Table 113. PIE Configuration and Control Registers

Name	Address	Size (x16)	Description
PIECTRL	0x0000 - 0CE0	1	PIE, Control Register
PIEACK	0x0000 - 0CE1	1	PIE, Acknowledge Register
PIEIER1	0x0000 - 0CE2	1	PIE, INT1 Group Enable Register
PIEIFR1	0x0000 - 0CE3	1	PIE, INT1 Group Flag Register
PIEIER2	0x0000 - 0CE4	1	PIE, INT2 Group Enable Register
PIEIFR2	0x0000 - 0CE5	1	PIE, INT2 Group Flag Register
PIEIER3	0x0000 - 0CE6	1	PIE, INT3 Group Enable Register
PIEIFR3	0x0000 - 0CE7	1	PIE, INT3 Group Flag Register
PIEIER4	0x0000 - 0CE8	1	PIE, INT4 Group Enable Register
PIEIFR4	0x0000 - 0CE9	1	PIE, INT4 Group Flag Register
PIEIER5	0x0000 - 0CEA	1	PIE, INT5 Group Enable Register
PIEIFR5	0x0000 - 0CEB	1	PIE, INT5 Group Flag Register
PIEIER6	0x0000 - 0CEC	1	PIE, INT6 Group Enable Register
PIEIFR6	0x0000 - 0CED	1	PIE, INT6 Group Flag Register
PIEIER7	0x0000 - 0CEE	1	PIE, INT7 Group Enable Register
PIEIFR7	0x0000 - 0CEF	1	PIE, INT7 Group Flag Register
PIEIER8	0x0000 - 0CF0	1	PIE, INT8 Group Enable Register
PIEIFR8	0x0000 - 0CF1	1	PIE, INT8 Group Flag Register
PIEIER9	0x0000 - 0CF2	1	PIE, INT9 Group Enable Register
PIEIFR9	0x0000 - 0CF3	1	PIE, INT9 Group Flag Register
PIEIER10	0x0000 - 0CF4	1	PIE, INT10 Group Enable Register
PIEIFR10	0x0000 - 0CF5	1	PIE, INT10 Group Flag Register
PIEIER11	0x0000 - 0CF6	1	PIE, INT11 Group Enable Register
PIEIFR11	0x0000 - 0CF7	1	PIE, INT11 Group Flag Register
PIEIER12	0x0000 - 0CF8	1	PIE, INT12 Group Enable Register
PIEIFR12	0x0000 - 0CF9	1	PIE, INT12 Group Flag Register

8.5 PIE Interrupt Registers

Figure 84. PIECTRL Register (Address CE0)

15	1	0
PIEVECT		ENPIE
R-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 114. PIECTRL Register Address Field Descriptions

Bits	Field	Value	Description
15-1	PIEVECT		These bits indicate the address within the PIE vector table from which the vector was fetched. The least significant bit of the address is ignored and only bits 1 to 15 of the address is shown. You can read the vector value to determine which interrupt generated the vector fetch. **For Example:** If PIECTRL = 0x0D27 then the vector from address 0x0D26 (illegal operation) was fetched.
0	ENPIE		Enable vector fetching from PIE vector table. **Note:** The reset vector is never fetched from the PIE, even when it is enabled. This vector is always fetched from boot ROM.
		0	If this bit is set to 0, the PIE block is disabled and vectors are fetched from the CPU vector table in boot ROM. All PIE block registers (PIEACK, PIEIFR, PIEIER) can be accessed even when the PIE block is disabled.
		1	When ENPIE is set to 1, all vectors, except for reset, are fetched from the PIE vector table. The reset vector is always fetched from the boot ROM.

Figure 85. PIE Interrupt Acknowledge Register (PIEACK) Register (Address CE1)

15	12	11	0
Reserved		PIEACK	
R-0		R/W1C-1	

LEGEND: R/W1C = Read/Write 1 to clear; R = Read only; -*n* = value after reset

Table 115. PIE Interrupt Acknowledge Register (PIEACK) Field Descriptions

Bits	Field	Value	Description
15-12	Reserved		Reserved
11-0	PIEACK		Each bit in PIEACK refers to a specific PIE group. Bit 0 refers to interrupts in PIE group 1 that are MUXed into $\overline{\text{INT1}}$ up to Bit 11, which refers to PIE group 12 which is MUXed into CPU $\overline{\text{INT12}}$
		bit x = 0 (1)	If a bit reads as a 0, it indicates that the PIE can send an interrupt from the respective group to the CPU. Writes of 0 are ignored.
		bit x = 1	Reading a 1 indicates if an interrupt from the respective group has been sent to the CPU and all other interrupts from the group are currently blocked. Writing a 1 to the respective interrupt bit clears the bit and enables the PIE block to drive a pulse into the CPU interrupt input if an interrupt is pending for that group.

(1) bit x = PIEACK bit 0 - PIEACK bit 11. Bit 0 refers to CPU $\overline{\text{INT1}}$ up to Bit 11, which refers to CPU $\overline{\text{INT12}}$

8.5.1 PIE Interrupt Flag Registers

There are twelve PIEIFR registers, one for each CPU interrupt used by the PIE module (INT1-INT12).

Figure 86. PIEIFRx Register (x = 1 to 12)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
INTx.8	INTx.7	INTx.6	INTx.5	INTx.4	INTx.3	INTx.2	INTx.1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 116. PIEIFRx Register Field Descriptions

Bits	Field	Description
15-8	Reserved	Reserved
7	INTx.8	These register bits indicate whether an interrupt is currently active. They behave very much like the CPU interrupt flag register. When an interrupt is active, the respective register bit is set. The bit is cleared when the interrupt is serviced or by writing a 0 to the register bit. This register can also be read to determine which interrupts are active or pending. x = 1 to 12. INTx means CPU INT1 to INT12
6	INTx.7	
5	INTx.6	
4	INTx.5	The PIEIFR register bit is cleared during the interrupt vector fetch portion of the interrupt processing.
3	INTx.4	Hardware has priority over CPU accesses to the PIEIFR registers.
2	INTx.3	
1	INTx.2	
0	INTx.1	

NOTE: Never clear a PIEIFR bit. An interrupt may be lost during the read-modify-write operation. See Section Section 8.3.1 for a method to clear flagged interrupts.

8.5.2 PIE Interrupt Enable Registers

There are twelve PIEIER registers, one for each CPU interrupt used by the PIE module (INT1-INT12).

Figure 87. PIEIERx Register (x = 1 to 12)

15							8
Reserved							
R-0							
7	6	5	4	3	2	1	0
INTx.8	INTx.7	INTx.6	INTx.5	INTx.4	INTx.3	INTx.2	INTx.1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 117. PIEIERx Register (x = 1 to 12) Field Descriptions

Bits	Field	Description
15-8	Reserved	Reserved
7	INTx.8	These register bits individually enable an interrupt within a group and behave very much like the core interrupt enable register. Setting a bit to 1 enables the servicing of the respective interrupt. Setting a bit to 0 disables the servicing of the interrupt. x = 1 to 12. INTx means CPU INT1 to INT12
6	INTx.7	
5	INTx.6	
4	INTx.5	
3	INTx.4	
2	INTx.3	
1	INTx.2	
0	INTx.1	

NOTE: Care must be taken when clearing PIEIER bits during normal operation. See Section Section 8.3.2 for the proper procedure for handling these bits.

8.5.3 CPU Interrupt Flag Register (IFR)

The CPU interrupt flag register (IFR), is a 16-bit, CPU register and is used to identify and clear pending interrupts. The IFR contains flag bits for all the maskable interrupts at the CPU level (INT1-INT14, DLOGINT and RTOSINT). When the PIE is enabled, the PIE module multiplexes interrupt sources for INT1-INT12.

When a maskable interrupt is requested, the flag bit in the corresponding peripheral control register is set to 1. If the corresponding mask bit is also 1, the interrupt request is sent to the CPU, setting the corresponding flag in the IFR. This indicates that the interrupt is pending or waiting for acknowledgment.

To identify pending interrupts, use the PUSH IFR instruction and then test the value on the stack. Use the OR IFR instruction to set IFR bits and use the AND IFR instruction to manually clear pending interrupts. All pending interrupts are cleared with the AND IFR #0 instruction or by a hardware reset.

The following events also clear an IFR flag:

- The CPU acknowledges the interrupt.
- The 28x device is reset.

NOTE:

1. To clear a CPU IFR bit, you must write a zero to it, not a one.
2. When a maskable interrupt is acknowledged, only the IFR bit is cleared automatically. The flag bit in the corresponding peripheral control register is not cleared. If an application requires that the control register flag be cleared, the bit must be cleared by software.
3. When an interrupt is requested by an INTR instruction and the corresponding IFR bit is set, the CPU does not clear the bit automatically. If an application requires that the IFR bit be cleared, the bit must be cleared by software.
4. IMR and IFR registers pertain to core-level interrupts. All peripherals have their own interrupt mask and flag bits in their respective control/configuration registers. Note that several peripheral interrupts are grouped under one core-level interrupt.

Figure 88. Interrupt Flag Register (IFR) — CPU Register

15	14	13	12	11	10	9	8
RTOSINT	DLOGINT	INT14	INT13	INT12	INT11	INT10	INT9
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 118. Interrupt Flag Register (IFR) — CPU Register Field Descriptions

Bits	Field	Value	Description
15	RTOSINT		Real-time operating system flag. RTOSINT is the flag for RTOS interrupts.
		0	No RTOS interrupt is pending
		1	At least one RTOS interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
14	DLOGINT		Data logging interrupt fag. DLOGINT is the flag for data logging interrupts.
		0	No DLOGINT is pending
		1	At least one DLOGINT interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
13	INT14		Interrupt 14 flag. INT14 is the flag for interrupts connected to CPU interrupt level INT14.
		0	No INT14 interrupt is pending
		1	At least one INT14 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
12	INT13		Interrupt 13 flag. INT13 is the flag for interrupts connected to CPU interrupt level INT13I.
		0	No INT13 interrupt is pending
		1	At least one INT13 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
11	INT12		Interrupt 12 flag. INT12 is the flag for interrupts connected to CPU interrupt level INT12.
		0	No INT12 interrupt is pending
		1	At least one INT12 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
10	INT11		Interrupt 11 flag. INT11 is the flag for interrupts connected to CPU interrupt level INT11.
		0	No INT11 interrupt is pending
		1	At least one INT11 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
9	INT10		Interrupt 10 flag. INT10 is the flag for interrupts connected to CPU interrupt level INT10.
		0	No INT10 interrupt is pending
		1	At least one INT6 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
8	INT9		Interrupt 9 flag. INT9 is the flag for interrupts connected to CPU interrupt level INT6.
		0	No INT9 interrupt is pending
		1	At least one INT9 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
7	INT8		Interrupt 8 flag. INT8 is the flag for interrupts connected to CPU interrupt level INT6.
		0	No INT8 interrupt is pending
		1	At least one INT8 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
6	INT7		Interrupt 7 flag. INT7 is the flag for interrupts connected to CPU interrupt level INT7.
		0	No INT7 interrupt is pending
		1	At least one INT7 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request

Table 118. Interrupt Flag Register (IFR) — CPU Register Field Descriptions (continued)

Bits	Field	Value	Description
5	INT6		Interrupt 6 flag. INT6 is the flag for interrupts connected to CPU interrupt level INT6.
		0	No INT6 interrupt is pending
		1	At least one INT6 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
4	INT5		Interrupt 5 flag. INT5 is the flag for interrupts connected to CPU interrupt level INT5.
		0	No INT5 interrupt is pending
		1	At least one INT5 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
3	INT4		Interrupt 4 flag. INT4 is the flag for interrupts connected to CPU interrupt level INT4.
		0	No INT4 interrupt is pending
		1	At least one INT4 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
2	INT3		Interrupt 3 flag. INT3 is the flag for interrupts connected to CPU interrupt level INT3.
		0	No INT3 interrupt is pending
		1	At least one INT3 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
1	INT2		Interrupt 2 flag. INT2 is the flag for interrupts connected to CPU interrupt level INT2.
		0	No INT2 interrupt is pending
		1	At least one INT2 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
0	INT1		Interrupt 1 flag. INT1 is the flag for interrupts connected to CPU interrupt level INT1.
		0	No INT1 interrupt is pending
		1	At least one INT1 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request

8.5.4 Interrupt Enable Register (IER) and Debug Interrupt Enable Register (DBGIER)

The IER is a 16-bit CPU register. The IER contains enable bits for all the maskable CPU interrupt levels (INT1-INT14, RTOSINT and DLOGINT). Neither NMI nor XRS is included in the IER; thus, IER has no effect on these interrupts.

You can read the IER to identify enabled or disabled interrupt levels, and you can write to the IER to enable or disable interrupt levels. To enable an interrupt level, set its corresponding IER bit to one using the OR IER instruction. To disable an interrupt level, set its corresponding IER bit to zero using the AND IER instruction. When an interrupt is disabled, it is not acknowledged, regardless of the value of the INTM bit. When an interrupt is enabled, it is acknowledged if the corresponding IFR bit is one and the INTM bit is zero.

When using the OR IER and AND IER instructions to modify IER bits make sure they do not modify the state of bit 15 (RTOSINT) unless a real-time operating system is present.

When a hardware interrupt is serviced or an INTR instruction is executed, the corresponding IER bit is cleared automatically. When an interrupt is requested by the TRAP instruction the IER bit is not cleared automatically. In the case of the TRAP instruction if the bit needs to be cleared it must be done by the interrupt service routine.

At reset, all the IER bits are cleared to 0, disabling all maskable CPU level interrupts.

The IER register is shown in Figure 89, and descriptions of the bits follow the figure.

Figure 89. Interrupt Enable Register (IER) — CPU Register

15	14	13	12	11	10	9	8
RTOSINT	DLOGINT	INT14	INT13	INT12	INT11	INT10	INT9
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 119. Interrupt Enable Register (IER) — CPU Register Field Descriptions

Bits	Field	Value	Description
15	RTOSINT		Real-time operating system interrupt enable. RTOSINT enables or disables the CPU RTOS interrupt.
		0	Level INT6 is disabled
		1	Level INT6 is enabled
14	DLOGINT		Data logging interrupt enable. DLOGINT enables or disables the CPU data logging interrupt.
		0	Level INT6 is disabled
		1	Level INT6 is enabled
13	INT14		Interrupt 14 enable. INT14 enables or disables CPU interrupt level INT14.
		0	Level INT14 is disabled
		1	Level INT14 is enabled
12	INT13		Interrupt 13 enable. INT13 enables or disables CPU interrupt level INT13.
		0	Level INT13 is disabled
		1	Level INT13 is enabled
11	INT12		Interrupt 12 enable. INT12 enables or disables CPU interrupt level INT12.
		0	Level INT12 is disabled
		1	Level INT12 is enabled
10	INT11		Interrupt 11 enable. INT11 enables or disables CPU interrupt level INT11.
		0	Level INT11 is disabled
		1	Level INT11 is enabled
9	INT10		Interrupt 10 enable. INT10 enables or disables CPU interrupt level INT10.
		0	Level INT10 is disabled
		1	Level INT10 is enabled
8	INT9		Interrupt 9 enable. INT9 enables or disables CPU interrupt level INT9.
		0	Level INT9 is disabled
		1	Level INT9 is enabled
7	INT8		Interrupt 8 enable. INT8 enables or disables CPU interrupt level INT8.
		0	Level INT8 is disabled
		1	Level INT8 is enabled
6	INT7		Interrupt 7 enable. INT7 enables or disables CPU interrupt level INT7.
		0	Level INT7 is disabled
		1	Level INT7 is enabled
5	INT6		Interrupt 6 enable. INT6 enables or disables CPU interrupt level INT6.
		0	Level INT6 is disabled
		1	Level INT6 is enabled
4	INT5		Interrupt 5 enable.INT5 enables or disables CPU interrupt level INT5.
		0	Level INT5 is disabled
		1	Level INT5 is enabled

Table 119. Interrupt Enable Register (IER) — CPU Register Field Descriptions (continued)

Bits	Field	Value	Description
3	INT4		Interrupt 4 enable.INT4 enables or disables CPU interrupt level INT4.
		0	Level INT4 is disabled
		1	Level INT4 is enabled
2	INT3		Interrupt 3 enable.INT3 enables or disables CPU interrupt level INT3.
		0	Level INT3 is disabled
		1	Level INT3 is enabled
1	INT2		Interrupt 2 enable.INT2 enables or disables CPU interrupt level INT2.
		0	Level INT2 is disabled
		1	Level INT2 is enabled
0	INT1		Interrupt 1 enable.INT1 enables or disables CPU interrupt level INT1.
		0	Level INT1 is disabled
		1	Level INT1 is enabled

The Debug Interrupt Enable Register (DBGIER) is used only when the CPU is halted in real-time emulation mode. An interrupt enabled in the DBGIER is defined as a time-critical interrupt. When the CPU is halted in real-time mode, the only interrupts that are serviced are time-critical interrupts that are also enabled in the IER. If the CPU is running in real-time emulation mode, the standard interrupt-handling process is used and the DBGIER is ignored.

As with the IER, you can read the DBGIER to identify enabled or disabled interrupts and write to the DBGIER to enable or disable interrupts. To enable an interrupt, set its corresponding bit to 1. To disable an interrupt, set its corresponding bit to 0. Use the PUSH DBGIER instruction to read from the DBGIER and POP DBGIER to write to the DBGIER register. At reset, all the DBGIER bits are set to 0.

Figure 90. Debug Interrupt Enable Register (DBGIER) — CPU Register

15	14	13	12	11	10	9	8
RTOSINT	DLOGINT	INT14	INT13	INT12	INT11	INT10	INT9
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 120. Debug Interrupt Enable Register (DBGIER) — CPU Register Field Descriptions

Bits	Field	Value	Description
15	RTOSINT		Real-time operating system interrupt enable. RTOSINT enables or disables the CPU RTOS interrupt.
		0	Level INT6 is disabled
		1	Level INT6 is enabled
14	DLOGINT	.	Data logging interrupt enable. DLOGINT enables or disables the CPU data logging interrupt
		0	Level INT6 is disabled
		1	Level INT6 is enabled
13	INT14	.	Interrupt 14 enable. INT14 enables or disables CPU interrupt level INT14
		0	Level INT14 is disabled
		1	Level INT14 is enabled
12	INT13		Interrupt 13 enable. INT13 enables or disables CPU interrupt level INT13.
		0	Level INT13 is disabled
		1	Level INT13 is enabled

Table 120. Debug Interrupt Enable Register (DBGIER) — CPU Register Field Descriptions (continued)

Bits	Field	Value	Description
11	INT12		Interrupt 12 enable. INT12 enables or disables CPU interrupt level INT12.
		0	Level INT12 is disabled
		1	Level INT12 is enabled
10	INT11		Interrupt 11 enable. INT11 enables or disables CPU interrupt level INT11.
		0	Level INT11 is disabled
		1	Level INT11 is enabled
9	INT10		Interrupt 10 enable. INT10 enables or disables CPU interrupt level INT10.
		0	Level INT10 is disabled
		1	Level INT10 is enabled
8	INT9		Interrupt 9 enable. INT9 enables or disables CPU interrupt level INT9.
		0	Level INT9 is disabled
		1	Level INT9 is enabled
7	INT8		Interrupt 8 enable. INT8 enables or disables CPU interrupt level INT8.
		0	Level INT8 is disabled
		1	Level INT8 is enabled
6	INT7		Interrupt 7 enable. INT7 enables or disables CPU interrupt level INT77.
		0	Level INT7 is disabled
		1	Level INT7 is enabled
5	INT6		Interrupt 6 enable. INT6 enables or disables CPU interrupt level INT6.
		0	Level INT6 is disabled
		1	Level INT6 is enabled
4	INT5		Interrupt 5 enable.INT5 enables or disables CPU interrupt level INT5.
		0	Level INT5 is disabled
		1	Level INT5 is enabled
3	INT4		Interrupt 4 enable.INT4 enables or disables CPU interrupt level INT4.
		0	Level INT4 is disabled
		1	Level INT4 is enabled
2	INT3		Interrupt 3 enable.INT3 enables or disables CPU interrupt level INT3.
		0	Level INT3 is disabled
		1	Level INT3 is enabled
1	INT2		Interrupt 2 enable.INT2 enables or disables CPU interrupt level INT2.
		0	Level INT2 is disabled
		1	Level INT2 is enabled
0	INT1		Interrupt 1 enable.INT1 enables or disables CPU interrupt level INT1.
		0	Level INT1 is disabled
		1	Level INT1 is enabled

8.6 External Interrupt Control Registers

Seven external interrupts, XINT1 –XINT7 are supported. XINT13 is multiplexed with one non-maskable interrupt XNMI. Each of these external interrupts can be selected for negative or positive edge triggered and can also be enabled or disabled (including XNMI). The masked interrupts also contain a 16-bit free running up counter that is reset to zero when a valid interrupt edge is detected. This counter can be used to accurately time stamp the interrupt.

XINT1CR through XINT7 CR are identical except for the interrupt number; therefore, Figure 91 and Table 121 represent registers for external interrupts 1 through 7 as XINT*n*CR where *n* = the interrupt number.

Figure 91. External Interrupt *n* Control Register (XINT*n*CR)

15	4	3	2	1	0
Reserved		Polarity		Reserved	Enable
R-0		R/W-0		R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 121. External Interrupt *n* Control Register (XINT*n*CR) Field Descriptions

Bits	Field	Value	Description
15-4	Reserved		Reads return zero; writes have no effect.
3-2	Polarity		This read/write bit determines whether interrupts are generated on the rising edge or the falling edge of a signal on the pin.
		00	Interrupt generated on a falling edge (high-to-low transition)
		01	Interrupt generated on a rising edge (low-to-high transition)
		10	Interrupt is generated on a falling edge (high-to-low transition)
		11	Interrupt generated on both a falling edge and a rising edge (high-to-low and low-to-high transition)
1	Reserved		Reads return zero; writes have no effect
0	Enable		This read/write bit enables or disables external interrupt XINT*n.*
		0	Disable interrupt
		1	Enable interrupt

Figure 92. External NMI Interrupt Control Register (XNMICR) — Address 7077h

15	4	3	2	1	0
Reserved		Polarity		Select	Enable
R-0		R/W-0		R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 122. External NMI Interrupt Control Register (XNMICR) Field Descriptions

Bits	Field	Value	Description
15-4	Reserved		Reads return zero; writes have no effect.
3-2	Polarity		This read/write bit determines whether interrupts are generated on the rising edge or the falling edge of the signal on the pin.
		00	Interrupt generated on a falling edge (high-to-low transition)
		01	Interrupt generated on a rising edge low-to-high transition)
		10	Interrupt is generated on a falling edge (high to low transition)
		11	Interrupt generated on both a falling edge and a rising edge (high to low and low to high transition)
1	Select		Select the source for INT13
		0	Timer 1 connected To INT13
		1	XNMI_XINT13 connected To INT13

Table 122. External NMI Interrupt Control Register (XNMICR) Field Descriptions (continued)

Bits	Field	Value	Description
0	Enable		This read/write bit enables or disables external interrupt NMI
		0	Disable XNMI interrupt
		1	Enable XNMI interrupt

The XNMI Control Register (XNMICR) can be used to enable or disable the NMI interrupt to the CPU. In addition, you can select the source for the INT13 CPU interrupt. The source of the INT13 interrupt can be either the internal CPU Timer1 or the external GPIO signal assigned to XNMI.

The INT13 interrupt can be connected to XNMI_XINT13 for customer use.

Table 123 shows the relationship between the XNMICR Register settings and the interrupt sources to the 28x CPU.

Table 123. XNMICR Register Settings and Interrupt Sources

XNMICR	Register Bits	28x CPU Interrupt		Timestamp
ENABLE	SELECT	NMI Source	INT13 Source	(XNMICTR)
0	0	Disabled	CPU Timer 1	None
0	1	Disabled	XNMI	None
1	0	XNMI	CPU Timer 1	XNMI
1	1	Disabled	XNMI	XNMI

For XINT1 and XINT2, there is also a 16-bit counter that is reset to 0x000 whenever an interrupt edge is detected. These counters can be used to accurately time stamp an occurrence of the interrupt.

Figure 93. External Interrupt 1 Counter (XINT1CTR) (Address 7078h)

15	0
INTCTR[15-0]	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 124. External Interrupt 1 Counter (XINT1CTR) Field Descriptions

Bits	Field	Description
15-0	INTCTR	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. When the interrupt is disabled, the counter stops. The counter is a free-running counter and wraps 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.

Figure 94. External Interrupt 2 Counter (XINT2CTR) (Address 7079h)

15	0
INTCTR[15-0]	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 125. External Interrupt 2 Counter (XINT2CTR) Field Descriptions

Bits	Field	Description
15-0	INTCTR	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. When the interrupt is disabled, the counter stops. The counter is a free-running counter and wraps 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.

Figure 95. External NMI Interrupt Counter (XNMICTR) (Address 707Fh)

15	0
INTCTR[15-0]	
R-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 126. External NMI Interrupt Counter (XNMICTR) Field Descriptions

Bits	Field	Description
15-0	INTCTR	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. When the interrupt is disabled, the counter stops. The counter is a free-running counter and wraps 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.

Appendix A Revision History

This document has been revised to include the following technical change(s).

Table 127. Technical Changes

Location	Additions, Deletions, Modifications
Figure 7	STDBYWAIT reset value changed to R/W- 0x1FF
Figure 8	ACTIVEWAIT reset value changed to R/W-0x1FF
Figure 9	PAGEWAIT reset value changed to R/W-0xF; RANDWAIT reset value changed to R/W-0xF
Section 2.1	Added a sentence to the Flash Memory Paged Access section: "F2833x/F2823x devices require a minimum of 1 wait-state for PAGE/RANDOM flash access."
Section 5.2.4	Deleted the sentence, "When the CPU writes to the PLLCR[DIV]..."
Table 22	For bit 7, deleted "Select divide by 8 for CLKIN"
Section 5.5	Re-worded the paragraph just above Table 31 .
Figure 40	Added a note to this figure.
Figure 73	Added this register.
Table 43	Changed the sentence, "then the signal will be sampled at 150 MHz or one sample every **ns**" to:" then the signal will be sampled at 150 MHz or one sample every **6.67 ns**."

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Communications and Telecom	www.ti.com/communications
DSP	dsp.ti.com	Computers and Peripherals	www.ti.com/computers
Clocks and Timers	www.ti.com/clocks	Consumer Electronics	www.ti.com/consumer-apps
Interface	interface.ti.com	Energy	www.ti.com/energy
Logic	logic.ti.com	Industrial	www.ti.com/industrial
Power Mgmt	power.ti.com	Medical	www.ti.com/medical
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Space, Avionics & Defense	www.ti.com/space-avionics-defense
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video and Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless-apps

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265