TMS320F28x™ System Control and Interrupts Peripheral Reference Guide

Literature Number: SPRU078
May 2002

TEXAS INSTRUMENTS

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 that third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of 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.

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.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

Contents

Figures

Tables

Chapter 1

Memory

This chapter describes how flash and one-time programmable (OTP) memories can be used with the 28x™ device and peripherals. It also includes the registers associated with memory.

Topic **Page**

28x is a trademark of Texas Instruments.

1.1 Flash and OTP Memory

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

1.1.1 Flash Memory

The on-chip flash in flash devices is uniformly mapped in both program and data memory space. This flash memory is always enabled on 28x devices and features:

- Multiple sectors
- Code security
- Low Power Modes
- Configurable waitstates that can be adjusted based on CPU frequency.
- Flash pipeline mode for improved performance

1.1.2 OTP Memory

The 2K x 16 of one-time programmable (OTP) memory is used for storing TI-specific engineering and manufacturing information. The remaining memory is available for users to program their own data or code.

The flash and OTP memory can be configured by the registers shown in Table 1–1.

Table 1–1. Flash/OTP Configuration Registers

Name	Address	Size (x16)	Description
			Configuration Registers
FOPT	0x0000–0A80	1	Flash Option Register
Reserved	0x0000–0A81	1	Reserved
FPWR	0x0000–0A82	1	Flash Power Modes Register
FSTATUS	0x0000–0A83	1	Status Register
FSTDBYWAIT	0x0000–0A84	1	Flash Sleep To Standby Wait State Register
FACTIVEWAIT	0x0000–0A85	1	Flash Standby To Active Wait State Register
FBANKWAIT	0x0000–0A86	1	Flash Read Access Wait State Register
FOTPWAIT	0x0000–0A87	1	OTP Read Access Wait State Register

Write access to the registers, can only be enabled 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. These registers support both 16 and 32-bit accesses.

Note: Flash configuration registers should not be accessed while an access is in progress in flash or OTP memory

The flash registers should not be accessed from code that is running from OTP or flash memory or while an access 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 for this.

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

1.2 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 will be in a sleep state (lowest power). A CPU read or fetch accesses to the flash/OTP memory map area will stall the CPU. This access will automatically initiate a change in power modes to the active or read state.
- Standby State: In this state, the bank and pump are in standby power mode state. A CPU read or fetch accesses to the flash/OTP memory map area will stall the CPU. This will automatically initiate a change in power modes to the active state.
- 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 for improving fetch performance.

The flash/OTP bank and pump are always in the same power mode during read or execution operations from the flash/OTP.

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 will instantaneously move the flash/OTP bank to the lower power state. This register should be accessed only by code running outside the flash/OTP memory.
- Change from a lower power state to a higher power state:
  - Change the FPWR register from a lower state to a higher state. This access starts to bring the flash/OTP memory to the higher state.
  - Access the flash or OTP memory by a read access or program fetch access. This access automatically starts to bring the flash/OTP memory to the active state.

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

The duration of the delay is determined by the FSTDBYWAIT and FACTIVEWAIT registers. Moving from the sleep state to a standby state will be delayed

by a count determined by the FSTDBYWAIT register. Moving from the standby state to the active state will be delayed by a count determined by the FACTIVE-WAIT register. Moving from the sleep mode (lowest power) to the active mode (highest power) will be delayed by FSTDBYWAIT + FACTIVEWAIT.

1.2.1 Flash and OTP Performance

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

- ❏ 32-bit Instruction Fetch
- ❏ 16 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 three types:

- ❏ Flash Memory Random Access: The number of wait states, for a random access, is configured by the RANDWAIT bits in the FBANKWAIT register. This register defaults to a worst-case count and the user needs to program the appropriate number of wait states to improve performance based on the CPU clock rate and the access time of the flash.
- ❏ Flash Memory Paged Access: The flash array is organized into rows and columns. The rows contain 2048 bits of information. The first access to a row is considered a random access. Subsequent accesses within the same row can be made with a faster access time. This is termed a PAGE access.

  The flash can take advantage of this by the user configuring a lower number of wait states in the PAGEWAIT bits in the FBANKWAIT register. This mode will work for data space and program space reads as well as instruction fetches. See the device datasheet for more information on the access time of the flash.
- ❏ OTP Access: Read or fetch accesses to the OTP are controlled by the OTPWAIT register bits in the FOTPWAIT register. Accesses to the OTP take longer than the flash and there is no paged mode.

Notes: 1) Writes to the flash/OTP memory map area are ignored. They complete in a single cycle.

2) When 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.

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

1.2.2 Flash Pipelined Mode

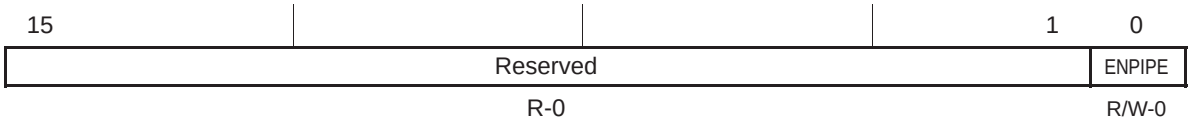
Flash memory is typically used for storing user code. To improve the performance of code execution, a flash pipeline mode is implemented. The ENPIPE

bit in the FOPT register enables this flash pipeline mode. This mode is independent of the CPU pipeline.

In this mode, a prefetch mechanism is used to reduce the effects of flash waitstates on overall code performance. Using this technique, it is possible to improve the overall efficiency of code execution from flash quite significantly.

1.3 Flash and OTP Registers

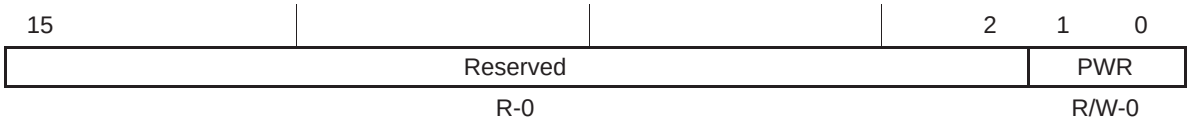
Figure 1–1. Flash Options (FOPT) Register



Legend: R = Read access, -0 = value after reset

Bits	Name	Description
15–1	Reserved	
0	ENPIPE	Enable Pipeline Mode Bit: Pipeline mode is active when this bit is set. The pipeline mode improves performance of instruction fetches by pre-fetching instructions. **Note:** When pipeline mode is enabled, the flash waitstates (paged and random) must be greater than zero.

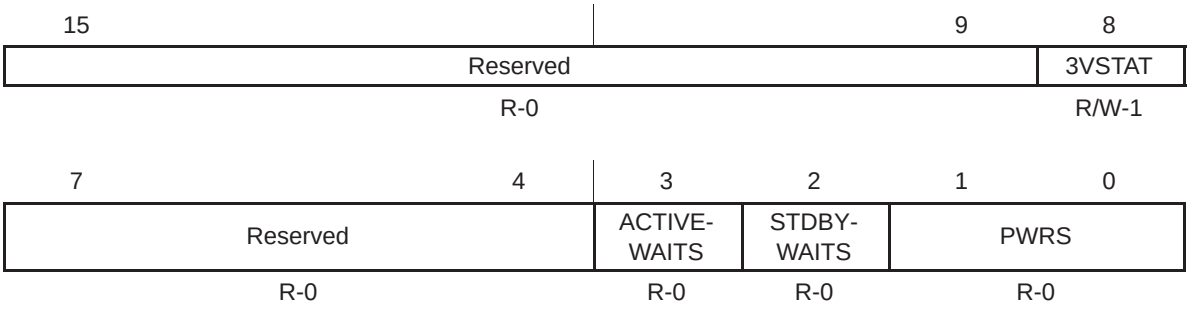
Figure 1–2. Flash Power Register (FPWR)



Legend: R = Read access, -0 = value after reset

Bits	Name	Description
15–2	Reserved	
1–0	PWR	Set Default Power Mode Bits: These bits set the default power mode of the bank or pump: 00 Pump and Bank Sleep (lowest power) 01 Pump and Bank Standby 10 Reserved (no effect) 11 Pump and Bank Active (highest power)

Figure 1–3. Flash Status Register (FSTATUS)

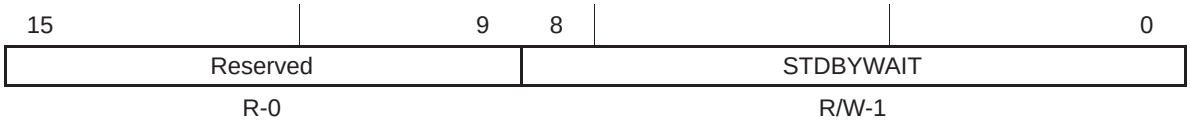


Legend: R = Read access, -0 = value after reset

Bits	Name	Description
15–9	Reserved	
8	3VSTAT	V_{DD3V} 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 3 V supply went out of allowable range. This bit is cleared by writing a 1, writes of 0 are ignored.
7–4	Reserved	
3	ACTIVEWAITS	Bank and Pump Standby To Active Wait Counter Status Bit: This bit indicates whether the respective wait counter is currently timing out an access. If the bit is set, then the counter is counting. If the bit is 0, then the counter is not counting.
2	STDBYWAITS	Bank and Pump Sleep To Standby Wait Counter Status Bit: This bit indicates whether the respective wait counter is currently timing out an access. If the bit is set, then the counter is counting. If the bit is 0, then the counter is not counting.
1–0	PWRS	Power Modes Status Bits: These bits indicate the current power mode the device is in: 00 Pump and Bank Sleep (lowest power) 01 Pump and Bank Standby 10 Reserved 11 Pump and Bank Active (highest power) **Note:** The above bits only get set to the new power mode once the appropriate timing delays have expired.

1.3.1 Flash Sleep to Standby Wait Counter

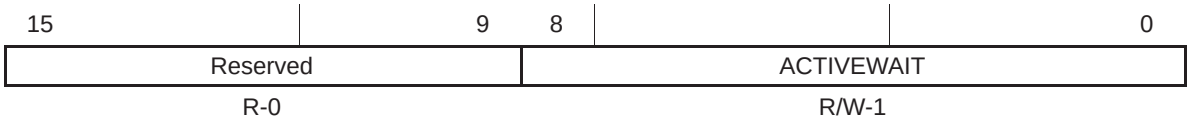
Figure 1–4. Flash Standby Wait (FSTDBYWAIT) Register



Legend: R = Read access, -0 = value after reset

Bits	Name	Description
15–9	Reserved	
8–0	STDBYWAIT	Bank and Pump Sleep To Standby Wait Count: When the bank and pump modules are in sleep mode and a write is performed to the PWR bits in the FPWR register (to change to a higher default power mode) or a CPU read or fetch access is performed to the flash bank or OTP, then a counter is initiated with the value specified in these register bits. The power mode lines for the Bank and Pump are set for standby mode. The counter then counts down to zero before the PWRS bits will be set to standby mode. If a CPU read or fetch access to the flash bank/OTP initiated the process, the CPU will be stalled until the complete access completes (see ACTIVEWAIT bits). The STDBYWAIT bits specify the number of CPU clock cycles (0..511 SYSCLKOUT cycles) of delay. See the flash and OTP Power Mode section for more details.

Figure 1–5. Flash Standby to Active Wait Counter (FACTIVEWAIT) Register



Legend: R = Read access, -0 = value after reset

Bits	Name	Description
15–9	Reserved	
8–0	ACTIVEWAIT	Bank and Pump Standby To Active Wait Count: When the Bank and Pump modules are in standby mode and a write is performed to the PWR bits in the FPWR register (to change to a higher default power mode) or a CPU read or fetch access is performed to the flash bank, then a counter is initiated with the value specified in these register bits. The power mode lines for the Bank and Pump are set for active mode. The counter then counts down to zero before allowing any CPU access to proceed. If a CPU read or fetch access to the flash bank initiated the process, the CPU will be stalled until the access completes (see PAGEWAIT and RANDWAIT bits). The ACTIVEWAIT bits specify the number of CPU clock cycles (0..511 SYSCLKOUT cycles) of delay. Refer to the Flash and OTP Power Modes section for more details.

Figure 1–6. Flash Waitstate (FBANKWAIT) Register

15–12	11–8	7–4	3–0
Reserved	PAGEWAIT	Reserved	RANDWAIT
R-0	R/W-1	R-0	R/W-1

Legend: R = Read access, -0 = value after reset

Bits	Name	Description
15–12	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 banks. See Notes 1 and 2.
7–4	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 (0..15 SYSCLKOUT cycles) to the banks. See Notes 1 and 2.

Notes: 1) 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.

2) When enabling flash pipeline mode, you must set PAGEWAIT and RANDWAIT to a value greater than zero.

Figure 1–7. OTP Waitstate (FOTPWAIT) Register

15–5	4–0
Reserved	OTPWAIT
R-0	R/W-1

Legend: R = Read access, -0 = value after reset

Bits	Name	Description
15–5	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 (0..31 SYSCLKOUT cycles) to the OTP. See CPU Read Or Fetch Access From Flash/OTP section for more details. There is no PAGE mode in the OTP.

Chapter 2

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).

Topic **Page**

28x is a trademark of Texas Instruments.

2.1 Functional Description

The security module restricts the CPU access to on-chip memory. 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 access through JTAG is blocked (i.e., 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-bit data (eight 16-bit words) that is used to secure or unsecure the device.

The device is unsecured by executing the password match flow (PMF), described later in this chapter. Table 2–1 shows the levels of security.

Table 2–1. Security Levels

PMF Executed With Correct Password?	Operating Mode	Program Fetch Location	Security Description
No	Secure	Outside secure memory	Only fetches are allowed to secure memory
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

Passwords are stored in code security password locations (PWL) in flash/ROM memory (0x003F 7FF8 – 0x003F 7FFF). These locations store the password predetermined by the system designer.

In flash devices, the password can be changed anytime if the old password is known. In ROM devices, the password cannot be changed after the device is manufactured by Texas Instruments (TI).

If the PWL 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 PWL is required to bring the device into unsecure mode. If the PWL 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 after performing a clear routine on the flash. If a device is reset when the PWL is all zeros, the device cannot be debugged or reprogrammed. To summarize, a device with an erased flash array is unsecure. A device with a cleared flash array is secure.

User accessible registers (eight 16-bit words) that will be used to secure or unsecure the device are referred to as key registers. These registers are mapped in the memory space at addresses 0x0000 0AE0 – 0x0000 0AE7 and are EALLOW protected.

2.2 CSM Impact on Other On-Chip Resources

The CSM 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 2–2 and Table 2–3 show which on-chip resources are affected (or are not affected) by the CSM on the TMS320F2810 and TMS320F2812 devices. For other devices, see the device-specific data sheet.

Table 2–2. F2810™/F2812™ Resources Affected by the CSM

Address	Block
0x0000 8000 – 0x0000 8FFF	L0 SARAM (4K X 16)
0x0000 9000 – 0x0000 9FFF	L1 SARAM (4K X 16)
0x003D 7800 – 0x003D 7FFF	One-time programmable (OTP) (2K X 16)
0x003D 8000 – 0x003F 7FFF	Flash (128K X 16)

F2810 and F2812 are trademarks of Texas Instruments.

Table 2–3. F2810/F2812 Resources Not Affected by the CSM

Address	Block
0x0000 0000 – 0x0000 03FF	M0 SARAM (1K X 16)
0x0000 0400 – 0x0000 07FF	M1 SARAM (1K X16)
0x0000 0800 – 0x0000 0CFF	Peripheral Frame 0 (2K X 16)
0x0000 0D00 – 0x0000 0FFF	PIE Vector RAM (256 X 16)
0x0000 6000 – 0x0000 6FFF	Peripheral Frame 2 (4K X 16)
0x0000 7000 – 0x0000 7FFF	Peripheral Frame 1 (4K X 16)
0x003F 8000 – 0x003F 9FFF	H0 SARAM (8K X 16)
0x003F F000 – 0x003F FFFF	Boot ROM (4K X 16)

To summarize, it is possible to load code onto the unprotected on-chip program RAM shown in Table 2–3 via the JTAG connector without any impact from the CSM. The code can be debugged and the peripheral registers initialized, independent of whether the device is in secure or unsecure mode.

2.3 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 (or committed to ROM), 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 2–4. Code Security Module (CSM) Registers

Memory Address	Register Name	Reset Values	Register Description
KEY Registers – Accessible by the user			
0x0000 – 0AE0	KEY0†	0xFFFF	Low word of the 128-bit KEY register
0x0000 – 0AE1	KEY1†	0xFFFF	Second word of the 128-bit KEY register
0x0000 – 0AE2	KEY2†	0xFFFF	Third word of the 128-bit KEY register
. . .	. . .	. . .	. . .
0x0000 – 0AE7	KEY7†	0xFFFF	High word of the 128-bit KEY register
0x0000 – 0AEF	CSMSCR†	‡	CSM status and control register
PWL in Memory – Reserved for passwords only			
0x003F – 7FF8	PWL0	User defined	Low word of the 128-bit password
0x003F – 7FF9	PWL1	User defined	Second word of the 128-bit password
. . .	. . .	. . .	. . .
0x003F – 7FFF	PWL7	User defined	High word of the 128-bit password

† EALLOW protected

‡ The CSM status and control register (CSMSCR) contains one control bit, called FORCESEC, located in bit 15 and one status bit, called SECURE, located in bit 0. Setting the FORCESEC bit to a one resets the KEY registers and the internal logic of the CSM so that the password match flow must be executed to unsecure the device again. A device reset also resecures the device. A read of the FORCESEC bit always returns a zero. The SECURE bit is a read-only bit, which reflects the security state of the device. Writes to this bit have no effect. Reads of bits 14–1 in the CSMSCR are undetermined.

2.3.1 Environments That Require Security Unlocking

Following are the typical situations under which unsecuring may 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

  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 custom passwords) require passwords to be supplied to the flash utilities in order to enable programming.

- Custom environment defined by the application

  In addition to the above, access to secure memory contents may be required in situations such as:

  - Using the on-chip bootloader to program the flash
  - Executing code from external memory or 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.

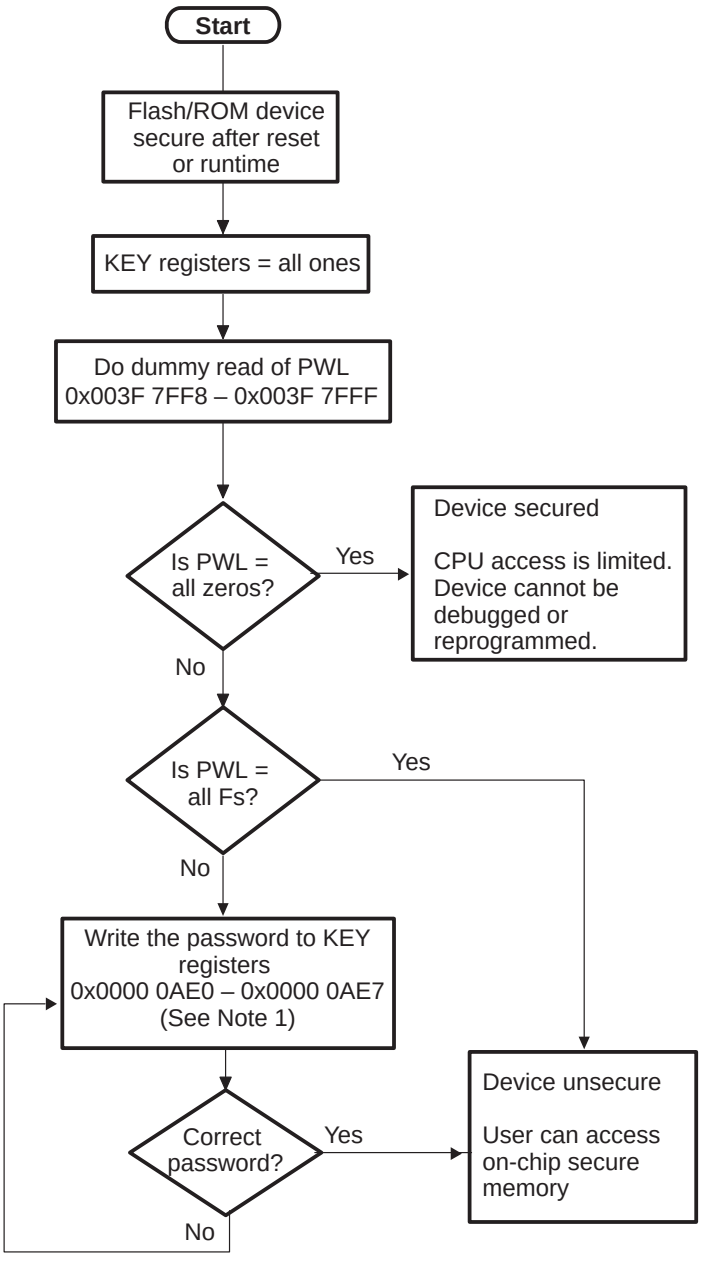
The unsecuring sequence is identical in all the above situations. This sequence is referred to as the *password match flow (PMF)* for simplicity. Figure 2–1 explains the sequence of operation that is required every time the user attempts to unsecure a device. A code example is listed for clarity.

2.3.2 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 2–1 shows how the PMF helps to initialize the security logic registers and disable security logic.

Figure 2–1. Password Match Flow (PMF)



1) The KEY registers are EALLOW protected.

2.3.3 Unsecuring Considerations for Devices With/Without Code Security

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

2.3.3.1 Case 1: Device With Code Security

A device with code security should have a predetermined password stored in the PWL (locations 0x003F 7FF8 – 0x003F 7FFF in memory). The following are steps to unsecure this device:

1) Perform a dummy read of the PWL.

2) Write the password into the KEY registers (locations 0x0000 0AE0 – 0x0000 0AE7 in memory).

3) If the password is correct, the device becomes unsecure; otherwise, it stays secure.

2.3.3.2 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 PWL. The following are steps to use this device:

1) Perform a dummy read of the PWL.

2) Secure memory is fully accessible immediately after this operation is completed.

Note: A dummy read operation must be performed prior to reading, writing, or programming secure memory, even though the device is not protected with a password.

2.3.4 C Code Example to Unsecure

```
volatile int *CSM = 0x000AE0;    //CSM register file
volatile int *PWL = 0x3F7FF8;    //Password location
volatile int tmp;
int i;
//Dummy read of password locations
//for (i = 0; i<8; i++) tmp = PWL++;
//If PWL = all ones, the code below is unnecessary to unsecure the CSM.
//Write PAWWSORD to KEY registers
asm (”   EALLOW”);               //KEY registers are EALLOW protected.
*CSM++ = PASSWORD0;
*CSM++ = PASSWORD1;
*CSM++ = PASSWORD2;
*CSM++ = PASSWORD3;
*CSM++ = PASSWORD4;
*CSM++ = PASSWORD5;
*CSM++ = PASSWORD6;
*CSM++ = PASSWORD7;
asm (”    EDIS”);
```

2.3.5 C Code Example to Resecure

```
volatile int *CSM = 0x000AE0;       //CSM register file
//Set FORCESEC bit
asm(”  EALLOW”);                     //CSMSCR register is EALLOW protected.
*CSM = 0x8000;
asm (”EDIS”);
```

2.4 DOs and DON'Ts to Protect Security Logic

2.4.1 DOs

- To keep the debug and code development phase simple, use the device in the unsecure mode; i.e., use 128 bits of all ones as PWL words (or use a password that is easy to remember). Use passwords after the development phase when the code is frozen.
- Recheck the passwords in PWL 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.

2.4.2 DON'Ts

- If code security is desired, do not embed the password in your application anywhere other than in the PWL or security may be compromised.
- Do not use 128 bits of all zeros as the password. This will automatically secure the device, regardless of the contents of the KEY register. The device will not be debuggable nor reprogrammable.
- Do not pull a reset after clearing the flash array but before erasing the array. This will leave zeros in the PWL that will automatically secure the device, regardless of the contents of the KEY register. The device will not be debuggable nor reprogrammable.

2.5 CSM Features – Summary

1) The flash is secured after a reset until the password match flow (PMF) is executed.

2) The standard way of running code out of the flash or ROM is to program the flash with the code (for ROM devices the program will be hardcoded at device fabrication) and powering up the DSP in microcomputer mode. Since instruction fetches are always allowed from secure memory, regardless of the state of the CSM, the code will function correctly even without executing the PMF.

3) Secure memory cannot be modified 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 by the debugger (i.e., Code Composer Studio™) at any time that the device is secured.

6) Complete access to secure memory from the CPU code and the debugger is granted while the device is unsecured.

Chapter 3

Clocking

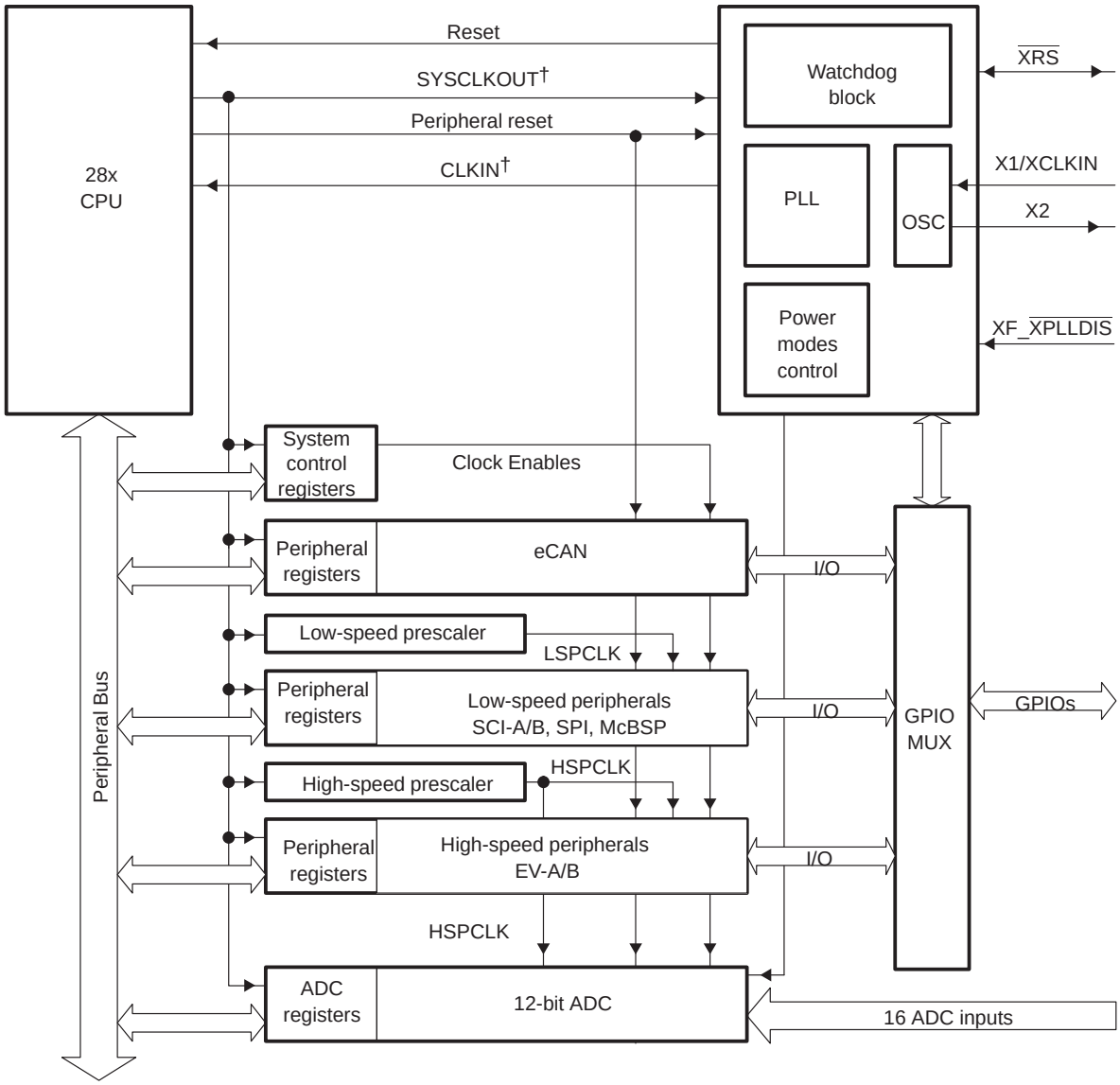
This section describes the F2810 and F2812 oscillator, PLL and clocking mechanisms, the watchdog function, and the low-power modes.

Topic **Page**

3.1 Clocking and System Control

Figure 3–1 shows the various clock and reset domains in the F2810 and F2812 devices.

Figure 3–1. Clock and Reset Domains



† CLKIN is the clock into the CPU. It is passed out of the CPU as SYSCLKOUT (that is, CLKIN = SYSCLKOUT).

The PLL, clocking, watchdog and low-power modes, are controlled by the registers listed in Table 3–1.

Table 3–1. PLL, Clocking, Watchdog, and Low-Power Mode Registers†

Name	Address	Size (x16)	Description
Reserved	0x0000 7010 0x0000 7019	10	
HISPCP	0x0000 701A	1	High-Speed Peripheral Clock Prescaler Register for HSPCLK clock
LOSPCP	0x0000 701B	1	Low-Speed Peripheral Clock Prescaler Register for HSPCLK clock
PCLKCR	0x0000 701C	1	Peripheral Clock Control Register
Reserved	0x0000 701D	1	
LPMCR0	0x0000 701E	1	Low Power Mode Control Register 0
LPMCR1	0x0000 701F	1	Low Power Mode Control Register 1
Reserved	0x0000 7020	1	
PLLCR	0x0000 7021	1	PLL Control Register‡
SCSR	0x0000 7022	1	System Control & Status Register
WDCNTR	0x0000 7023	1	Watchdog Counter Register
Reserved	0x0000 7024	1	
WDKEY	0x0000 7025	1	Watchdog Reset Key Register
Reserved	0x0000 7026 0x0000 7028	3	
WDCR	0x0000 7029	1	Watchdog Control Register
Reserved	0x0000 702A 0x0000 702F	6	

† All of the registers in this table can be accessed only by executing the EALLOW instruction.
‡ The PLL control register (PLLCR) is reset to a known state by the $\overline{\text{XRS}}$ signal only.

The PCLKCR register enables/disables clocks to the various peripheral modules in the F2810 and F2812 devices. Figure 3–2 lists the bit descriptions of the PCLKCR register.

Figure 3–2. Peripheral Clock Control (PCLKCR) Register †

15	14	13	12	11	10	9	8
Reserved	ECANENCLK	Reserved	MCBSPENCLK	SCIBENCLK	SCIAENCLK	Reserved	SPIENCLK
R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0

7 – 4	3	2	1	0
Reserved	ADCENCLK	Reserved	EVBENCLK	EVAENCLK
R-0	R/W-0	R-0	R/W-0	R/W-0

Legend: R = Read access, -0 = value after reset

Bits	Name	Description
15	Reserved	Reserved
14	ECANENCLK	If this bit is set, it enables the system clock within the CAN peripheral. For low power operation, this bit is set to zero by the user or by reset.
13	Reserved	Reserved
12	MCBSPENCLK	If this bit is set, it enables the low-speed clock (LSPCLK) within the McBSP peripheral. For low power operation, this bit is set to zero by the user or by reset.
11	SCIBENCLK	If this bit is set, it enables the low-speed clock (LSPCLK) within the SCI-B peripheral. For low power operation, this bit is set to zero by the user or by reset.
10	SCIAENCLK	If this bit is set, it enables the low-speed clock (LSPCLK) within the SCI-A peripheral. For low power operation, this bit is set to zero by the user or by reset.
9	Reserved	Reserved
8	SPIAENCLK	If this bit is set, it enables the low-speed clock (LSPCLK) within the SPI peripheral. For low power operation, this bit is set to zero by the user or by reset.
7–4	Reserved	
3	ADCENCLK	If this bit is set, it enables the high-speed clock (HSPCLK) within the ADC peripheral. For low power operation, this bit is set to zero by the user or by reset.
2	Reserved	Reserved

† If a peripheral block is not used, then the clock to that peripheral can be turned off to minimize power consumption.

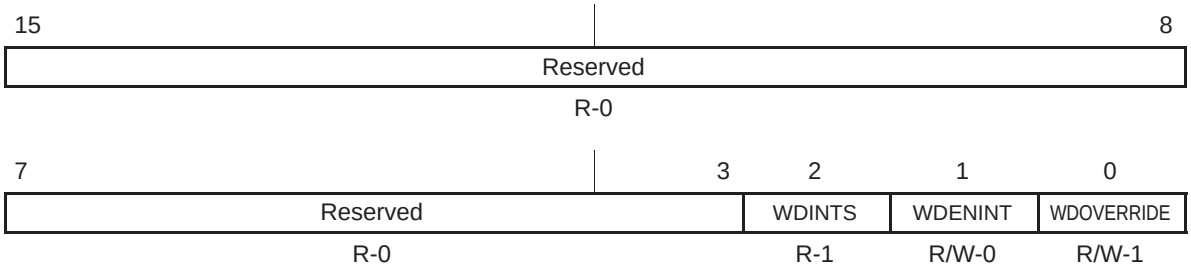
Figure 3–2.Peripheral Clock Control (PCLKCR) Register (Continued)

Bits	Name	Description
1	EVBENCLK	If this bit is set, it enables the high-speed clock (HSPCLK) within the EV-B peripheral. For low power operation, this bit is set to zero by the user or by reset.
0	EVAENCLK	If this bit is set, it enables the high-speed clock (HSPCLK) within the EV-A peripheral. For low power operation, this bit is set to zero by the user or by reset.

† If a peripheral block is not used, then the clock to that peripheral can be turned off to minimize power consumption.

The system control and status register contains the watchdog override bit and the watchdog interrupt enable/disable bit. Figure 3–3 describes the bit functions of the SCSR register.

Figure 3–3. System Control & Status (SCSR) Register

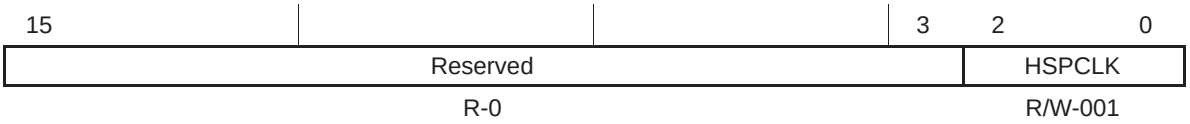


Legend: R = Read access, -0 = value after reset

Bits	Name	Description
15–3	Reserved	
2	WDINTS	Watchdog interrupt status bit. This bit reflects the current state of the $\overline{\text{WDINT}}$ signal from the watchdog block.
1	WDENINT	If this bit is set to 1, the watchdog reset ($\overline{\text{WDRST}}$) output signal is disabled and the watchdog interrupt ($\overline{\text{WDINT}}$) output signal is enabled. If this bit is zero, then the $\overline{\text{WDRST}}$ output signal is enabled and the $\overline{\text{WDINT}}$ output signal is disabled. This is the default state on reset ($\overline{\text{XRS}}$).
0	WDOVERRIDE	If this bit is set to 1, the user is allowed to change the state of the Watchdog disable (WDDIS) bit in the Watchdog Control (WDCR) register (see Watchdog Block section of this data sheet). If the WDOVERRIDE bit is cleared, by writing a 1 the WDDIS bit cannot be modified by the user. Writing a 0 will have no effect. If this bit is cleared, then it will remain in this state until a reset occurs. The current state of this bit is readable by the user.

The HISPCP and LOSPCP registers are used to configure the high- and low-speed peripheral clocks, respectively. See Figure 3–4 for the HISPCP bit definitions and Figure 3–5 for the LOSPCP bit definitions.

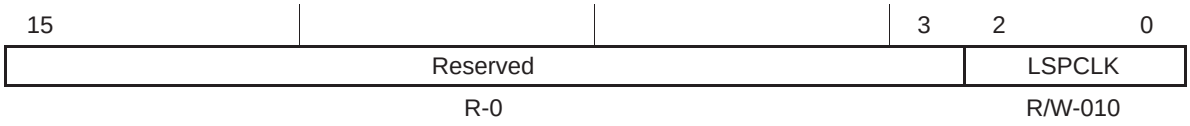
Figure 3–4. High-Speed Peripheral Clock Prescaler (HISPCP) Register



Legend: R = Read access, -0 = value after reset

Bits	Name	Description
15–3	Reserved	
2–0	HSPCLK	These bits configure the high-speed peripheral clock (HSPCLK) rate relative to 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

Figure 3–5. Low-Speed Peripheral Clock Prescaler (LOSPCP) Register

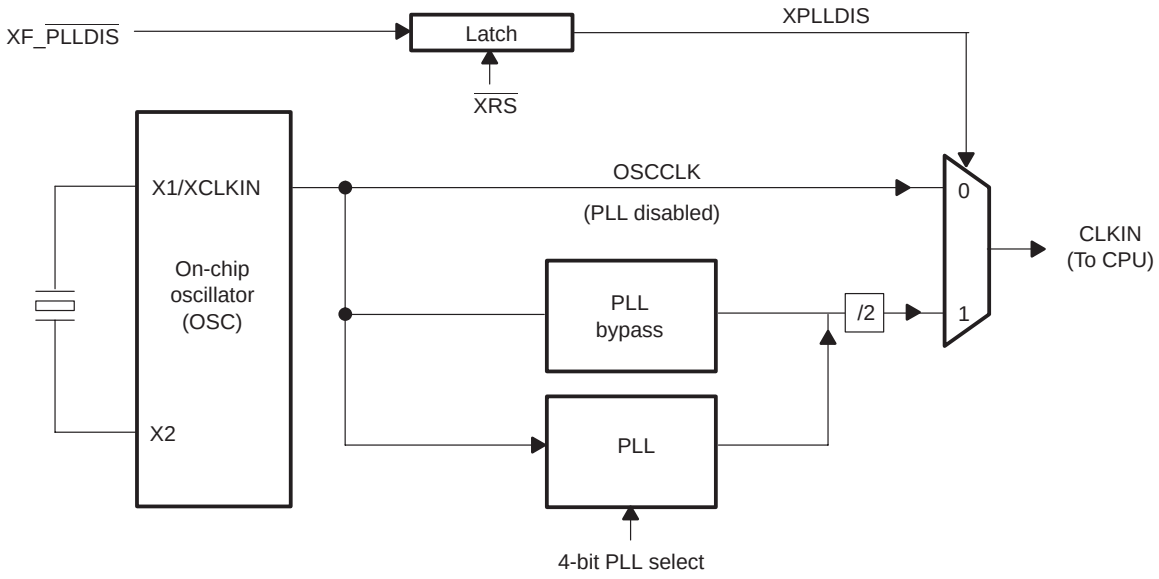


Bits	Name	Description
15–3	Reserved	
2–0	LSPCLK	These bits configure the low-speed peripheral clock (LSPCLK) rate relative to SYSCLKOUT: 000 low speed clock = SYSCLKOUT/1 001 low speed clock= SYSCLKOUT/2 010 low speed clock= SYSCLKOUT/4 (reset default) 011 low speed clock= SYSCLKOUT/6 100 low speed clock= SYSCLKOUT/8 101 low speed clock= SYSCLKOUT/10 110 low speed clock= SYSCLKOUT/12 111 low speed clock= SYSCLKOUT/14

3.2 OSC and PLL Block

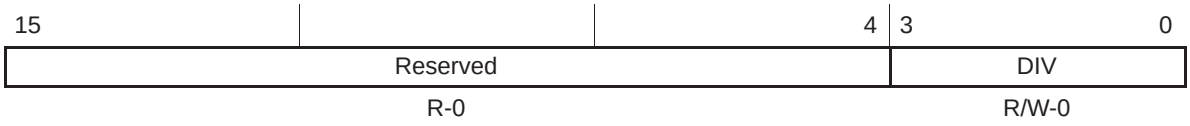
Figure 3–6 shows the OSC and PLL block on the F2810 and F2812.

Figure 3–6. OSC and PLL Block



The OSC circuit enables a crystal to be attached to the F2810 and F2812 devices using the X1/XCLKIN and X2 pins. If a crystal is not used, then an external oscillator can be directly connected to the X1/XCLKIN pin and the X2 pin is left unconnected.

Figure 3–7. PLLCR Register



Bits	Name	Description
15–4	Reserved	
3–0	DIV	The DIV field controls whether the PLL is bypassed or not, and sets the PLL clocking ration when not bypassed.
		0000 CLKIN = OSCCLK/2 (PLL bypass)
		0001 CLKIN (OSCCLK * 1.0)/2
		0010 CLKIN = (OSCCLK * 2.0)/2
		0011 CLKIN = (OSCCLK * 3.0)/2
		0100 CLKIN = (OSCCLK * 4.0)/2
		0101 CLKIN = (OSCCLK * 5.0)/2
		0110 CLKIN = (OSCCLK * 6.0)/2
		0111 CLKIN = (OSCCLK * 7.0)/2
		1000 CLKIN = (OSCCLK * 8.0)/2
		1001 CLKIN = (OSCCLK * 9.0)/2
		1010 CLKIN = (OSCCLK * 10.0)/2
		1011 reserved
		1100 reserved
		1101 reserved
		1110 reserved
		1111 reserved

† The PLLCR register is reset to a known state by the $\overline{\text{XRS}}$ reset line. If a reset is issued by the debugger, the PLL clocking ratio is not changed.

3.2.1 PLL-Based Clock Module

The F2810 and F2812 have an on-chip, PLL-based clock module. This module provides all the necessary clocking signals for the device, as well as control for low-power mode entry. The PLL has a 4-bit ratio control to select different CPU clock rates.

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

- Crystal-operation
  This mode allows the use of an external crystal/resonator to provide the time base to the device.

- External clock source operation
  This mode allows the internal oscillator to be bypassed. The device clocks are generated from an external clock source input on the XTAL1/CLKIN pin. In this case, an external oscillator clock is connected to the XTAL1/CLKIN pin.

3.2.2 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 that will ensure start-up and stability over the entire operating range.

3.3 Low-Power Modes Block

The low-power modes on the F2810 and F2812 are similar to the 240x devices. Table 3–2 summarizes the various modes.

Table 3–2. F2810 and F2812 Low-Power Modes

Mode	IDLES	LPM(1:0)	OSCCLK	CLKIN	SYSCLKOUT	Exit†
Normal	low	X,X	on	on	on	–
IDLE	high	0,0	on	on	on‡	$\overline{\text{XRS}}$, $\overline{\text{WDINT}}$, Any Enabled Interrupt, XNMI
STANDBY	high	0,1	on (watchdog still running)	off	off	$\overline{\text{XRS}}$, $\overline{\text{WDINT}}$, XINT1, XNMI, $\overline{\text{T1/2/3/4CTRIP}}$, $\overline{\text{C1/2/3/4/5/6TRIP}}$, SCIRXDA, SCIRXDB, CANRX, Debugger§
HALT	high	1,X	off (oscillator and PLL turned off, watchdog not functional)	off	off	$\overline{\text{XRS}}$, XNMI, Debugger§

† The Exit column lists which signals or under what conditions the low power mode will be exited. This signal must be kept low long enough for an interrupt to be recognized by the device. Otherwise the IDLE mode will not be exited and the device will go back into the indicated low power mode.
‡ The IDLE mode on the 28x behaves differently than on the 24x/240x. On the 28x, the clock output from the core (SYSCLKOUT) is still functional while on the 24x/240x the clock is turned off.
§ On the 28x, the JTAG port can still function even if the clock to the CPU (CLKIN) is turned off.

The various low-power modes operate as follows:

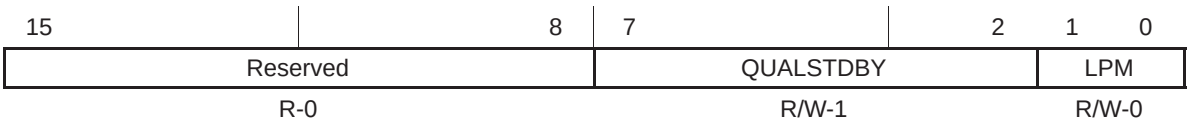
IDLE Mode: This mode is exited by any enabled interrupt or an NMI that is recognized by the processor. The LPM block performs no tasks during this mode as long as the LPMCR[1:0] bits are set to 0,0.

HALT Mode: Only the $\overline{\text{XRS}}$ and XNMI external signals can wake the device from HALT mode. The XNMI input to the CPU has an enable/disable bit in the XMNICR register. Therefore, it is safe to use the XNMI signal for this function.

STANDBY Mode: All other signals (including XNMI) wake the device from STANDBY mode if selected by the LPMCR1 register. You must select which signal(s) will wake the device. The selected signal(s) are also qualified by the OSCCLK before waking the device. The number of OSCCLKs is specified in the LPMCR0 register.

The low-power modes are controlled by the LPMCR0 register (see Figure 3–8) and the LPMCR1 register (see Figure 3–9).

Figure 3–8. Low Power Mode Control 0 (LPMCR0) Register



Bits	Name	Type	Reset†	Description
15–8	Reserved	R = 0	0:0	
7–2	QUALSTDBY	R/W	1:1	Select number of OSCCLK clock cycles to qualify the selected inputs when waking the LPM from STANDBY mode: 000000 = 2 OSCCLKs 000001 = 3 OSCCLKs . 111111 = 65 OSCCLKs
1–0	LPM‡	R/W	0,0	These bits set the low power mode for the device.

† These bits are cleared by a reset ($\overline{\text{XRS}}$).

‡ The low power mode bits (LPM) are only valid when the IDLE instruction is executed. Therefore, the user must set the LPM bits to the appropriate mode before executing the IDLE instruction.

Figure 3–9. Low Power Mode Control 1 (LPMCR1) Register

15	14	13	12	11	10	9	8
CANRX	SCIRXB	SCIRXA	C6TRIP	C5TRIP	C4TRIP	C3TRIP	C2TRIP
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
C1TRIP	T4CTRIP	T3CTRIP	T2CTRIP	T1CTRIP	WDINT	XNMI	XINT1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

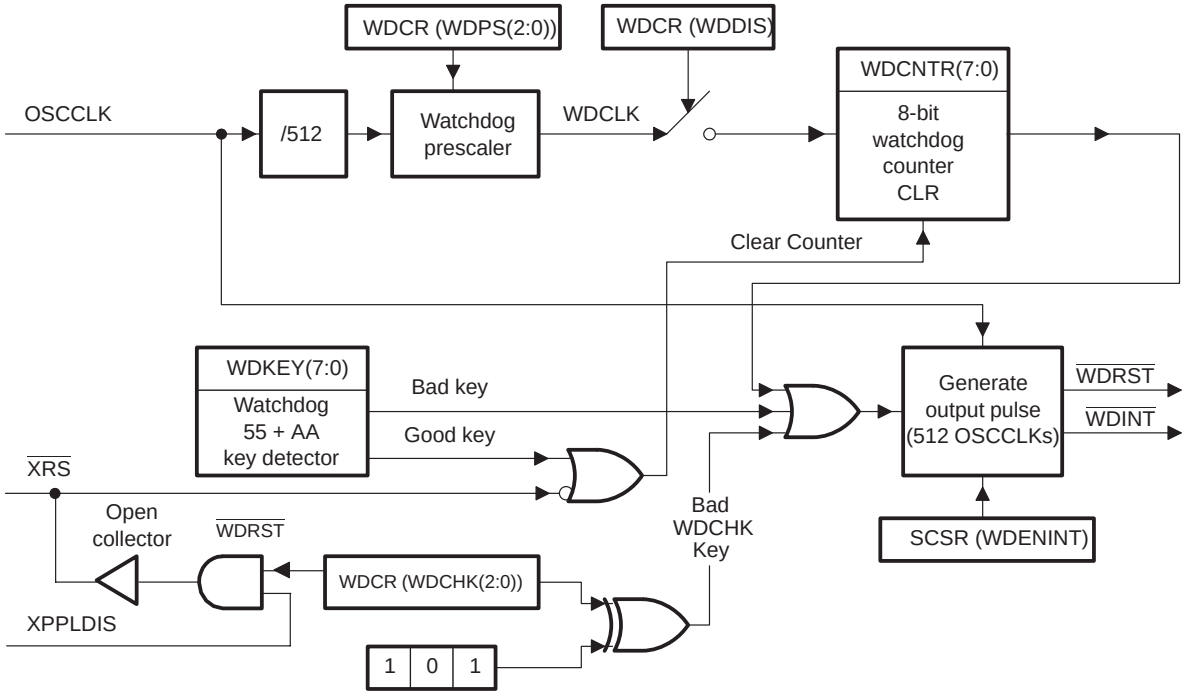
Bits	Name	Description
0	XINT1	If the respective bit is set to 1, it will enable the selected signal to wake the device from STANDBY mode. If the bit is cleared, the signal will have no effect.
1	XNMI	
2	$\overline{\text{WDINT}}$	
3	T1CTRIP	
4	T2CTRIP	
5	T3CTRIP	
6	T4CTRIP	
7	C1TRIP	
8	C2TRIP	
9	C3TRIP	
10	C4TRIP	
11	C5TRIP	
12	C6TRIP	
13	SCIRXA	
14	SCIRXB	
15	CANRX	

† These bits are cleared by a reset ($\overline{\text{XRS}}$).

3.4 Watchdog Block

The watchdog block on the F2810 and F2812 is identical to the one used on the 240x devices. The watchdog module generates an output pulse, 512 oscillator clocks wide (OSCCLK), whenever the 8-bit watchdog up counter has reached its maximum value. To prevent this, the user disables the counter or the software must periodically write a 0x55 + 0xAA sequence into the watchdog key register which will reset the watchdog counter. Figure 3–10 shows the various functional blocks within the watchdog module.

Figure 3–10. Watchdog Module



NOTE A: The $\overline{\text{WDRST}}$ signal is driven low for 512 OSCCLK cycles (similarly for the $\overline{\text{WDINT}}$ signal if enabled).

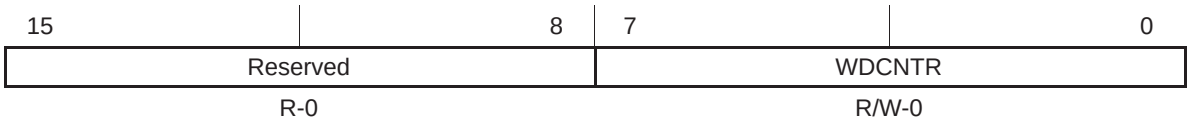
The $\overline{\text{WDINT}}$ signal enables the watchdog to be used as a wakeup from IDLE/ STANDBY mode timer.

In STANDBY mode, all peripherals are turned off on the device. The only peripheral that remains functional is the watchdog. The WATCHDOG module will run off the PLL clock or the oscillator clock. The $\overline{\text{WDINT}}$ signal is fed to the LPM block so that it can wake the device from STANDBY (if enabled). See Low-Power Modes Block section of this data sheet for more details.

In IDLE mode, the $\overline{\text{WDINT}}$ signal can generate an interrupt to the CPU, (the WAKEINT interrupt in the PIE), to take the CPU out of IDLE mode.

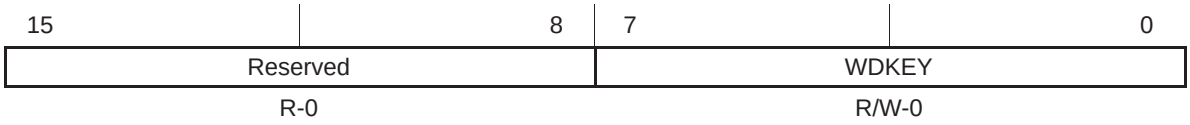
In HALT mode, this feature cannot be used because the oscillator (and PLL) are turned off and hence so is the WATCHDOG.

Figure 3–11. Watchdog Counter (WDCNTR) Register



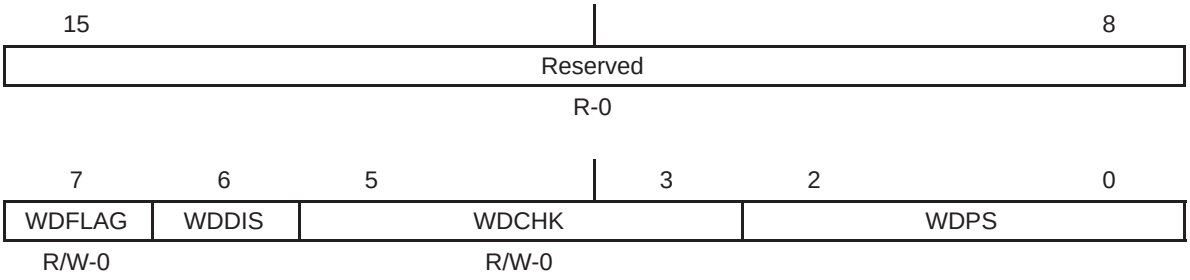
Bits	Name	Description
15–8	Reserved	
7–0	WDCNTR	These bits contain the current value of the WD counter. The 8-bit counter continually increments at the WDCLK rate. If the counter overflows, then the watchdog initiates a reset. If the WDKEY register is written with a valid combination, then the counter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

Figure 3–12. Watchdog Reset Key (WDKEY) Register



Bits	Name	Type	Reset	Description
15–8	Reserved	R = 0	0:0	
7–0	WDKEY	R/W	0:0	Writing 0x55 followed by 0xAA will cause the WDCNTR bits to be cleared. Writing any other value will cause an immediate watchdog reset to be generated. Reads return the value of the WDCR register.

Figure 3–13. Watchdog Control (WDCR) Register



Bits	Name	Description
15–8	Reserved	
7	WDFLAG	Watchdog reset status flag bit. This bit, if set, indicates a watchdog reset ($\overline{\text{WDRST}}$) generated the reset condition. If 0, then it was an external device or power-up reset condition. This bit remains latched until the user writes a 1 to clear the condition. Writes of 0 will be ignored.
6	WDDIS	Writing a 1 to this bit will disable the watchdog module. Writing a 0 will enable the module. This bit can only be modified if the WDOVERRIDE bit in the SCSR2 register is set to 1. On reset, the watchdog module is enabled.
5–3	WDCHK(2–0)	The user must ALWAYS write 1,0,1 to these bits whenever a write to this register is performed. Writing any other value will cause an immediate reset to the core (if WD enabled).
2–0	WDPS(2–0)	These bits configure the watchdog counter clock (WDCLK) rate relative to OSCCLK/512: 000 WDCLK = OSCCLK/512/1 001 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 will only be set if a rising edge on $\overline{\text{WDRST}}$ signal is detected (after synch and a 4 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 will remain at 0. In a typical application, the $\overline{\text{WDRST}}$ signal will connect to the $\overline{\text{XRS}}$ input. Hence to distinguish be-

tween a watchdog reset and an external device reset, an external reset must be longer in duration then the watchdog pulse.

3.4.1 Emulation Considerations

The watchdog module behaves as follows under various debug conditions:

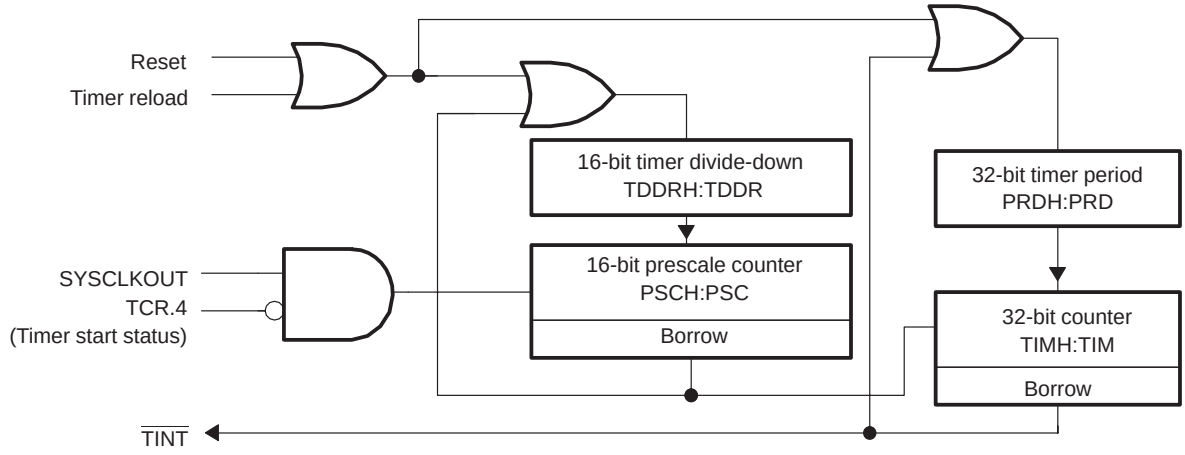
CPU Suspended:	When the CPU is suspended, the watchdog clock (WDCLK) is suspended.
Run-Free Mode:	When the CPU is placed in run-free mode, then the watchdog module resumes operation as normal.
Real-Time Single-Step Mode:	When the CPU is in real-time single-step mode, the watchdog clock (WDCLK) is suspended.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

3.5 32-Bit CPU Timers 0/1/2

This section describes the three 32-bit CPU-timers on the F2810 and F2812 devices (TIMER0/1/2).

CPU-Timers 1 and 2 are Reserved for the Real-Time OS (such as DSP-BIOS).† CPU-Timer 0 can be used in user applications.

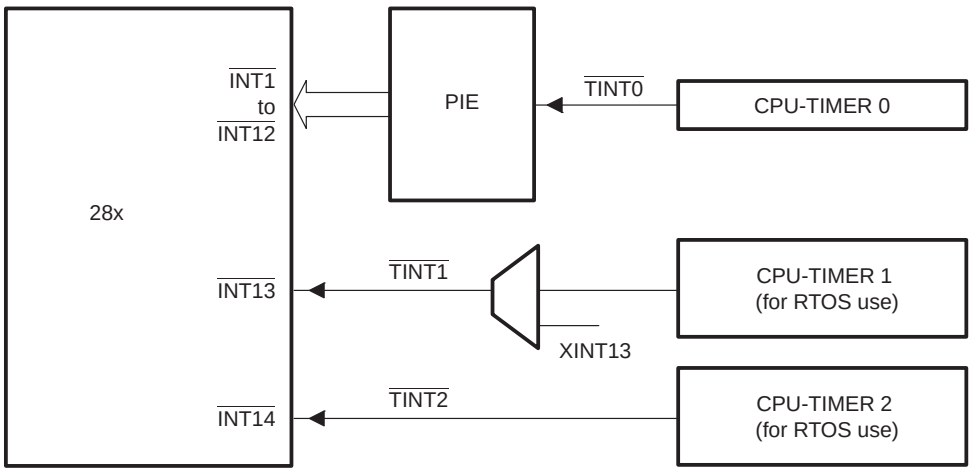
Figure 3–14. CPU-Timers



NOTE A: The CPU-Timers are different from the general-purpose (GP) timers that are present in the Event Manager modules (EVA, EVB).

In the F2810 and F2812 devices, the timer interrupt signals ($\overline{\text{TINT0}}$, $\overline{\text{TINT1}}$, $\overline{\text{TINT2}}$) are connected as shown in Figure 3–15.

Figure 3–15. CPU-Timer Interrupts Signals and Output Signal



NOTES: 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 timer is as follows: The 32-bit counter register TIMH:TIM is loaded with the value in the period register PRDH:PRD. The counter register, decrements at the SYSCLKOUT rate of the 28x. When the counter reaches 0, a timer interrupt output signal generates an interrupt pulse. The registers listed in Table 3–3 are used to configure the timers.

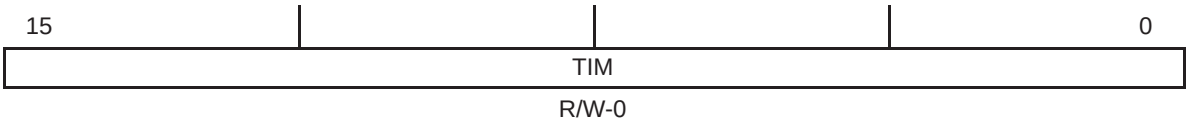
Table 3–3. CPU-Timers 0, 1, 2 Configuration and Control Registers

Name	Address	Size (x16)	Description
TIMER0TIM	0x0000 0C00	1	CPU-Timer 0, Counter Register
TIMER0TIMH	0x0000 0C01	1	CPU-Timer 0, Counter Register High
TIMER0PRD	0x0000 0C02	1	CPU-Timer 0, Period Register
TIMER0PRDH	0x0000 0C03	1	CPU-Timer 0, Period Register High
TIMER0TCR	0x0000 0C04	1	CPU-Timer 0, Control Register
Reserved	0x0000 0C05	1	
TIMER0TPR	0x0000 0C06	1	CPU-Timer 0, Prescale Register
TIMER0TPRH	0x0000 0C07	1	CPU-Timer 0, Prescale Register High
TIMER1TIM	0x0000 0C08	1	CPU-Timer 1, Counter Register
TIMER1TIMH	0x0000 0C09	1	CPU-Timer 1, Counter Register High
TIMER1PRD	0x0000 0C0A	1	CPU-Timer 1, Period Register
TIMER1PRDH	0x0000 0C0B	1	CPU-Timer 1, Period Register High
TIMER1TCR	0x0000 0C0C	1	CPU-Timer 1, Control Register
Reserved	0x0000 0C0D	1	
TIMER1TPR	0x0000 0C0E	1	CPU-Timer 1, Prescale Register
TIMER1TPRH	0x0000 0C0F	1	CPU-Timer 1, Prescale Register High
TIMER2TIM	0x0000 0C10	1	CPU-Timer 2, Counter Register
TIMER2TIMH	0x0000 0C11	1	CPU-Timer 2, Counter Register High
TIMER2PRD	0x0000 0C12	1	CPU-Timer 2, Period Register
TIMER2PRDH	0x0000 0C13	1	CPU-Timer 2, Period Register High
TIMER2TCR	0x0000 0C14	1	CPU-Timer 2, Control Register
Reserved	0x0000 0C15	1	

Table 3–3. CPU-Timers 0, 1, 2 Configuration and Control Registers (Continued)

Name	Address	Size (x16)	Description
TIMER2TPR	0x0000 0C16	1	CPU-Timer 2, Prescale Register
TIMER2TPRH	0x0000 0C17	1	CPU-Timer 2, Prescale Register High
Reserved	0x0000 0C18 0x0000 0C3F	40	

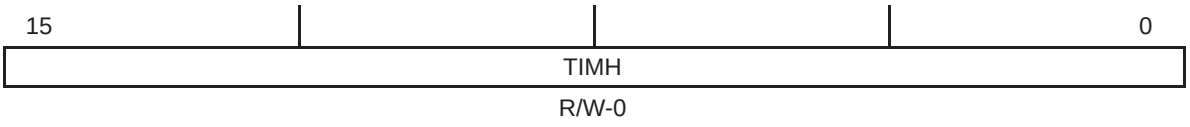
Figure 3–16. TIMERxTIM Register†



Bits	Name	Description
15–0	TIM	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 ($\overline{\text{TINT}}$) signal is generated.

† x = 0, 1, or 2

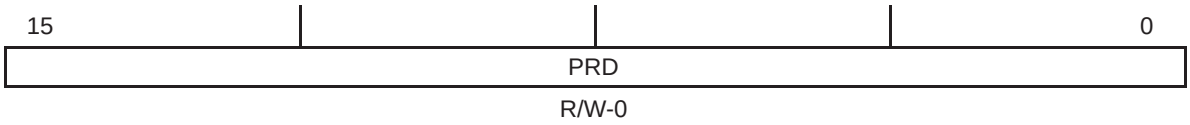
Figure 3–17. TIMERxTIMH Register†



Bits	Name	Description
15–0	TIMH	See description for TIMERxTIM.

† x = 0, 1, or 2

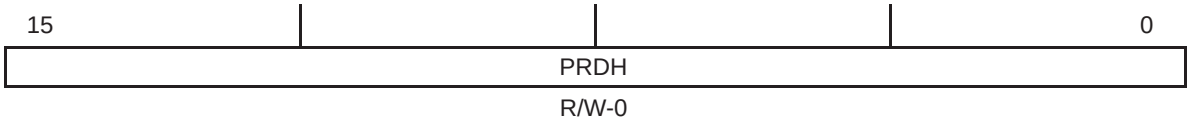
Figure 3–18. TIMERxPRD Register†



Bits	Name	Description
15–0	PRD	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).

† x = 0, 1, or 2

Figure 3–19. TIMERxPRDH Register †



Bits	Name	Description
15–0	PRDH	See description for TIMERxPRD

† x = 0, 1, or 2

Figure 3–20. TIMERxTCR Register†

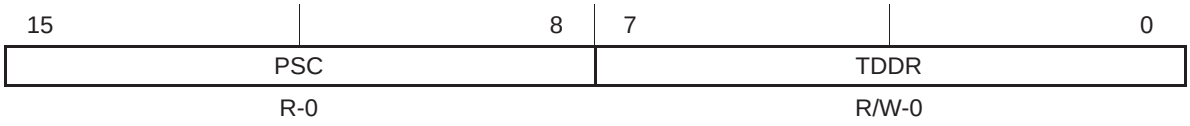
15	14	13 12	11	10	9 8
TIF	TIE	Reserved	FREE	SOFT	Reserved
R/W-0	R/W-0	R-0	R/W-0	R/W-0	R-0

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

Bits	Name	Description
15	TIF	Timer Interrupt Flag. This flag gets set when the timer decrements to zero. This bit can be cleared by software writing a 1, but it can only be set by the timer reaching zero. Writing a 1 to this bit will clear it, writing a zero has no effect.
14	TIE	Timer Interrupt Enable. If the timer decrements to zero, and this bit is set, the timer will assert its interrupt request.
13–12	Reserved	Reserved
11	FREE	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.
10	SOFT	**FREE SOFT 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 Note: That in the SOFT STOP mode, the timer will generate an interrupt before shutting down (since reaching 0 is the interrupt causing condition).
9–6	Reserved	Reserved
5	TRB	Timer Reload bit. 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). The TRB bit is always read as zero.
4	TSS	Timer stop status bit. TSS is a 1-bit flag that stops or starts the timer. To stop the timer, set TSS to 1. To start or restart the timer, set TSS to 0. At reset, TSS is cleared to 0 and the timer immediately starts.
3–0	Reserved	Reserved

† x = 0, 1, or 2

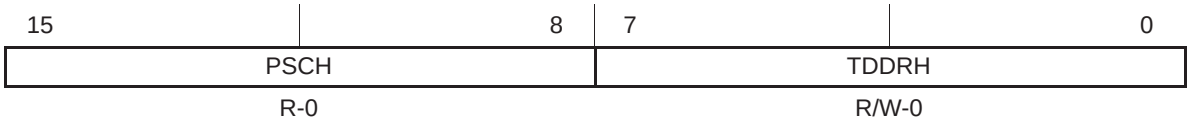
Figure 3–21. TIMERxTPR Register†



Bits	Name	Description
15–8	PSC	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	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.

† x = 0, 1, or 2

Table 3–4. TIMERxTPRH Register Bit Definitions†



Bits	Name	Description
15–8	PSCH	See description of TIMERxTPR.
7–0	TDDRH	See description of TIMERxTPR.

† x = 0, 1, or 2

Chapter 4

General-Purpose Input/Output (GPIO)

The GPIO MUX registers are used to select the operation of shared pins on the F2810™ and F2812™ devices. The pins can be individually selected to operate as Digital I/O or connected to Peripheral I/O signals (via the GPxMUX registers). If selected for Digital I/O mode, registers are provided to configure the pin direction (via the GPxDIR registers) and to qualify the input signal to remove unwanted noise (via the GPxQUAL) registers).

Topic **Page**

F2810 and F2812 are trademarks of Texas Instruments.

4.1 GPIO MUX

Table 4–1 lists the GPIO Mux Registers.

Table 4–1. GPIO Mux Registers†‡

Name	Address	Size (x16)	Register Description
GPAMUX	0x0000 70C0	1	GPIO A MUX Control Register
GPADIR	0x0000 70C1	1	GPIO A Direction Control Register
GPAQUAL	0x0000 70C2	1	GPIO A Input Qualification Control Register
Reserved	0x0000 70C3	1	
GPBMUX	0x0000 70C4	1	GPIO B MUX Control Register
GPBDIR	0x0000 70C5	1	GPIO B Direction Control Register
GPBQUAL	0x0000 70C6	1	GPIO B Input Qualification Control Register
Reserved	0x0000 70C7 – 0x0000 70CB	5	
GPDMUX	0x0000 70CC	1	GPIO D MUX Control Register
GPDDIR	0x0000 70CD	1	GPIO D Direction Control Register
GPDQUAL	0x0000 70CE	1	GPIO D Input Qualification Control Register
Reserved	0x0000 70CF	1	
GPEMUX	0x0000 70D0	1	GPIO E MUX Control Register
GPEDIR	0x0000 70D1	1	GPIO E Direction Control Register
GPEQUAL	0x0000 70D2	1	GPIO E Input Qualification Control Register
Reserved	0x0000 70D3	1	
GPFMUX	0x0000 70D4	1	GPIO F MUX Control Register
GPFDIR	0x0000 70D5	1	GPIO F Direction Control Register
Reserved	0x0000 70D6 – 0x0000 70D7	2	
GPGMUX	0x0000 70D8	1	GPIO F MUX Control Register

† For reserved locations, read values are undefined and writes have no effect.
‡ These registers are EALLOW protected. This prevents spurious writes from overwriting the contents and corrupting the system.

Table 4–1. GPIO Mux Registers†‡ (Continued)

Name	Address	Size (x16)	Register Description
GPGDIR	0x0000 70D9	1	GPIO F Direction Control Register
Reserved	0x0000 70DA – 0x0000 70DF	6	

† For reserved locations, read values are undefined and writes have no effect.
‡ These registers are EALLOW protected. This prevents spurious writes from overwriting the contents and corrupting the system.

If configured for Digital I/O mode, additional registers are provided for setting individual I/O signals (via the GPxSET registers), for clearing individual I/O signals (via the GPxCLEAR registers), for toggling individual I/O signals (via the GPxTOGGLE registers), or for reading/writing to the individual I/O signals (via the GPxDAT registers). Table 4–2 lists the GPIO Data Registers.

Table 4–2. GPIO Data Registers†‡

Name	Address	Size (x16)	Register Description
GPADAT	0x0000 70E0	1	GPIO A Data Register
GPASET	0x0000 70E1	1	GPIO A Set Register
GPACLEAR	0x0000 70E2	1	GPIO A Clear Register
GPATOGGLE	0x0000 70E3	1	GPIO A Toggle Register
GPBDAT	0x0000 70E4	1	GPIO B Data Register
GPBSET	0x0000 70E5	1	GPIO B Set Register
GPBCLEAR	0x0000 70E6	1	GPIO B Clear Register
GPBTOGGLE	0x0000 70E7	1	GPIO B Toggle Register
Reserved	0x0000 70E8 – 0x0000 70EB	4	
GPDDAT	0x0000 70EC	1	GPIO D Data Register
GPDSET	0x0000 70ED	1	GPIO D Set Register
GPDCLEAR	0x0000 70EE	1	GPIO D Clear Register
GPDTOGGLE	0x0000 70EF	1	GPIO D Toggle Register

† For reserved locations, read values are undefined and writes have no effect.
‡ These registers are NOT EALLOW protected and will typically be accessed regularly by the user.

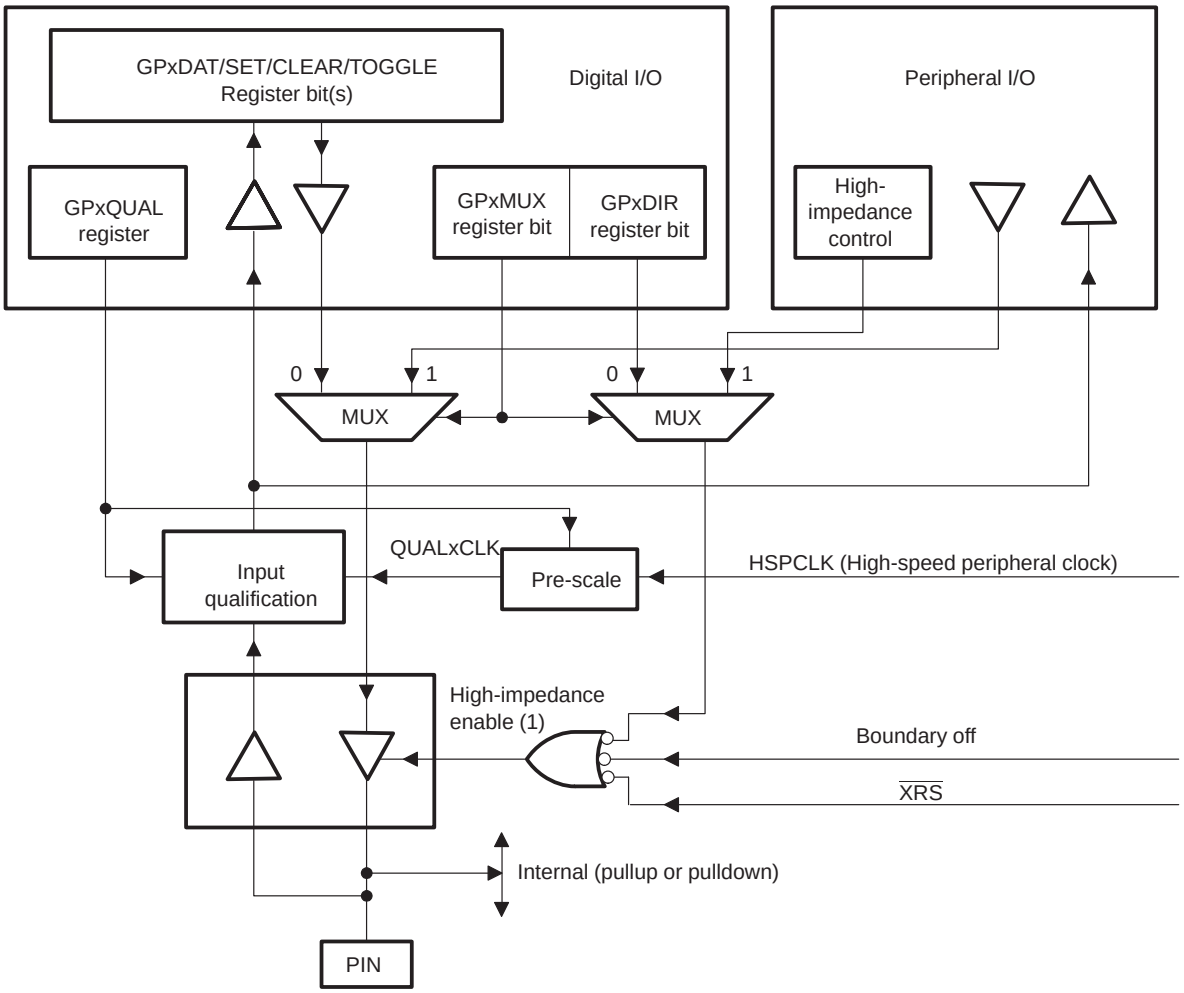
Table 4–2. GPIO Data Registers†‡ (Continued)

Name	Address	Size (x16)	Register Description
GPEDAT	0x0000 70F0	1	GPIO E Data Register
GPESET	0x0000 70F1	1	GPIO E Set Register
GPECLEAR	0x0000 70F2	1	GPIO E Clear Register
GPETOGGLE	0x0000 70F3	1	GPIO E Toggle Register
GPFDAT	0x0000 70F4	1	GPIO F Data Register
GPFSET	0x0000 70F5	1	GPIO F Set Register
GPFCLEAR	0x0000 70F6	1	GPIO F Clear Register
GPFTOGGLE	0x0000 70F7	1	GPIO F Toggle Register
GPGDAT	0x0000 70F8	1	GPIO G Data Register
GPGSET	0x0000 70F9	1	GPIO G Set Register
GPGCLEAR	0x0000 70FA	1	GPIO G Clear Register
GPGTOGGLE	0x0000 70FB	1	GPIO G Toggle Register
Reserved	0x0000 70FC – 0x0000 70FF	4	

† For reserved locations, read values are undefined and writes have no effect.
‡ These registers are NOT EALLOW protected and will typically be accessed regularly by the user.

Figure 4–1 shows how the various register bits select the various modes of operation.

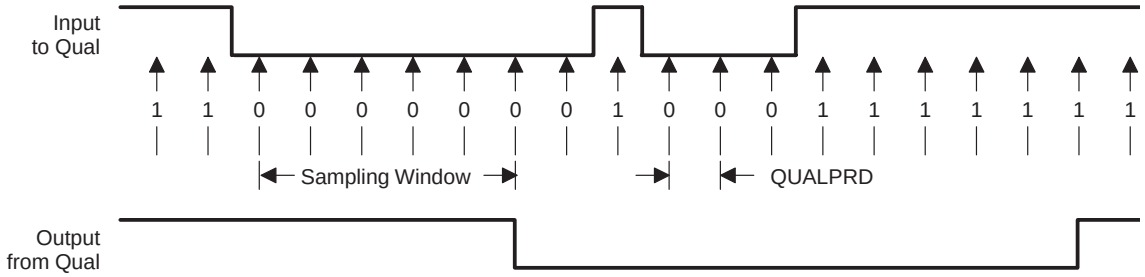
Figure 4–1. Modes of Operation



Notes: 1) Via the GPxDAT register, the state of any PIN can be read, regardless of the operating mode.

2) Some selected input signals are qualified by the QUALxCLK, which is a prescaled version of the high-speed peripheral clock (HSPCLK). The GPxQUAL register specifies the qualification sampling period. The sampling window is six samples wide and the output is only changed when all samples are the same (all 0s or all 1s) as shown in Figure 4–2. This feature removes unwanted spikes from the input signal.

Figure 4–2. I/P Qualifier Clock Cycles

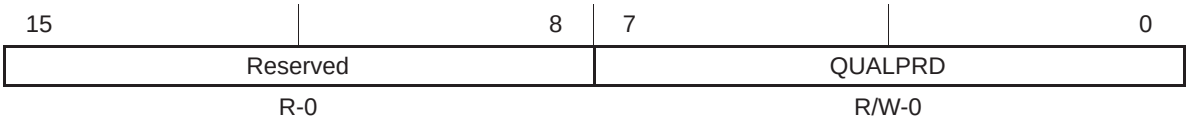


4.2 Registers

Table 4–3. GPIO A MUX Control (GPAMUX), GPIO A Direction Control (GPADIR) Register Bit Definitions

GPAMUX BIT	Peripheral Name (bit = 1)	GPIO Name (bit = 0)	GPADIR Bit	Type	Reset	Input Qual
EV-A Peripheral						
0	PWM1 (O)	GPIOA0	0	R/W	0	yes
1	PWM2 (O)	GPIOA1	1	R/W	0	yes
2	PWM3 (O)	GPIOA2	2	R/W	0	yes
3	PWM4 (O)	GPIOA3	3	R/W	0	yes
4	PWM5 (O)	GPIOA4	4	R/W	0	yes
5	PWM6 (O)	GPIOA5	5	R/W	0	yes
6	T1PWM_T1CMP (0)	GPIOA6	6	R/W	0	yes
7	T2PWM_T2CMP (0)	GPIOA7	7	R/W	0	yes
8	CAP1_QEP1 (I)	GPIOA8	8	R/W	0	yes
9	CAP2_QEP2 (I)	GPIOA9	9	R/W	0	yes
10	CAP3_QEPI1 (I)	GPIOA10	10	R/W	0	yes
11	TDIRA (I)	GPIOA11	11	R/W	0	yes
12	TCLKINA (I)	GPIOA12	12	R/W	0	yes
13	$\overline{\text{C1TRIP}}$ (I)	GPIOA13	13	R/W	0	yes
14	$\overline{\text{C2TRIP}}$ (I)	GPIOA14	14	R/W	0	yes
15	$\overline{\text{C3TRIP}}$ (I)	GPIOA15	15	R/W	0	yes

Figure 4–3. GPIO A Input Qualification Control (GPAQUAL) Register



Bits	Name	Description
15–8	Reserved	
7–0	QUALPRD	Specifies the qualification sampling period: 0x00 no qualification (just SYNC to SYSCLKOUT) 0x01 QUALPRD = SYSCLKOUT/2 0x02 QUALPRD = SYSCLKOUT/4 0xFF QUALPRD = SYSCLKOUT/510

Notes: 1) GPADIR bit = 0, configures corresponding GPIO pin as an input. GPADIR bit = 1, configures corresponding GPIO pin as an output.

2) The GPADAT, GPASET, GPACLEAR, GPATOGGLE registers have the same bit to I/O signal mapping as the GPAMUX and GPADIR registers.

3) The GPADAT register is a R/W register. Reading the register will reflect the current state of the input I/O signal (after qualification). Writing to the register will set the corresponding state of any I/O signal configured as an output.

4) The GPASET register is a write-only register (reads back 0). Writing a 1 to the corresponding bit of an I/O signal will cause the I/O signal to go high. Writing a 0 will have no effect.

5) The GPACLEAR register is a write-only register (reads back 0). Writing a 1 to the corresponding bit of an I/O signal will cause the I/O signal to go low. Writing a 0 will have no effect.

6) The GPATOGGLE register is a write-only register (reads back 0). Writing a 1 to the corresponding bit of an I/O signal will cause the I/O signal to toggle. Writing a 0 will have no effect.

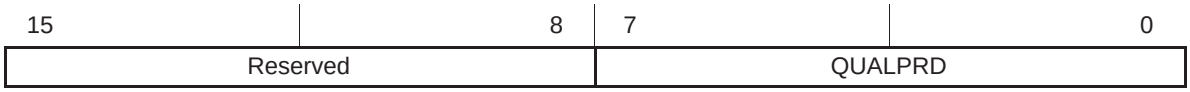
Table 4–4. GPIO B MUX Control (GPBMUX), GPIO B Direction Control (GPBDIR) Register Bit Definitions

GPBMUX Bit	Peripheral Name (bit = 1)	GPIO Name (bit = 0)	GPBDIR Bit	Type	Reset	Input Qual
		EV-B Peripheral				
0	PWM7 (O)	GPIOB0	0	R/W	0	yes
1	PWM8 (O)	GPIOB1	1	R/W	0	yes
2	PWM9 (O)	GPIOB2	2	R/W	0	yes
3	PWM10 (O)	GPIOB3	3	R/W	0	yes
4	PWM11 (O)	GPIOB4	4	R/W	0	yes
5	PWM12 (O)	GPIOB5	5	R/W	0	yes
6	T3PWM_T3CMP (0)	GPIOB6	6	R/W	0	yes
7	T4PWM_T4CMP (0)	GPIOB7	7	R/W	0	yes
8	CAP4_QEP3 (I)	GPIOB8	8	R/W	0	yes

Table 4–4. GPIO B MUX Control (GPBMUX), GPIO B Direction Control (GPBDIR) Register Bit Definitions (Continued)

GPBMUX Bit	Peripheral Name (bit = 1)	GPIO Name (bit = 0)	GPBDIR Bit	Type	Reset	Input Qual
9	CAP5_QEP4 (I)	GPIOB9	9	R/W	0	yes
10	CAP6_QEPI2 (I)	GPIOB10	10	R/W	0	yes
11	TDIRB (I)	GPIOB11	11	R/W	0	yes
12	TCLKINB (I)	GPIOB12	12	R/W	0	yes
13	$\overline{\text{C4TRIP}}$ (I)	GPIOB13	13	R/W	0	yes
14	$\overline{\text{C5TRIP}}$ (I)	GPIOB14	14	R/W	0	yes
15	$\overline{\text{C6TRIP}}$ (I)	GPIOB15	15	R/W	0	yes

Figure 4–4. GPIO B Input Qualification Control (GPBQUAL) Register



Bits	Name	Description
15–8	Reserved	
7–0	QUALPRD	Specifies the qualification sampling period: 0x00 no qualification (just SYNC to SYSCLKOUT) 0x01 QUALPRD = SYSCLKOUT/2 0x02 QUALPRD = SYSCLKOUT/4 . 0xFF QUALPRD = SYSCLKOUT/510

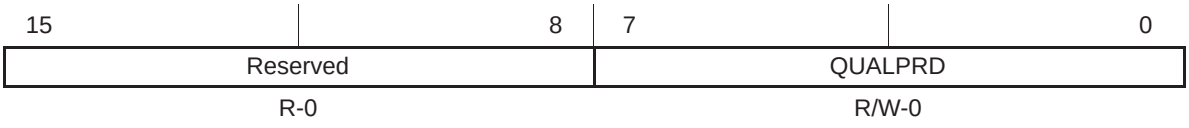
Table 4–5. GPIO D MUX Control (GPDMUX), GPIO D Direction Control (GPDDIR) Register Bit Definitions

GPDMUX Bit	Peripheral Name (bit = 1)	GPIO Name (bit = 0)	GPDDIR Bit	Input Qual
EV-A Peripheral:				
0	$\overline{\text{T1CTRIP\_PDPINTA}}$ (I)	GPIOD0	0	yes
1	$\overline{\text{T2CTRIP}}$ (I)	GPIOD1	1	yes
2	Reserved	GPIOD2	2	–
3	Reserved	GPIOD3	3	–

Table 4–5. GPIO D MUX Control (GPDMUX), GPIO D Direction Control (GPDDIR) Register Bit Definitions (Continued)

GPDMUX Bit	Peripheral Name (bit = 1)	GPIO Name (bit = 0)	GPDDIR Bit	Input Qual
4	Reserved	GPIOD4	4	–
EV-B Peripheral:				
5	$\overline{\text{T3CTRIP\_PDPINTB}}$ (I)	GPIOD5	5	yes
6	$\overline{\text{T4CTRIP}}$ (I)	GPIOD6	6	yes
7	Reserved	GPIOD7	7	–
8	Reserved	GPIOD8	8	–
9	Reserved	GPIOD9	9	–
10	Reserved	GPIOD10	10	–
11	Reserved	GPIOD11	11	–
12	Reserved	GPIOD12	12	–
13	Reserved	GPIOD13	13	–
14	Reserved	GPIOD14	14	–
15	Reserved	GPIOD15	15	–

Figure 4–5. GPIO D Input Qualification Control (GPDQUAL) Register

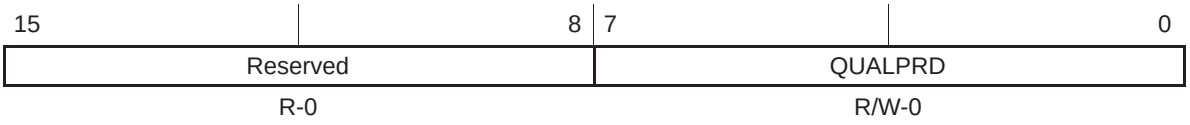


Bits	Name	Description
15–8	Reserved	
7–0	QUALPRD	Specifies the qualification sampling period: 0x00 no qualification (just SYNC to SYSCLKOUT) 0x01 QUALPRD = SYSCLKOUT/2 0x02 QUALPRD = SYSCLKOUT/4 . 0xFF QUALPRD = SYSCLKOUT/510

Table 4–6. GPIO E MUX Control (GPEMUX), GPIO E Direction Control (GPEDIR) Register Bit Definitions

GPEMUX Bit	Peripheral Name (bit = 1)	GPIO Name (bit = 0)	GPEDIR Bit	Type	Reset	Input Qual
		Interrupts:				
0	XINT1_$\overline{\text{XBIO}}$ (I)	GPIOE0	0	R/W	0	yes
1	XINT2_ADCSOC (I)	GPIOE1	1	R/W	0	yes
2	XNMI_XINT13 (I)	GPIOE2	2	R/W	0	yes
3	Reserved	GPIOE3	3	R	0	–
4	Reserved	GPIOE4	4	R	0	–
5	Reserved	GPIOE5	5	R = 0	0	–
6	Reserved	GPIOE6	6	R = 0	0	–
7	Reserved	GPIOE7	7	R = 0	0	–
8	Reserved	GPIOE8	8	R = 0	0	–
9	Reserved	GPIOE9	9	R = 0	0	–
10	Reserved	GPIOE10	10	R = 0	0	–
11	Reserved	GPIOE11	11	R = 0	0	–
12	Reserved	GPIOE12	12	R = 0	0	–
13	Reserved	GPIOE13	13	R = 0	0	–
14	Reserved	GPIOE14	14	R = 0	0	–
15	Reserved	GPIOE15	15	R = 0	0	–

Figure 4–6. GPIO E Input Qualification Control (GPEQUAL) Register



Bits	Name	Reset	Description
15–8	Reserved	0:0	
7–0	QUALPRD	0:0	Specifies the qualification sampling period: 0x00 no qualification (just SYNC to SYSCLKOUT) 0x01 QUALPRD = SYSCLKOUT/2 0x02 QUALPRD = SYSCLKOUT/4 . 0xFF QUALPRD = SYSCLKOUT/510

Table 4–7. GPIO F MUX Control (GPFMUX), GPIO F Direction Control (GPFDIR) Register Bit Definitions

GPFMUX Bit	Peripheral Name (bit = 1)	GPIO Name (bit = 0)	GPFDIR Bit	Type	Reset	Input Qual
SPI Peripheral						
0	SPISIMO (O)	GPIOF0	0	R/W	0	no
1	SPISOMI (I)	GPIOF1	1	R/W	0	no
2	SPICLK (I/O)	GPIOF2	2	R/W	0	no
3	SPISTE (I/O)	GPIOF3	3	R/W	0	no
SCIA Peripheral						
4	SCITXDA (O)	GPIOF4	4	R/W	0	no
5	SCIRXDA (I)	GPIOF5	5	R/W	0	no
CAN Peripheral						
6	CANTX (O)	GPIOF6	6	R/W	0	no
7	CANRX (I)	GPIOF7	7	R/W	0	no
McBSP Peripheral						
8	MCLKX (I/O)	GPIOF8	8	R/W	0	no
9	MCLKR (I/O)	GPIOF9	9	R/W	0	no
10	MFSX (I/O)	GPIOF10	10	R/W	0	no
11	MFSR (I/O)	GPIOF11	11	R/W	0	no

Table 4–7. GPIO F MUX Control (GPFMUX), GPIO F Direction Control (GPFDIR) Register Bit Definitions (Continued)

GPFMUX Bit	Peripheral Name (bit = 1)	GPIO Name (bit = 0)	GPFDIR Bit	Type	Reset	Input Qual
12	MDX (O)	GPIOF12	12	R/W	0	no
13	MDR (I)	GPIOF13	13	R/W	0	no
XF CPU Output Signal						
14	XF (0)	GPIOF14	14	R/W	0	no

Table 4–8. GPIO G MUX Control (GPGMUX), GPIO G Direction Control (GPGDIR) Register Bit Definitions

GPFMUX Bit	Peripheral Name (bit = 1)	GPIO Name (bit = 0)	GPFDIR Bit	Type	Reset	Input Qual
0	Reserved	GPIOG0	0	R	0	–
1	Reserved	GPIOG1	1	R	0	–
2	Reserved	GPIOG2	2	R	0	–
3	Reserved	GPIOG3	3	R	0	–
SCI-B Peripheral:						
4	SCITXDB (O)	GPIOG4	4	R/W	0	no
5	SCIRXDB (I)	GPIOG5	5	R/W	0	no
6	Reserved	GPIOG6	6	R	0	–
7	Reserved	GPIOG7	7	R	0	–
8	Reserved	GPIOG8	8	R	0	–
9	Reserved	GPIOG9	9	R	0	–
10	Reserved	GPIOG10	10	R	0	–
11	Reserved	GPIOG11	11	R	0	–
12	Reserved	GPIOG12	12	R	0	–
13	Reserved	GPIOG13	13	R	0	–
14	Reserved	GPIOG14	14	R	0	–
15	Reserved	GPIOG15	15	R	0	–

Chapter 5

System Configuration

This chapter describes peripheral frames and system configuration using the frames. It also describes the device emulation registers.

Topic **Page**

5.1 Peripheral Frame Registers

The F2810 device contains three 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 5–1.
- Peripheral Frame 1: These are peripherals that are mapped to the 32-bit peripheral bus. See Table 5–2.
- Peripheral Frame 2: These are peripherals that are mapped to the 16-bit peripheral bus. See Table 5–3.

Table 5–1. Peripheral Frame 0 Registers

Name	Address Range	Size (x16)	Access Type†
Device Emulation Registers	0x0000 0880 0x0000 09FF	384	EALLOW protected
reserved	0x0000 0A00 0x0000 0B00	128	
FLASH Registers‡	0x0000 0A80 0x0000 0ADF	96	EALLOW protected CSM Protected
Code Security Module Registers	0x0000 0AE0 0x0000 0AEF	16	EALLOW protected
reserved	0x0000 0AF0 0x0000 0B1F	48	
XINTF Registers	0x0000 0B20 0x0000 0B3F	32	Not EALLOW protected
reserved	0x0000 0B40 0x0000 0BFF	192	
CPU-TIMER0/1/2 Registers	0x0000 0C00 0x0000 0C3F	64	Not EALLOW protected
reserved	0x0000 0C40 0x0000 0CDF	160	
PIE Registers	0x0000 0CE0 0x0000 0CFF	32	Not EALLOW protected

† If registers are EALLOW protected, you cannot perform writes until you execute the EALLOW instruction. The EDIS instruction disables writes to prevent stray code or pointers from corrupting register contents.
‡ The flash registers are also protected by the Code Security Module (CSM).

Table 5–1. Peripheral Frame 0 Registers (Continued)

Name	Address Range	Size (x16)	Access Type†
PIE Vector Table	0x0000 0D00 0x0000 0DFF	256	EALLOW protected
reserved	0x0000 0E00 0x0000 0FFF	512	

† If registers are EALLOW protected, you cannot perform writes until you execute the EALLOW instruction. The EDIS instruction disables writes to prevent stray code or pointers from corrupting register contents.
‡ The flash registers are also protected by the Code Security Module (CSM).

Table 5–2. Peripheral Frame 1 Registers†

Name	Address Range	Size (x16)	Access Type†
eCAN Registers	0x0000 6000 0x0000 61FF	512	Not eallow protected
Reserved	0x0000 6200 0x0000 6FFF	3584	

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

Table 5–3. Peripheral Frame 2 Registers†

Name	Address Range	Size (x16)	Access Type§
reserved	0x0000 7000 0x0000 700F	16	
System Control Registers	0x0000 7010 0x0000 702F	32	EALLOW Protected
reserved	0x0000 7030 0x0000 703F	16	
SPI Registers	0x0000 7040 0x0000 704F	16	Not EALLOW Protected
SCI-A Registers	0x0000 7050 0x0000 705F	16	Not EALLOW Protected
reserved	0x0000 7060 0x0000 706F	16	
External Interrupt Registers	0x0000 7070 0x0000 707F	16	Not EALLOW Protected
reserved	0x0000 7080 0x0000 70BF	64	

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

Table 5–3. Peripheral Frame 2 Registers† (Continued)

Name	Address Range	Size (x16)	Access Type§
GPIO Mux Registers	0x0000 70C0 0x0000 70DF	32	EALLOW Protected
GPIO Data Registers	0x0000 70E0 0x0000 70FF	32	Not EALLOW Protected
ADC Registers	0x0000 7100 0x0000 711F	32	Not EALLOW Protected
reserved	0x0000 7120 0x0000 73FF	736	
EV-A Registers	0x0000 7400 0x0000 743F	64	Not EALLOW Protected
reserved	0x0000 7440 0x0000 74FF	192	
EV-B Registers	0x0000 7500 0x0000 753F	64	Not EALLOW Protected
reserved	0x0000 7540 0x0000 774F	528	
SCI-B Registers	0x0000 7750 0x0000 775F	16	Not EALLOW Protected
reserved	0x0000 7760 0x0000 77FF	160	
McBSP Registers	0x0000 7800 0x0000 783F	64	Not EALLOW Protected
reserved	0x0000 7840 0x0000 7FFF	1984	

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

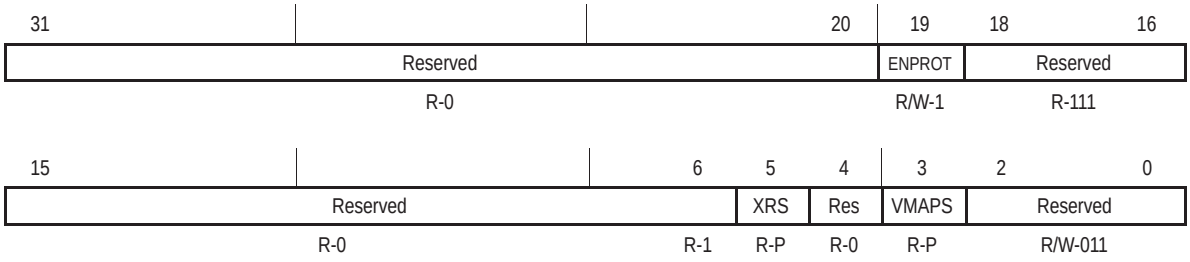
5.2 Device Emulation Registers

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 5–4.

Table 5–4. Device Emulation Registers

Name	Address Range	Size (x16)	Description
DEVICECNF	0x0000 0880 0x0000 0881	2	Device Configuration Register
DEVICEID	0x0000 0882 0x0000 0883	2	Device ID Register
PROTSTART	0x0000 0884	1	Block Protection Start Address Register
PROTRANGE	0x0000 0885	1	Block Protection Range Address Register
reserved	0x0000 0886 0x0000 09FF	378	

Figure 5–1. Device Configuration (DEVICECNF) Register



Legend: R = Read, W = Write, P = pin value after reset, -n = reset value

Table 5–5. Device Configuration (DEVICECNF) Register Bit Definitions

Bits	Name	Description
31–20	Reserved	
19	ENPROT	Enable Write-Read Protection Mode Bit. 1 Enables write-read protection as specified by the PROTSTART and PROTRANGE registers 0 Disables write-read protection mode
18–6	Reserved	
5	XRS	Reset Input Signal Status. This is connected directly to the $\overline{\text{XRS}}$ input pin.
4	Reserved	

Table 5–5. Device Configuration (DEVICECNF) Register Bit Definitions (Continued)

Bits	Name	Description
3	VMAPS	VMAP Configure Status. This indicates the status of VMAP.
2–0	Reserved	

Table 5–6. DEVICEID Register Bit Definitions

Bits	Name	Type	Reset	Description
31–16	REVID	R	0x0000 (for first silicon)	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.
15–0	PARTID	R	Dependent on device	These 16 bits specify the part number of the device as follows: 0x0001: F2810 device 0x0002: F2812 device

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 will be 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
```

NOTE: The C28x CPU automatically protects writes followed by reads to the same memory address. The protection mechanism described above is for cases where the address is not the same, but within a given region in memory (as defined by the PROTSTART and PROTRANGE registers).

Table 5–7. PROTSTART and PROTRANGE Registers

Name	Address	Size	Type	Reset	Description
PROTSTART	0x0000 0884	16	R/W	0x0100†	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	0x0000 0885	16	R/W	0x00FF†	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).

† The default values of these registers on reset are selected to cover the Peripheral Frame 1, Peripheral Frame 2, and XINTF Zone 1 areas of the memory map (address range 0x0000 4000 to 0x0000 8000).

Table 5–8. PROTSTART Valid Values†

Start Address	Register Value	Register Bits															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000 0000	0x0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0000 0040	0x0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x0000 0080	0x0002	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x0000 00C0	0x0003	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
0x003F FF00	0xFFFC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
0x003F FF40	0xFFFD	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0x003F FF80	0xFFFE	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0x003F FFC0	0xFFFF	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 5–9. PROTRANGE Valid Values‡

Block Size	Register Value	Register Bits															
		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
2K	0x001F	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
4K	0x003F	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
8K	0x007F	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
16K	0x00FF	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
32K	0x01FF	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
64K	0x03FF	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
128K	0x07FF	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
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

‡ 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.

Chapter 6

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 ($\overline{\text{INT1}}$ to $\overline{\text{INT12}}$). 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.

Topic **Page**

6.1 Overview of the PIE Controller

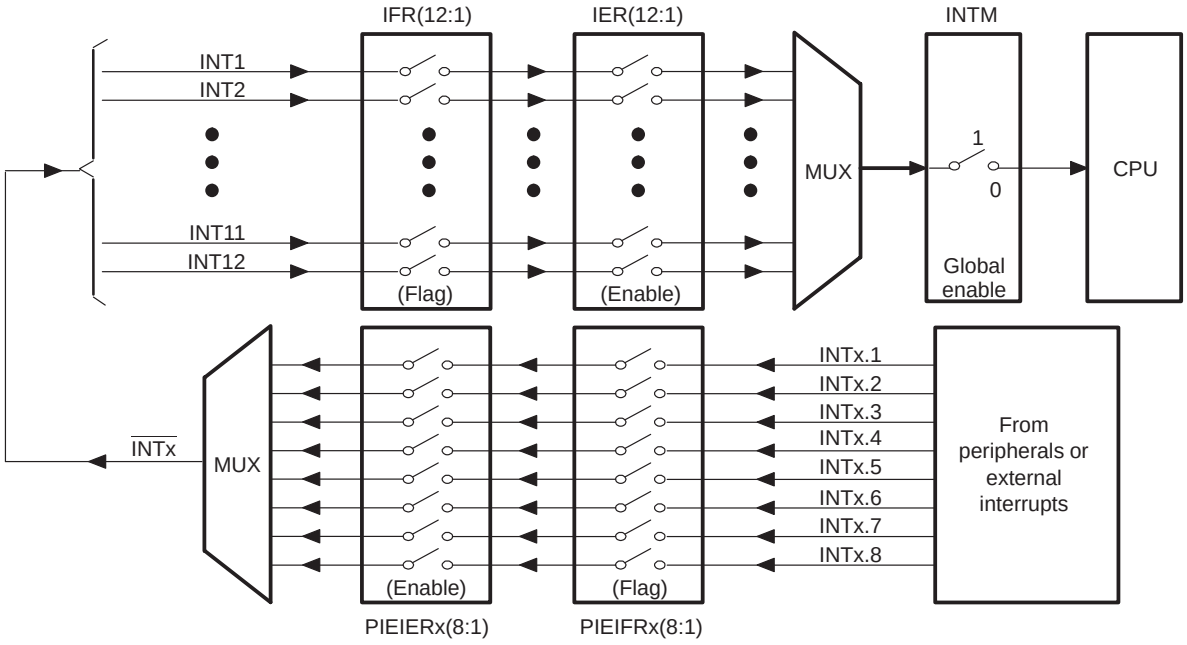
The TMS320C/F28x™ CPU supports one nonmaskable interrupt (NMI) and 16 maskable prioritized interrupt requests (INT1–INT14, RTOSINT, and DLOGINT) at the CPU level. The 28x devices have many peripherals and each peripheral is capable of generating one or more interrupts in response to many events at the peripheral level. Because the CPU does not have sufficient capacity to handle all peripheral interrupt requests at the CPU level, a centralized peripheral interrupt controller (PIE) is required to arbitrate the interrupt requests from various sources such as peripherals and other external pins.

6.1.1 Interrupt Operation Sequence

The PIE block multiplexes eight peripheral and external pin interrupts into one CPU interrupt. In total, the PIE block can handle 96 interrupts. These interrupts are divided into 12 CPU interrupt groups (INT1 – INT12) with 8 interrupts per group.

The interrupt operation sequence for all PIE interrupts is shown in Figure 6–1.

Figure 6–1. 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 will be 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

Each interrupt in the PIE block has an associated flag bit (PIEIFRx.y) and enable bit (PIEIERx.y. In addition, there is one acknowledge bit (PIEACK) for every CPU interrupt group (INT1 to INT12) and is referred to as PIEACKx. Figure 6–2 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 will then check 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 will send the interrupt request to the CPU. If PIEACKx is set, then the PIE will wait until it is cleared to send the request for INTx.

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.

As shown in Table 6–1, 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 F281x 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.

Figure 6–2. Typical PIE/CPU Interrupt Response – INTx.y

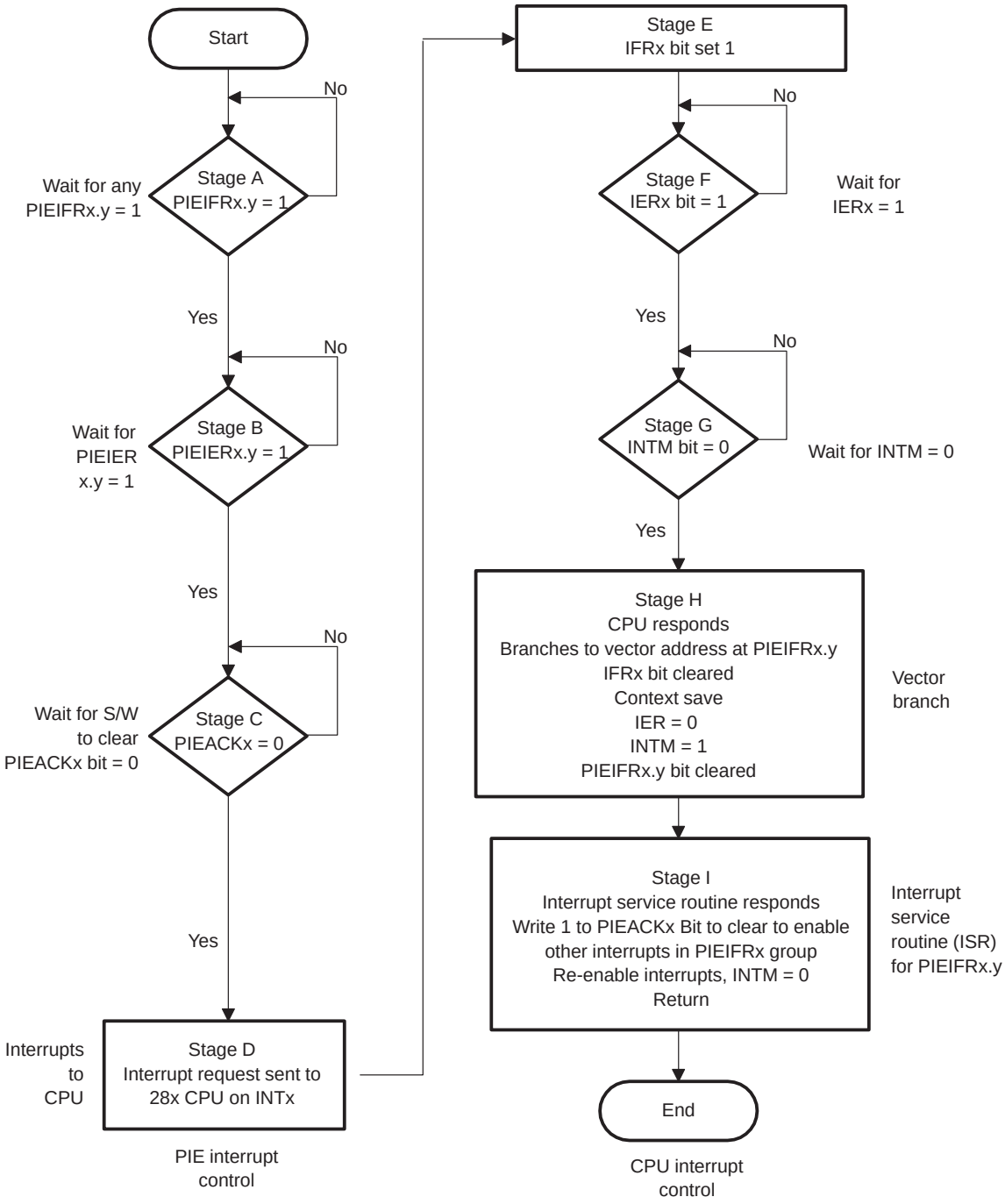


Table 6–1. Enabling Interrupt

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

Once an interrupt has been requested and properly enabled, the CPU prepares for and then executes the corresponding interrupt service routine. This process is described in detail in the *TMS320C28x DSP CPU and Instruction Set Reference Guide* (literature number 430).

The address for the interrupt service routine that will be 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.

6.1.2 Vector Table Mapping

On 28xx devices, the interrupt vector table can be mapped to five distinct locations in memory. In practice only the PIE vector table mapping will be used for F28xx devices.

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

VMAP: This bit 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 F2810/12 operation leave this bit set.

M0M1MAP: This bit 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.

MP/MC: This bit is found in XINTCNF2 Register (bit 8). On the devices with an external interface (XINTF) the default value of this bit, on reset, is set by the XMP/MC input device signal. On devices without an XINTF, the XMP/MC is tied low internally. The state of this bit can be modified after reset by writing to the XINTCNF2 register (address 0x0000 0B34).

ENPIE: This bit 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 6–2.

Table 6–2. Interrupt Vector Table Mapping†

Vector MAPS	Vectors Fetched From	Address Range	VMAP	M0M1MAP	MP/$\overline{MC}$	ENPIE
M1 Vector‡	M1 SARAM Block	0x000000–0x00003F	0	0	X	X
M0 Vector	M0 SARAM Block	0x000000–0x00003F	0	1	X	X
BROM Vector	ROM Block	0x3FFFC0–0x3FFFFF	1	X	0	0
XINTF Vector§	XINTF Zone 7 Block	0x3FFFC0–0x3FFFFF	1	X	1	0
PIE Vector	PIE Block	0x000D00–0x000DFF	1	X	X	1

† On the F2810 and F2812 devices, the VMAP and M0M1MAP modes are set to “1” on reset. The ENPIE mode is forced to “0” on reset.
‡ Vector map M1 Vector is a reserved mode only.
§ Valid on F2812 only

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

After a device reset operation, the vector table will mapped as shown in Table 1–5.

Table 6–3. Vector Table Mapping After Reset Operation†

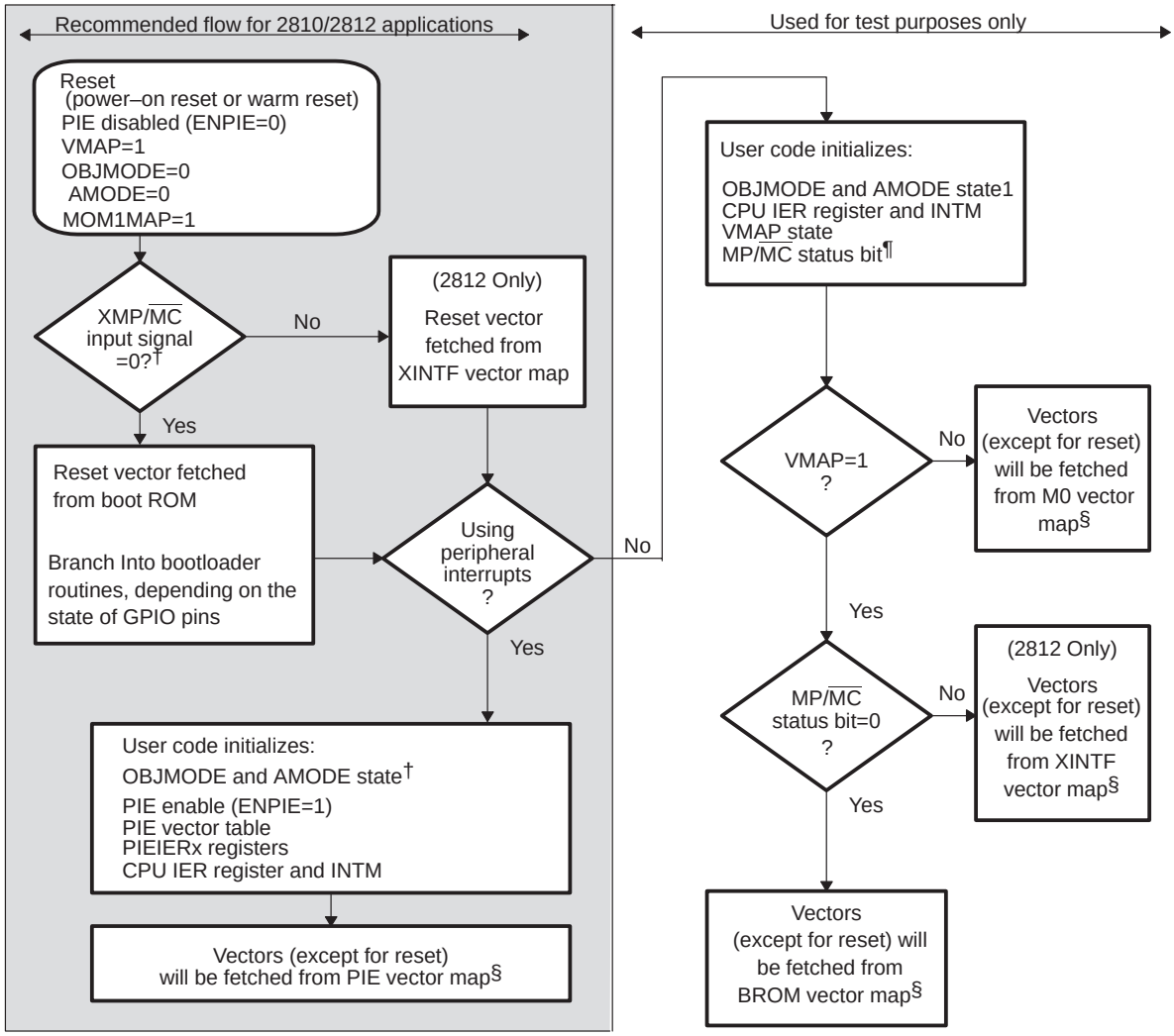
VECTOR MAPS	RESET FETCHED FROM	ADDRESS RANGE	VMAP	M0M1MAP	MP/$\overline{MC}$	ENPIE
BROM Vector	ROM Block	0x3FFFC0–0x3FFFFF	1	1	0	0
XINTF Vector§	XINTF Zone 7 Block	0x3FFFC0–0x3FFFFF	1	1	1	0

† On the F2810 and F2812 devices, the VMAP and M0M1MAP modes are set to “1” on reset. The ENPIE mode is forced to “0” on reset.
§ Valid on F2812 only

After the reset and boot is complete, the PIE vector table should be initialized by the user’s code. The application will then enable the PIE vector table. From that point on the interrupt vectors will be fetched from the PIE vector table. Note: when a reset occurs, the reset vector will always be fetched from the vector table as shown in Table 1–5. After a reset the PIE vector table is always disabled.

Figure 6–3 illustrates the process by which the vector table mapping is selected.

Figure 6–3. Reset Flow Diagram



† The XMP/MC input signal is tied low internally on the F2810.
‡ The compatibility operating mode of the F2810 and F2812 is determined by a combination of the OBJMODE and AMODE bits in Status Register 1 (ST1):

Operating Mode	OBJMODE	AMODE	
C28x Mode	1	0	
C2xLP Source-Compatible	1	1	
C27x Object-Compatible	0	0	(Default at reset)

§ The reset vector is always fetched from either the BROM or XINTF vector map depending on the XMP/MC input signal.
¶ The state of the XMP/MC signal is latched into the MP/MC bit at reset, it can then be modified by software.

6.1.3 The PIE Vector Table

The PIE vector table (see Table 6–4) consists of a 256 x 16 SARAM block that can also be used as RAM 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 will be presented to the CPU simultaneously by the PIE block, and the CPU will service INT1.1 first. If INT1.1 should occur simultaneously with INT1.8, then INT1.1 will be sent to the CPU first and then INT1.8 will follow. Interrupt prioritization is performed during the vector fetch portion of the interrupt processing.

A “TRAP 1” to “TRAP 12” instruction or an “INTR INT1” to “INTR INT12” instruction will fetch the vector from the first location of each group (“INTR1.1” to “INT12.1”). Similarly an ”OR IFR,#16bit” operation will cause 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,#16bit” operations will fetch the vector from the respective table location. You should avoid using such operations for INTR1 to INTR12. The “TRAP #0” operation will return a vector value of 0x000000

The vector table is EALLOW protected.

Table 6–4. PIE Vector Table

Name	Address	Size (x16)	Description	Core Priority	PIE Group Priority
not used	0x0000 0D00	2	RESET never fetched here	1 (highest)	–
not used	0x0000 0D02	2	reserved	–	–
not used	0x0000 0D04	2	reserved	–	–
not used	0x0000 0D06	2	reserved	–	–
not used	0x0000 0D08	2	reserved	–	–
not used	0x0000 0D0A	2	reserved	–	–
not used	0x0000 0D0C	2	reserved	–	–
not used	0x0000 0D0E	2	reserved	–	–
not used	0x0000 0D10	2	reserved	–	–
not used	0x0000 0D12	2	reserved	–	–
not used	0x0000 0D14	2	reserved	–	–
not used	0x0000 0D16	2	reserved	–	–

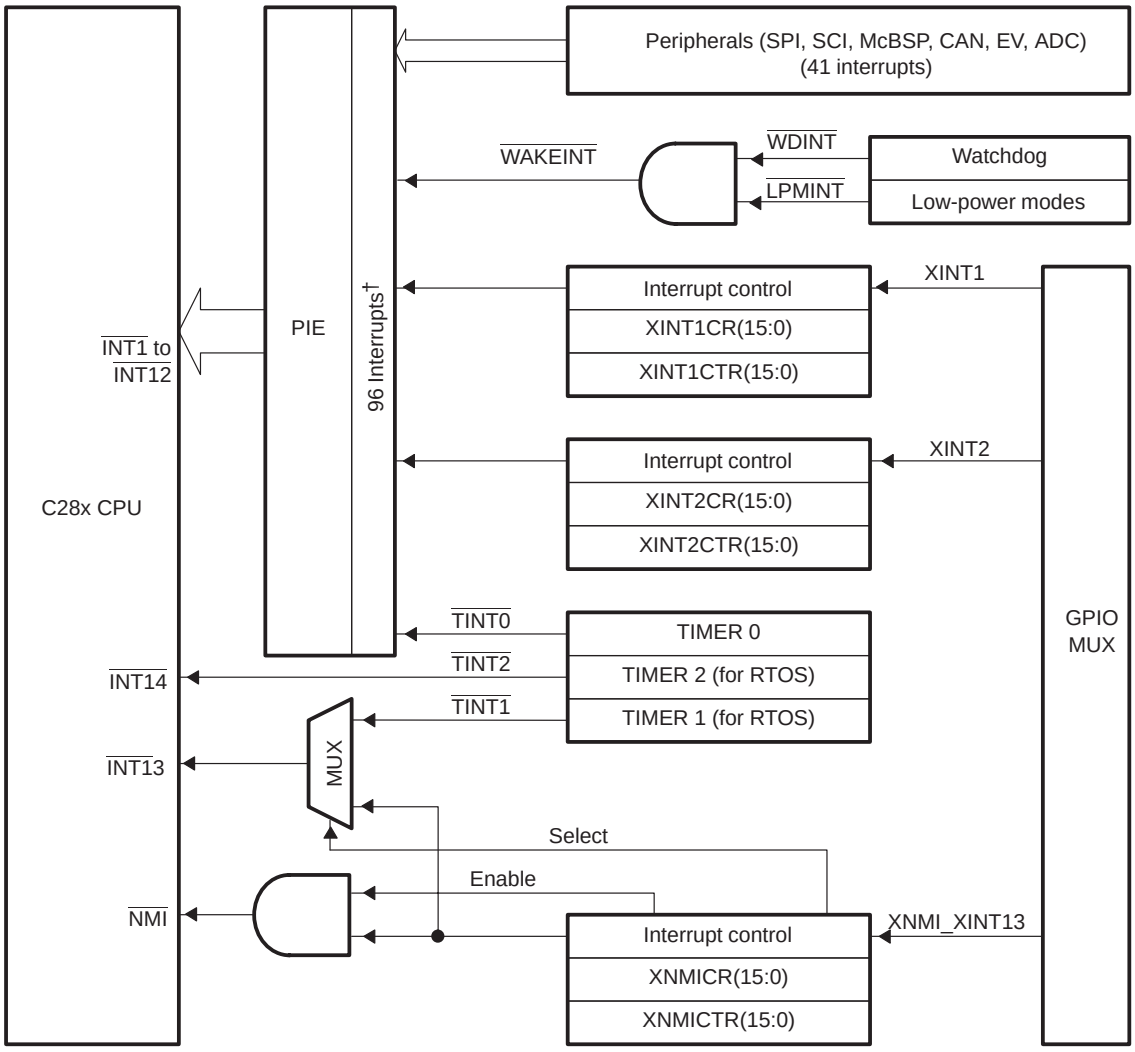
Table 6–4. PIE Vector Table (Continued)

Name	Address	Size (x16)	Description	Core Priority	PIE Group Priority
not used	0x0000 0D18	2	reserved	–	–
INT13	0x0000 0D1A	2	External Interrupt 13 (XINT13) or CPU-Timer 1 (for RTOS use)	17	–
INT14	0x0000 0D1C	2	CPU-Timer 2 (for RTOS use)	18	–
DATALOG	0x0000 0D1E	2	CPU Data Logging Interrupt	19 (lowest)	–
RTOSINT	0x0000 0D20	2	CPU Real-Time OS Interrupt	4	–
EMUINT	0x0000 0D22	2	CPU Emulation Interrupt	2	–
NMI	0x0000 0D24	2	External Non-Maskable Interrupt	3	–
ILLEGAL	0x0000 0D26	2	Illegal Operation	–	–
USER0	0x0000 0D28	2	User-Defined Trap	–	–
.	.	.	.	.	.
USER11	0x0000 0D3E	2	User-Defined Trap	–	–
INT1.1	0x0000 0D40	2			1 (highest)
.	.	.	Group 1 Interrupt Vectors	5	.
INT1.8	0x0000 0D4E	2			8 (lowest)
.	.	.			
.	.	.	Group 2 Interrupt Vectors to Group 11 Interrupt Vectors	6 to 15	
.	.	.			
INT12.1	0x0000 0DF0	2			1 (highest)
.	.	.	Group 12 Interrupt Vectors	16	.
INT12.8	0x0000 0DFE	2			8 (lowest)

6.2 Interrupt Sources

Figure 6–4 shows how the various interrupt sources are multiplexed within the F2810 and F2812 devices. This muxing scheme may not be exactly the same on all F28xx devices. See the datasheet of your particular device for details.

Figure 6–4. Interrupt Sources



† Out of a possible 96 interrupts, 45 are currently used by peripherals.

Notes: 1) In the ”GPIO Mux”, the XINT1, XINT2 and XNMI signals are synchronized and optionally qualified by a user programmable number of clock cycles. This filters out glitches from the input source. See the ”GPIO Mux” section for more details.

2) The WAKEINT input must be synchronized (using SYSCLKOUT of the processor) before being fed to the PIE block.

The interrupt grouping for peripherals and external interrupts connected to the PIE module is shown in Table 6–5. Each row in the table shows the 8 interrupts multiplexed into a particular CPU interrupt.

Table 6–5. PIE Peripheral Interrupts†

CPU Interrupts	PIE Interrupts							
	INTx.1	**INTx.2**	**INTx.3**	**INTx.4**	**INTx.5**	**INTx.6**	**INTx.7**	**INTx.8**
INT1	PDPINTA (EV-A)	PDPINTB (EV-B)	reserved	XINT1	XINT2	ADCINT (ADC)	TINT0 (TIMER 0)	$\overline{\text{WAKEINT}}$ (LPM/WD)
INT2	CMP1INT (EV-A)	CMP2INT (EV-A)	CMP3INT (EV-A)	T1PINT (EV-A)	T1CINT (EV-A)	T1UFINT (EV-A)	T1OFINT (EV-A)	reserved
INT3	T2PINT (EV-A)	T2CINT (EV-A)	T2UFINT (EV-A)	T2OFINT (EV-A)	CAPINT1 (EV-A)	CAPINT2 (EV-A)	CAPINT3 (EV-A)	reserved
INT4	CMP4INT (EV-B)	CMP5INT (EV-B)	CMP6INT (EV-B)	T3PINT (EV-B)	T3CINT (EV-B)	T3UFINT (EV-B)	T3OFINT (EV-B)	reserved
INT5	T4PINT (EV-B)	T4CINT (EV-B)	T4UFINT (EV-B)	T4OFINT (EV-B)	CAPINT4 (EV-B)	CAPINT5 (EV-B)	CAPINT6 (EV-B)	reserved
INT6	SPIRXINTA (SPI)	SPITXINTA (SPI)	reserved	reserved	MRINT (McBSP)	MXINT (McBSP)	***reserved***	***reserved***
INT7	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved
INT8	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved
INT9	SCIRXINTA (SCI-A)	SCITXINTA (SCI-A)	SCIRXINTB (SCI-B)	SCITXINTB (SCI-B)	ECAN0INT (CAN)	ECAN1INT (CAN)	reserved	reserved
INT10	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved
INT11	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved
INT12	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved

† Out of the 96 possible interrupts, 45 interrupts are currently used. the remaining interrupts are reserved for future devices. However, these interrupts can be used as software interrupts if they are enabled at the PIEIFRx level.

6.3 PIE Configuration Registers

The registers controlling the functionality of the PIE block are shown in Table 6–6.

Table 6–6. 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

Note: The PIE configuration and control registers are not protected by EALLOW mode. The PIE vector table is protected.

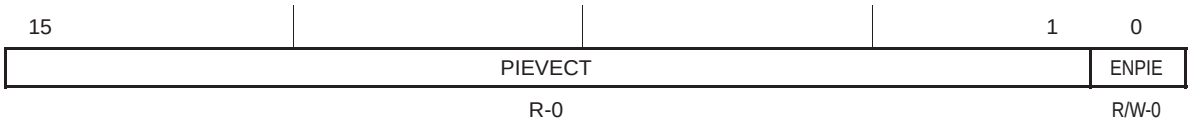
Table 6–6. PIE Configuration and Control Registers (Continued)

Name	Address	Size (x16)	Description
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
Reserved	0x0000–0CFA 0x0000–0CFF	6	Reserved

Note: The PIE configuration and control registers are not protected by EALLOW mode. The PIE vector table is protected.

6.4 PIE Interrupt Registers

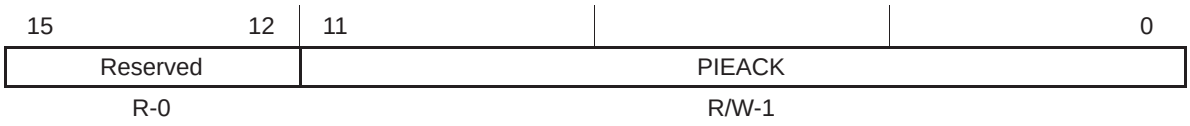
Table 6–7. PIECTRL Register



Bits	Name	Description
15–1	PIEVECT	These bits indicate the vector address of the vector fetched from the PIE Vector Table. 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.
0	ENPIE	Enable vector fetching from PIE block. When this bit is set to 1, all vectors are fetched from the PIE vector table. If this bit is set to 0, the PIE block is disabled and vectors are fetched from the CPU vector table in boot ROM or external interface Zone 7. All PIE block registers (PIEACK, PIEIFR, PIEIER) can be accessed even when the PIE block is disabled. **Note:** The reset vector is never fetched from the PIE, even when it is enabled. This vector is always fetched from boot ROM or XINTF Zone 7 depending on the state of the XMPNMC input signal.

6.4.1 PIE Interrupt Acknowledge Register (PIEACKx)

Figure 6–5. PIE Interrupt Acknowledge Register (PIEACKx) Register



Bits	Name	Description
15–12	Reserved	
11–0	PIEACK	Writing a 1 to the respective interrupt bit enables the PIE block to drive a pulse into the core interrupts input, if an interrupt is pending on any of the group interrupts. Reading this register indicates if an interrupt is pending in the respective group. Bit 0 refers to $\overline{\text{INT1}}$ up to Bit 11, which refers to $\overline{\text{INT12}}$. Note: Writes of 0 are ignored.

6.4.2 PIE Interrupt Flag Registers

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

Figure 6–6. PIEIFRx Register (x = 1 to 12)

15	14	13	12	11	10	9	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

Bits	Name	Description
15–8	Reserved	
7	INTx.8	These register bits indicate if an interrupt is currently active. They behave very much like the core interrupt flag register. When an interrupt is active, the respective register bit is set. The bit is cleared when the interrupt is serviced or by writing a 0 to the register bit. This register can also be read to determine which interrupts are active or pending. 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	

Notes: 1) All of the above registers reset values are set by reset.
2) Hardware has priority over CPU accesses to the PIEIFR registers.
3) The PIEIFR register bit is cleared during the interrupt vector fetch portion of the interrupt processing.

6.4.3 PIE Interrupt Enable Registers

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

Figure 6–7. 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

Bits	Name	Description
15–8	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 will disable the servicing of the bit.
6	INTx.7	
5	INTx.6	x = 1 to 12. INTx means CPU INT1 to INT12
4	INTx.5	
3	INTx.4	
2	INTx.3	
1	INTx.2	
0	INTx.1	

Note: All of the above registers reset values are set by reset.

6.4.4 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 acknowledgement.

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.

Notes:

1) To clear an IFR bit, you must write a one to it, not a zero.

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 6–8. 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

Note: R = Read access, W = Write access, –0 = value after reset

Bits	Name	Description
15	RTOSINT	Real-time operating system flag. This bit 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. This bit 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

Figure 6–8.Interrupt Flag Register (IFR) — CPU Register (Continued)

Bits	Name	Description
13	INT14	Interrupt 14 flag. This bit 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. This bit is the flag for interrupts connected to CPU interrupt level INT13. 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. This bit 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. This bit 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. This bit 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. This bit 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. This bit 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

Figure 6–8.Interrupt Flag Register (IFR) — CPU Register (Continued)

Bits	Name	Description
6	INT7	Interrupt 7 flag. This bit 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
5	INT6	Interrupt 6 flag. This bit 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. This bit 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. This bit 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. This bit 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. This bit 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. This bit 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

6.4.5 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 2–7, and descriptions of the bits follow the figure.

Figure 6–9. 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

Note: R = Read access, W = Write access, –0 = value after reset

Bits	Name	Description
15	RTOSINT	Real-time operating system interrupt enable. This bit enables or disables the CPU RTOS interrupt. 0 Level INT6 is disabled 1 Level INT6 is enabled
14	DLOGINT	Data logging interrupt enable. This bit enables or disables the CPU data logging interrupt. 0 Level INT6 is disabled 1 Level INT6 is enabled
13	INT14	Interrupt 14 enable. This bit enables or disables CPU interrupt level INT14. 0 Level INT14 is disabled 1 Level INT14 is enabled
12	INT13	Interrupt 13 enable. This bit enables or disables CPU interrupt level INT13. 0 Level INT13 is disabled 1 Level INT13 is enabled
11	INT12	Interrupt 12 enable. This bit enables or disables CPU interrupt level INT12. 0 Level INT12 is disabled 1 Level INT12 is enabled
10	INT11	Interrupt 11 enable. This bit enables or disables CPU interrupt level INT11. 0 Level INT11 is disabled 1 Level INT11 is enabled
9	INT10	Interrupt 10 enable. This bit enables or disables CPU interrupt level INT10. 0 Level INT10 is disabled 1 Level INT10 is enabled

Figure 6–9.Interrupt Enable Register (IER) — CPU Register (Continued)

Bits	Name	Description
8	INT9	Interrupt 9 enable. This bit enables or disables CPU interrupt level INT9. 0 Level INT9 is disabled 1 Level INT9 is enabled
7	INT8	Interrupt 8 enable. This bit enables or disables CPU interrupt level INT8. 0 Level INT8 is disabled 1 Level INT8 is enabled
6	INT7	Interrupt 7 enable. This bit enables or disables CPU interrupt level INT7. 0 Level INT7 is disabled 1 Level INT7 is enabled
5	INT6	Interrupt 6 enable. This bit enables or disables CPU interrupt level INT6. 0 Level INT6 is disabled 1 Level INT6 is enabled
4	INT5	Interrupt 5 enable. This bit enables or disables CPU interrupt level INT5. 0 Level INT5 is disabled 1 Level INT5 is enabled
3	INT4	Interrupt 4 enable. This bit enables or disables CPU interrupt level INT4. 0 Level INT4 is disabled 1 Level INT4 is enabled
2	INT3	Interrupt 3 enable. This bit enables or disables CPU interrupt level INT3. 0 Level INT3 is disabled 1 Level INT3 is enabled
1	INT2	Interrupt 2 enable. This bit enables or disables CPU interrupt level INT2. 0 Level INT2 is disabled 1 Level INT2 is enabled
0	INT1	Interrupt 1 enable. This bit 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 DEBIER 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 DEBIER register. At reset, all the DBGIER bits are set to 0.

Figure 6–10. 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							

7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
R/W-0							

Note: R = Read access, W = Write access, –0 = value after reset

Bits	Name	Description
15	RTOSINT	Real-time operating system interrupt enable. This bit enables or disables the CPU RTOS interrupt. 0 Level INT6 is disabled 1 Level INT6 is enabled
14	DLOGINT	Data logging interrupt enable. This bit enables or disables the CPU data logging interrupt. 0 Level INT6 is disabled 1 Level INT6 is enabled
13	INT14	Interrupt 14 enable. This bit enables or disables CPU interrupt level INT14. 0 Level INT14 is disabled 1 Level INT14 is enabled
12	INT13	Interrupt 13 enable. This bit enables or disables CPU interrupt level INT13. 0 Level INT13 is disabled 1 Level INT13 is enabled
11	INT12	Interrupt 12 enable. This bit enables or disables CPU interrupt level INT12. 0 Level INT12 is disabled 1 Level INT12 is enabled

Figure 6–10. Debug Interrupt Enable Register (DBGIER) — CPU Register (Continued)

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

Figure 6–10. Debug Interrupt Enable Register (DBGIER) — CPU Register (Continued)

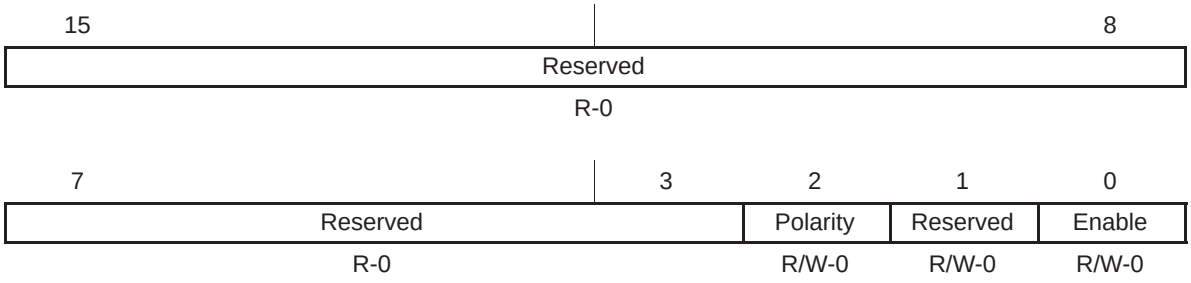
Bits	Name	Description
0	INT1	Interrupt 1 enable. This bit enables or disables CPU interrupt level INT1. 0 Level INT1 is disabled 1 Level INT1 is enabled

6.5 External Interrupt Control Registers

Some devices support three external masked external interrupts XINT1, XINT2, XINT13. 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 which is reset to zero when a valid interrupt edge is detected. This counter can be used to accurately time stamp the interrupt.

6.5.1 External Interrupt 1 Control Register (XINT1CR)

Figure 6–11. External Interrupt 1 Control Register (XINT1CR) — Address 7070h

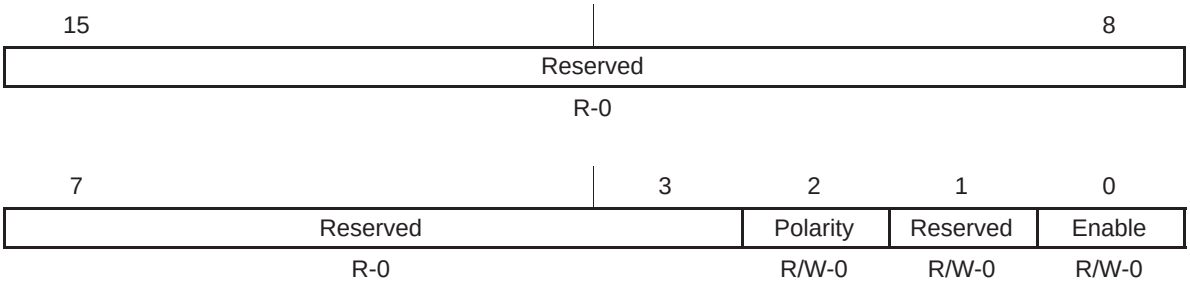


Note: R = Read access, W = Write access, –0 = value after reset

Bits	Name	Description
15–3	Reserved	Reads return zero; writes have no effect.
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.
		0 Interrupt generated on a falling edge (high-to-low transition)
		1 Interrupt generated on a rising edge low-to-high transition)
1	Reserved	Reads return Zero; writes have no effect
0	Enable	This read/write bit enables or disables external interrupt XINT1.
		0 Disable interrupt
		1 Enable interrupt

6.5.2 External Interrupt 2 Control Register (XINT2CR)

Figure 6–12. External Interrupt 2 Control Register (XINT2CR) — Address 7071h

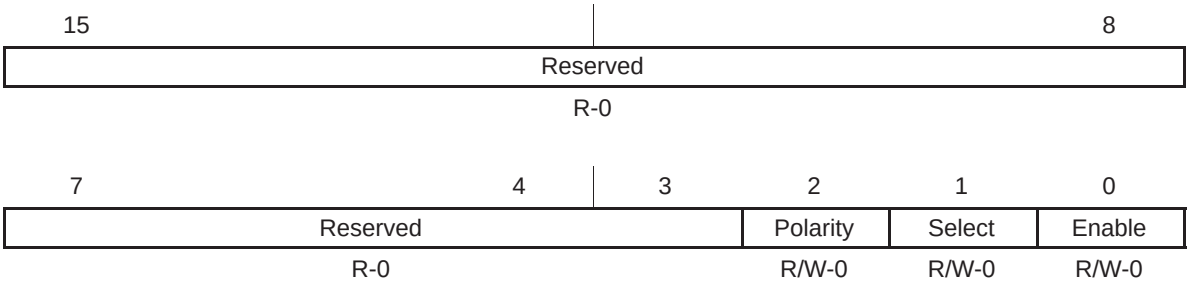


Note: R = Read access, W = Write access, –0 = value after reset

Bits	Name	Description
15–3	Reserved	Reads return zero; writes have no effect.
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. Interrupt generated on a falling edge (high-to-low transition) Interrupt generated on a rising edge low-to-high transition)
1	Reserved	Reads return Zero; writes have no effect
0	Enable	This read/write bit enables or disables external interrupt XINT2. Disable interrupt Enable interrupt

6.5.3 External NMI Interrupt Control Register (XMNICR)

Figure 6–13. NMI Interrupt Control Register (XNMICR) — Address 7077h



Note: R = Read access, W = Write access, –0 = value after reset

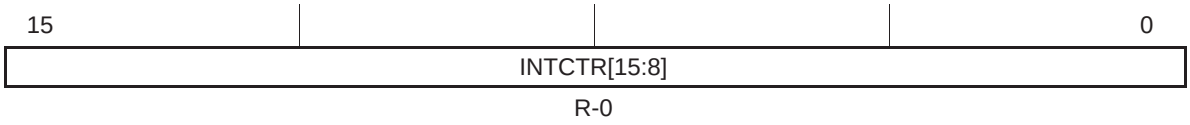
Figure 6–13. NMI Interrupt Control Register (XNMICR) — Address 7077h (Continued)

Bits	Name	Description
15–3	Reserved	Reads return zero; writes have no effect.
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.
		0 Interrupt generated on a falling edge (high-to-low transition)
		1 Interrupt generated on a rising edge low-to-high transition)
1	Select	
		0 Timer 1 Connected To INT13
		1 XNMI Connected To INT13
0	Enable	This read/write bit enables or disables external interrupt NMI
		0 Disable interrupt
		1 Enable interrupt

For each external interrupt, 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.

6.5.4 External Interrupt 1 Counter Register (XINT1CTR)

Figure 6–14. External Interrupt 1 Counter (XINT1CTR) — Address 7078h

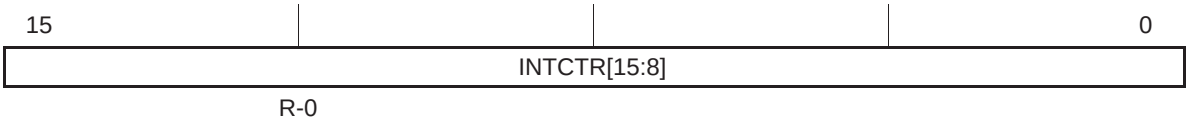


Note: R = Read access, –0 = value after reset

Bits	Name	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 will stop. The counter is a free-running counter and will wrap around to zero when the max value is reached. The counter is a read only register and can only be reset to zero by a valid interrupt edge or by reset.

6.5.5 External Interrupt 2 Counter Register (XINT2CTR)

Figure 6–15. External Interrupt 2 Counter (XINT2CTR) — Address 7079h

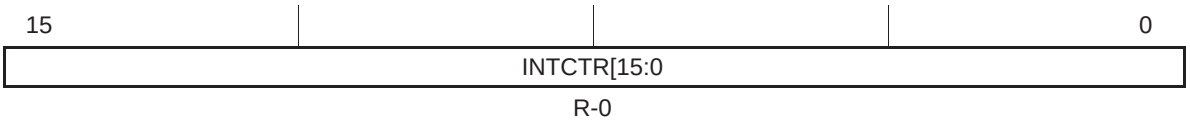


Note: R = Read access, –0 = value after reset

Bits	Name	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 will stop. The counter is a free–running counter and will wrap around to zero when the max value is reached. The counter is a read only register and can only be reset to zero by a valid interrupt edge or by reset.

6.5.6 External NMI Interrupt Counter Register (XNMICTR)

Figure 6–16. External NMI Interrupt Counter (XNMICTR) — Address 707Fh



Note: R = Read access, –0 = value after reset

Bits	Name	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 will stop. The counter is a free–running counter and will wrap around to zero when the max value is reached. The counter is a read only register and can only be reset to zero by a valid interrupt edge or by reset.