TEXAS INSTRUMENTS

TMS320C24x DSP Controllers

Peripheral Library and Specific Devices

Reference Set

Volume 2

1997 **Digital Signal Processing Solutions**

TEXAS INSTRUMENTS

Printed in U.S.A., March 1997
D412016-9761 revision A

SPRU161A

Reference Set

Volume 2

TMS320C24x DSP Controllers
Peripheral Library and Specific Devices

1997

TMS320C24x DSP Controllers Reference Set

Volume 2: Peripheral Library and Specific Devices

This document contains preliminary data
current as of publication date and is subject
to change without notice.

Literature Number: SPRU161A
Manufacturing Part Number: D412016-9761 revision A
March 1997

PRINTED WITH SOY INK™

TEXAS INSTRUMENTS

Printed on Recycled Paper

Preface

Read This First

About This Manual

This manual (volume 2 of a 2-volume set) consists of two sections. Section I describes the peripherals available in the TMS320C24x digital signal processor (DSP) controller family and their operation. Section II describes specific device configurations of the 'C24x family. In this document, the TMS320C24x is also referred to as the 'C24x.

The *TMS320C24x DSP Controllers Reference Set, Volume 1: CPU, System, and Instruction Set* (literature number SPRU160) describes the architecture, central processing unit (CPU), system hardware, assembly language instructions, and general operation of the 'C24x.

How to Use This Manual

The following table summarizes the 'C24x information contained in this manual:

If you are looking for information about	Turn to
Digital I/O Ports	Chapter 9, *Digital I/O Ports*
Dual 10-Bit A/D Converter	Chapter 3, *Dual 10-Bit Analog to Digital Converter (ADC) Module*
Event Manager	Chapter 2, *Event Manager Module*
External Memory Interface	Chapter 8, *External Memory Interface*
Flash Memory	Chapter 7, *Flash Memory Module*
PLL Clock Module	Chapter 10, *PLL Clock Module*
Serial Communications Interface	Chapter 4, *Serial Communications Interface (SCI) Module*
Serial Peripheral Interface	Chapter 5, *Serial Peripheral Interface (SPI) Module*
TMS320C240 DSP Controller	Chapter 11, *TMS320C240 DSP Controller*
Watchdog and Real-Time Interrupt Module	Chapter 6, *Watchdog (WD) and Real-Time Interrupt (RTI) Module*

Notational Conventions

This document uses the following conventions:

❑ Program listings and program examples are shown in a `special typeface`.

Here is a segment of a program listing:

```
OUTPUT  LDP     #6         ;select data page 6
        BLDD    #300, 20h  ;move data at address 300h to 320h
        RET
```

Information About Cautions

This book contains cautions.

CAUTION

This is an example of a caution statement.

A caution statement describes a situation that could potentially damage your software or equipment.

The information in a caution is provided for your protection. Please read each caution carefully.

Related Documentation from Texas Instruments

The following books describe the 'C24x and related support tools. To obtain a copy of any of these TI documents, call the Texas Instruments Literature Response Center at (800) 477–8924. When ordering, please identify the book by its title and literature number.

TMS320C24x DSP Controllers Reference Set, Volume 1: CPU, System, and Instruction Set (literature number SPRU160) describes the TMS320C24x 16-bit, fixed-point, digital signal processor controller. Covered are its architecture, internal register structure, data and program addressing, and instruction set. Also includes instruction set comparisons and design considerations for using the XDS510 emulator.

TMS320C240, TMS320F240 DSP Controllers (literature number SPRS042) data sheet contains the electrical and timing specifications for these devices, as well as signal descriptions and pinouts for all of the available packages.

TMS320C1x/C2x/C2xx/C5x Code Generation Tools Getting Started Guide (literature number SPRU121) describes how to install the TMS320C1x, TMS320C2x, TMS320C2xx, and TMS320C5x assembly language tools and the C compiler for the 'C1x, 'C2x, 'C2xx, and 'C5x devices. The installation for MS-DOS™, OS/2™, SunOS™, and Solaris™ systems is covered.

TMS320C1x/C2x/C2xx/C5x Assembly Language Tools User's Guide (literature number SPRU018) describes the assembly language tools (assembler, linker, and other tools used to develop assembly language code), assembler directives, macros, common object file format, and symbolic debugging directives for the 'C1x, 'C2x, 'C2xx, and 'C5x generations of devices.

TMS320C2x/C2xx/C5x Optimizing C Compiler User's Guide (literature number SPRU024) describes the 'C2x/C2xx/C5x C compiler. This C compiler accepts ANSI standard C source code and produces TMS320 assembly language source code for the 'C2x, 'C2xx, and 'C5x generations of devices.

TMS320C2xx C Source Debugger User's Guide (literature number SPRU151) tells you how to invoke the 'C2xx emulator and simulator versions of the C source debugger interface. This book discusses various aspects of the debugger interface, including window management, command entry, code execution, data management, and breakpoints. It also includes a tutorial that introduces basic debugger functionality.

TMS320C2xx Simulator Getting Started (literature number SPRU137) describes how to install the TMS320C2xx simulator and the C source debugger for the 'C2xx. The installation for MS-DOS™, PC-DOS™, SunOS™, Solaris™, and HP-UX™ systems is covered.

TMS320C2xx Emulator Getting Started Guide (literature number SPRU209) tells you how to install the Windows™ 3.1 and Windows™ 95 versions of the 'C2xx emulator and C source debugger interface.

XDS51x Emulator Installation Guide (literature number SPNU070) describes the installation of the XDS510™, XDS510PP™, and XDS510WS™ emulator controllers. The installation of the XDS511™ emulator is also described.

XDS522/XDS522A Emulation System Installation Guide (literature number SPRU171) describes the installation of the emulation system. Instructions include how to install the hardware and software for the XDS522™ and XDS522A™.

XDS522A Emulation System User's Guide (literature number SPRU169) tells you how to use the XDS522A™ emulation system. This book describes the operation of the breakpoint, tracing, and timing functionality in the XDS522A emulation system. This book also discusses BTT software interface and includes a tutorial that uses step-by-step instructions to demonstrate how to use the XDS522A emulation system.

XDS522A Emulation System Online Help (literature number SPRC002) is an online help file that provides descriptions of the BTT software user interface, menus, and dialog boxes.

JTAG/MPSD Emulation Technical Reference (literature number SPDU079) provides the design requirements of the XDS510™ emulator controller, discusses JTAG designs (based on the IEEE 1149.1 standard), and modular port scan device (MPSD) designs.

TMS320 DSP Development Support Reference Guide (literature number SPRU011) describes the TMS320 family of digital signal processors and the tools that support these devices. Included are code-generation tools (compilers, assemblers, linkers, etc.) and system integration and debug tools (simulators, emulators, evaluation modules, etc.). Also covered are available documentation, seminars, the university program, and factory repair and exchange.

Digital Signal Processing Applications with the TMS320 Family, Volumes 1, 2, and 3 (literature numbers SPRA012, SPRA016, SPRA017) Volumes 1 and 2 cover applications using the 'C10 and 'C20 families of fixed-point processors. Volume 3 documents applications using both fixed-point processors, as well as the 'C30 floating-point processor.

TMS320 DSP Designer's Notebook: Volume 1 (literature number SPRT125) presents solutions to common design problems using 'C2x, 'C3x, 'C4x, 'C5x, and other TI DSPs.

TMS320 Third-Party Support Reference Guide (literature number SPRU052) alphabetically lists over 100 third parties that provide various products that serve the family of TMS320 digital signal processors. A myriad of products and applications are offered—software and hardware development tools, speech recognition, image processing, noise cancellation, modems, etc.

Related Technical Articles

The following technical articles contain beneficial information regarding designs, operations, and applications for signal-processing systems; all of the documents provide additional references.

“A Greener World Through DSP Controllers”, Panos Papamichalis, *DSP & Multimedia Technology*, September 1994.

“A Single-Chip Multiprocessor DSP for Image Processing—TMS320C80”, Dr. Ing. Dung Tu, *Industrie Elektronik*, Germany, March 1995.

“Application Guide with DSP Leading-Edge Technology”, Y. Nishikori, M. Hattori, T. Fukuhara, R.Tanaka, M. Shimoda, I. Kudo, A.Yanagitani, H. Miyaguchi, et al., *Electronics Engineering*, November 1995.

“Approaching the No-Power Barrier”, Jon Bradley and Gene Frantz, *Electronic Design*, January 9, 1995.

“Beware of BAT: DSPs Add Brilliance to New Weapons Systems”, Panos Papamichalis, *DSP & Multimedia Technology*, October 1994.

“Choose DSPs for PC Signal Processing”, Panos Papamichalis, *DSP & Multimedia Technology*, January/February 1995.

“Developing Nations Take Shine to Wireless”, Russell MacDonald, Kara Schmidt and Kim Higden, *EE Times*, October 2, 1995.

“Digital Signal Processing Solutions Target Vertical Application Markets”, Ron Wages, *ECN*, September 1995.

“Digital Signal Processors Boost Drive Performance”, Tim Adcock, *Data Storage*, September/October 1995.

“DSP and Speech Recognition, An Origin of the Species”, Panos Papamichalis, *DSP & Multimedia Technology*, July 1994.

“DSP Design Takes Top-Down Approach”, Andy Fritsch and Kim Asal, *DSP Series Part III*, *EE Times*, July 17, 1995.

“DSPs Advance Low-Cost ‘Green’ Control”, Gregg Bennett, *DSP Series Part II*, *EE Times*, April 17, 1995.

“DSPs Do Best on Multimedia Applications”, Doug Rasor, *Asian Computer World*, October 9–16, 1995.

“DSPs: Speech Recognition Technology Enablers”, Gene Frantz and Gregg Bennett, *I&CS*, May 1995.

“Easing JTAG Testing of Parallel-Processor Projects”, Tony Coomes, Andy Fritsch, and Reid Tatge, *Asian Electronics Engineer*, Manila, Philippines, November 1995.

“Fixed or Floating? A Pointed Question in DSPs”, Jim Larimer and Daniel Chen, *EDN*, August 3, 1995.

“Function-Focused Chipsets: Up the DSP Integration Core”, Panos Papamichalis, *DSP & Multimedia Technology*, March/April 1995.

“GSM: Standard, Strategien und Systemchips”, Edgar Auslander, *Elektronik Praxis*, Germany, October 6, 1995.

“High Tech Copiers to Improve Images and Reduce Paperwork”, Karl Guttag, *Document Management*, July/August 1995.

“Host-Enabled Multimedia: Brought to You by DSP Solutions”, Panos Papamichalis, *DSP & Multimedia Technology*, September/October 1995.

“Integration Shrinks Digital Cellular Telephone Designs”, Fred Cohen and Mike McMahan, *Wireless System Design*, November 1994.

“On-Chip Multiprocessing Melds DSPs”, Karl Guttag and Doug Deao, *DSP Series Part III*, *EE Times*, July 18, 1994.

“Real-Time Control”, Gregg Bennett, *Appliance Manufacturer*, May 1995.

“Speech Recognition”, P.K. Rajasekaran and Mike McMahan, *Wireless Design & Development*, May 1995.

“Telecom Future Driven by Reduced Milliwatts per DSP Function”, Panos Papamichalis, *DSP & Multimedia Technology*, May/June 1995.

“The Digital Signal Processor Development Environment”, Greg Peake, *Embedded System Engineering*, United Kingdom, February 1995.

“The Growing Spectrum of Custom DSPs”, Gene Frantz and Kun Lin, *DSP Series Part II*, *EE Times*, April 18, 1994.

"The Wide World of DSPs, " Jim Larimer, *Design News*, June 27, 1994.

"Third-Party Support Drives DSP Development for Uninitiated and Experts Alike", Panos Papamichalis, *DSP & Multimedia Technology*, December 1994/January 1995.

"Toward an Era of Economical DSPs", John Cooper, *DSP Series Part I*, *EE Times*, Jan. 23, 1995.

Trademarks

HP-UX is a trademark of Hewlett-Packard Company.

MS-DOS and Windows are registered trademarks of Microsoft Corporation.

OS/2, PC, and PC-DOS are trademarks of International Business Machines Corporation.

PAL® is a registered trademark of Advanced Micro Devices, Inc.

Solaris and SunOS are trademarks of Sun Microsystems, Inc.

320 Hotline On-line, TI, XDS510, XDS510PP, XDS510WS, XDS511, XDS522, and XDS522A are trademarks of Texas Instruments Incorporated.

If You Need Assistance. . .

❑ World-Wide Web Sites

TI Online	http://www.ti.com
Semiconductor Product Information Center (PIC)	http://www.ti.com/sc/docs/pic/home.htm
DSP Solutions	http://www.ti.com/dsps
320 Hotline On-line™	http://www.ti.com/sc/docs/dsps/support.htm

❑ North America, South America, Central America

Product Information Center (PIC)	(972) 644-5580		
TI Literature Response Center U.S.A.	(800) 477-8924		
Software Registration/Upgrades	(214) 638-0333	Fax: (214) 638-7742	
U.S.A. Factory Repair/Hardware Upgrades	(281) 274-2285		
U.S. Technical Training Organization	(972) 644-5580		
DSP Hotline	(281) 274-2320	Fax: (281) 274-2324	Email: dsph@ti.com
DSP Modem BBS	(281) 274-2323		

DSP Internet BBS via anonymous ftp to ftp://ftp.ti.com/pub/tms320bbs

❑ Europe, Middle East, Africa

European Product Information Center (EPIC) Hotlines:

Multi-Language Support	+33 1 30 70 11 69	Fax: +33 1 30 70 10 32	Email: epic@ti.com
Deutsch	+49 8161 80 33 11 or +33 1 30 70 11 68		
English	+33 1 30 70 11 65		
Francais	+33 1 30 70 11 64		
Italiano	+33 1 30 70 11 67		
EPIC Modem BBS	+33 1 30 70 11 99		
European Factory Repair	+33 4 93 22 25 40		
Europe Customer Training Helpline		Fax: +49 81 61 80 40 10	

❑ Asia-Pacific

Literature Response Center	+852 2 956 7288	Fax: +852 2 956 2200
Hong Kong DSP Hotline	+852 2 956 7268	Fax: +852 2 956 1002
Korea DSP Hotline	+82 2 551 2804	Fax: +82 2 551 2828
Korea DSP Modem BBS	+82 2 551 2914	
Singapore DSP Hotline		Fax: +65 390 7179
Taiwan DSP Hotline	+886 2 377 1450	Fax: +886 2 377 2718
Taiwan DSP Modem BBS	+886 2 376 2592	

Taiwan DSP Internet BBS via anonymous ftp to ftp://dsp.ee.tit.edu.tw/pub/TI/

❑ Japan

Product Information Center	+0120-81-0026 (in Japan)	Fax: +0120-81-0036 (in Japan)
	+03-3457-0972 or (INTL) 813-3457-0972	Fax: +03-3457-1259 or (INTL) 813-3457-1259
DSP Hotline	+03-3769-8735 or (INTL) 813-3769-8735	Fax: +03-3457-7071 or (INTL) 813-3457-7071
DSP BBS via Nifty-Serve	Type “Go TIASP”	

❑ Documentation

When making suggestions or reporting errors in documentation, please include the following information that is on the title page: the full title of the book, the publication date, and the literature number.

Mail: Texas Instruments Incorporated
Technical Documentation Services, MS 702
P.O. Box 1443
Houston, Texas 77251-1443

Email: comments@books.sc.ti.com

Note: When calling a Literature Response Center to order documentation, please specify the literature number of the book.

Contents

Section II: Specific TMS320C24x Information

Figures

Tables

Examples

Section I
Peripheral Descriptions

Section II
Specific TMS320C24x Information

Chapter 1

Introduction

The TMS320C24x (’C24x) series is a member of the TMS320 family of digital signal processors (DSPs). The ’C24x series is designed to meet a wide range of digital motor control (DMC) applications. This chapter provides an overview of the current TMS320 family, describes the background and benefits of the ’C24x DSP controller products, and introduces the ’C240 device.

Topic **Page**

1.1 TMS320 Family Overview

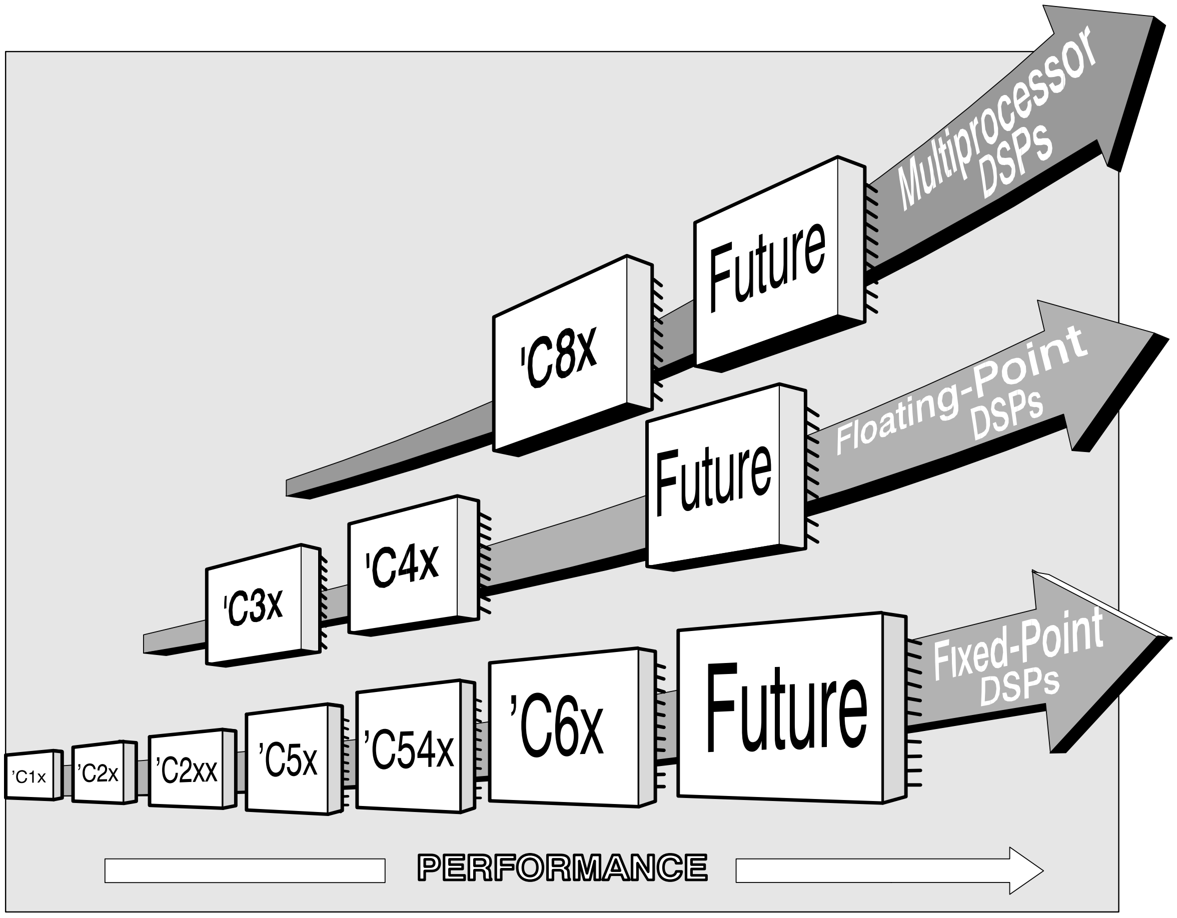
The TMS320 family consists of fixed-point, floating-point, multiprocessor digital signal processors (DSPs), and fixed-point DSP controllers. TMS320 DSPs have an architecture designed specifically for real-time signal processing. The ’C24x series of DSP controllers combines this real-time processing capability with controller peripherals to create an ideal solution for control system applications. The following characteristics make the TMS320 family the right choice for a wide range of processing applications:

- ❑ Very flexible instruction set
- ❑ Inherent operational flexibility
- ❑ High-speed performance
- ❑ Innovative parallel architecture
- ❑ Cost-effectiveness

In 1982, Texas Instruments introduced the TMS32010, the first fixed-point DSP in the TMS320 family. Before the end of the year, *Electronic Products* magazine awarded the TMS32010 the title “Product of the Year”. Today, the TMS320 family consists of these generations (Figure 1–1): ’C1x, ’C2x, ’C2xx, ’C5x, ’C54x, and ’C6x fixed-point DSPs; ’C3x and ’C4x floating-point DSPs; and ’C8x multiprocessor DSPs. The ’C24x is considered part of the ’C2xx family of fixed-point DSPs.

Devices within a generation of the TMS320 family have the same CPU structure but different on-chip memory and peripheral configurations. Spin-off devices use new combinations of on-chip memory and peripherals to satisfy a wide range of needs in the worldwide electronics market. By integrating memory and peripherals onto a single chip, TMS320 devices reduce system costs and save circuit board space.

Figure 1–1. TMS320 Family



1.2 TMS320C24x Series of DSP Controllers

Designers are recognizing the opportunity to redesign existing DMC systems to use advanced algorithms, yielding better performance and reducing system component count. DSPs are enabling:

- ❑ Design of robust controllers for a new generation of inexpensive motors, such as AC induction, DC permanent magnet, and switched-reluctance motors
- ❑ Full variable-speed control of brushless motor types that have lower manufacturing cost and higher reliability
- ❑ Energy savings through variable-speed control, saving up to 25% of the energy used by fixed-speed controllers
- ❑ Increased fuel economy, improved performance, and elimination of hydraulic fluid in automotive electronic power steering (EPS) systems
- ❑ Reduced manufacturing and maintenance costs by eliminating hydraulic fluids in automotive electronic braking systems
- ❑ More efficient and quieter operation due to less generation of torque ripple, resulting in less loss of power, lower vibration, and longer life
- ❑ Elimination or reduction of memory lookup tables through real-time polynomial calculation, thereby reducing system cost
- ❑ Use of advanced algorithms that can reduce the number of sensors required in a system
- ❑ Control of power switching inverters along with control algorithm processing
- ❑ Single-processor control of multimotor systems

The ’C24x DSP controllers are designed to meet the needs of control-based applications. By integrating the high performance of a DSP core and the on-chip peripherals of a microcontroller into a single-chip solution, the ’C24x series yields a device that is an affordable alternative to traditional microcontroller units (MCUs) and expensive multichip designs. At 20 million instructions per second (MIPS), the ’C24x DSP controllers offer significant performance over traditional 16-bit microcontrollers and microprocessors.

The 16-bit, fixed-point DSP core of the ’C24x devices provides analog designers a digital solution that does not sacrifice the precision and performance of their systems. In fact, system performance can be enhanced through the use of advanced control algorithms for techniques such as adaptive control, Kalman filtering, and state control. The ’C24x DSP controllers offer reliability and programmability. Analog control systems, on the other hand, are hard-wired solutions and can experience performance degradation due to aging, component tolerance, and drift.

The high-speed central processing unit (CPU) allows the digital designer to process algorithms in real time rather than approximate results with look-up tables. The instruction set of these DSP controllers, which incorporates both signal processing instructions and general-purpose control functions, coupled with the extensive development support available for the ’C24x devices, reduces development time and provides the same ease of use as traditional 8- and 16-bit microcontrollers. The instruction set also allows you to retain your software investment when moving from other general-purpose TMS320 fixed-point DSPs. It is source- and object-code compatible with the other members of the ’C2xx generation, source code compatible with the ’C2x generation, and upward source code compatible with the ’C5x generation of DSPs from Texas Instruments.

The ’C24x architecture is also well-suited for processing control signals. A 16-bit word length is used along with 32-bit registers for storing intermediate results, and two hardware shifters are available to scale numbers independently of the CPU. This combination minimizes quantization and truncation errors, and increases processing power for additional functions. Such functions might include a notch filter that could cancel mechanical resonances in a system or an estimation technique that could eliminate state sensors in a system.

The ’C24x DSP controllers take advantage of an existing set of peripheral functions that allow Texas Instruments to quickly configure various series members for different price/performance points or for application optimization. This library of both digital and mixed-signal peripherals includes:

- ❑ Timers
- ❑ Serial communications ports (SCI, SPI)
- ❑ Analog-to-digital converters (ADC)
- ❑ Event manager
- ❑ System protection, such as low-voltage detection and watchdog timers

The DSP controller peripheral library is continually growing and changing to suit the needs of tomorrow’s embedded control marketplace.

1.3 TMS320C240 Overview

The TMS320C240 is the first standard device introduced in the ’C24x series of DSP controllers. It sets the standard for a single-chip digital motor controller. The ’C240 can execute 20 MIPS. Almost all instructions are executed in a single cycle of 50 ns. This high performance allows real-time execution of very complex control algorithms, such as adaptive control and Kalman filters. Very high sampling rates can also be used to minimize loop delays.

The ’C240 has the architectural features necessary for high-speed signal processing and digital control functions, and it has the peripherals needed to provide a single-chip solution for motor control applications. The ’C240 is manufactured using submicron CMOS technology, achieving a low power dissipation rating. Also included are several power-down modes for further power savings.

Applications that benefit from the advanced processing power of the ’C240 include:

❑ Industrial motor drives

❑ Power inverters and controllers

❑ Automotive systems, such as electronic power steering, anti-lock brakes, and climate control

❑ Appliance and HVAC blower/compressor motor controls

❑ Printers, copiers, and other office products

❑ Tape drives, magnetic optical drives, and other mass storage products

❑ Robotics and CNC milling machines

To function as a system manager, DSPs must have robust on-chip I/O and other peripherals. The event manager of the ’C240 is unlike any other available on a DSP. This application-optimized peripheral unit, coupled with the high-performance DSP core, enables the use of advanced control techniques for high-precision and high-efficiency full variable-speed control of all motor types. Included in the event manager are special pulse-width modulation (PWM) generation functions, such as a programmable dead-band function and a space vector PWM state machine for three-phase motors that provides state-of-the-art maximum efficiency in the switching of power transistors. Three independent up/down timers, each with it’s own compare register, support the generation of asymmetric (noncentered) as well as symmetric (centered) PWM waveforms. Two of the four capture inputs are direct connections for quadrature encoder pulse signals from an optical encoder.

Here is a summary of ’C240 features:

- TMS320C2xx core CPU:
  - 32-bit central arithmetic logic unit (CALU)
  - 32-bit accumulator
  - 16-bit $\times$ 16-bit parallel multiplier with a 32-bit product capability
  - Three scaling shifters
  - Eight 16-bit auxiliary registers with a dedicated arithmetic unit for indirect addressing of data memory
- Memory:
  - 544 words $\times$ 16 bits of on-chip data/program dual-access RAM
  - 16K words $\times$ 16 bits of on-chip program ROM or flash EEPROM
  - 224K words $\times$ 16 bits of maximum addressable memory space (64K words of program space, 64K words of data space, 64K words of I/O space, and 32K words of global space)
  - External Memory Interface Module with a software wait-state generator, a 16-bit address bus, and a 16-bit data bus
  - Support of hardware wait-states
- Program control:
  - Four-level pipeline operation
  - Eight-level hardware stack
  - Six external interrupts: power-drive protection interrupt, reset, NMI, and three maskable interrupts
- Instruction set:
  - Source code compatibility with ’C2x, ’C2xx, and ’C5x fixed-point generations of the TMS320 family
  - Single-instruction repeat operation
  - Single-cycle multiply/accumulate instructions
  - Memory block move instructions for program/data management
  - Indexed-addressing capability
  - Bit-reversed indexed-addressing capability for radix-2 fast Fourier transforms (FFTs)
- Power:
  - Static CMOS technology
  - Four power-down modes to reduce power consumption

- ❑ Emulation: IEEE Standard 1149.1 test access port interface to on-chip scan-based emulation logic
- ❑ Speed: 50-ns (20 MIPS) instruction cycle time, with most instructions single-cycle
- ❑ Event manager:
  - ■ 12 compare/pulse-width modulation (PWM) channels (9 independent)
  - ■ Three 16-bit general-purpose timers with six modes, including continuous up counting and continuous up/down counting
  - ■ Three 16-bit full compare units with dead band capability
  - ■ Three 16-bit simple compare units
  - ■ Four capture units, two of which have quadrature encoder-pulse interface capability
- ❑ Dual 10-bit analog-to-digital converter
- ❑ 28 individually programmable, multiplexed I/O pins
- ❑ Phase-locked loop (PLL)-based clock module
- ❑ Watchdog timer module with real-time interrupt
- ❑ Serial communication interface (SCI)
- ❑ Serial peripheral interface (SPI)

Chapter 2

Event Manager (EV) Module

This chapter describes the Event Manager (EV) module in the peripheral library. The device pins used by the EV module can be shared between EV and other modules. See chapters on the individual devices for device specific details.

Topic **Page**

2.1 Event Manager (EV) Module Overview

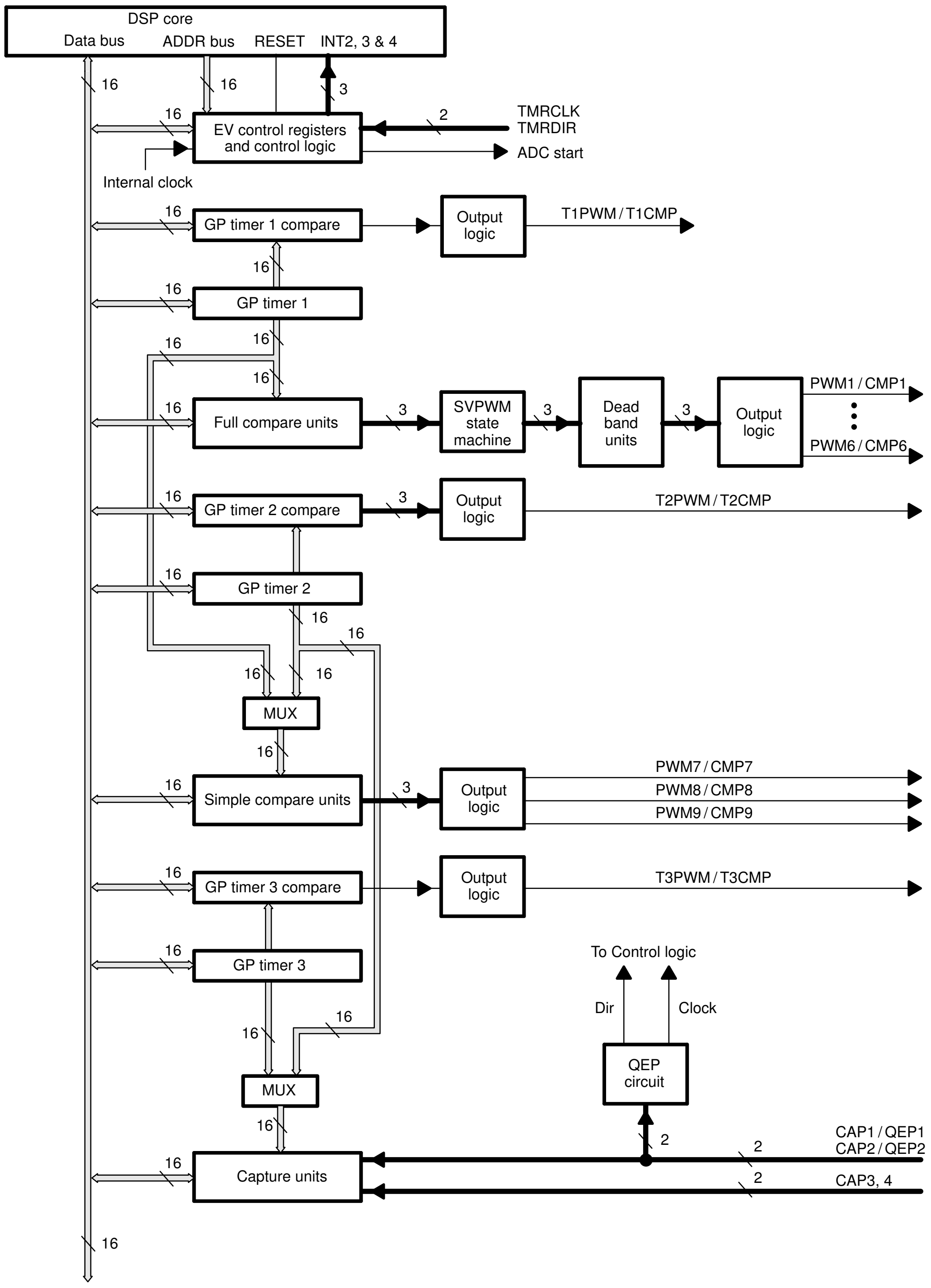
The Event Manager (EV) module provides a broad range of functions and features that are particularly useful in motion control and motor control applications.

2.1.1 EV Functional Blocks

Figure 2–1 shows a block diagram of the EV module. The EV module contains the following functional blocks:

- ❑ Three general purpose (GP) timers (described in Section 2.3 on page 2-11).
- ❑ Three full compare units (described in Section 2.4 on page 2-41).
- ❑ Three simple compare units (described in Section 2.4 on page 2-41).
- ❑ Pulse-width modulation (PWM) circuits that include space vector PWM circuit, dead-band generation units, and output logic (described in Section 2.5 on page 2-52, Section 2.6 on page 2-61, and Section 2.7 on page 2-66).
- ❑ Four capture units (described in Section 2.8 on page 2-72).
- ❑ Quadrature encoder pulse (QEP) circuit (described in Section 2.9 on page 2-83).
- ❑ Interrupt logic.

Figure 2–1. Event Manager (EV) Block Diagram



2.1.2 EV Pins

The EV module uses 12 device pins for compare/PWM outputs:

- Three GP timer compare/PWM output pins:
  - T1PWM/T1CMP
  - T2PWM/T2CMP
  - T3PWM/T3CMP
- Six full compare/PWM output pins:
  - PWM1/CMP1
  - PWM2/CMP2
  - PWM3/CMP3
  - PWM4/CMP4
  - PWM5/CMP5
  - PWM6/CMP6
- Three simple compare/PWM output pins:
  - PWM7/CMP7
  - PWM8/CMP8
  - PWM9/CMP9

The EV module uses 4 device pins, CAP1/QEP1, CAP2/QEP2, CAP3, and CAP4, as capture or quadrature encoder pulse inputs.

The GP timers in the EV module can be programmed to operate based on external or internal CPU clock. The device pin TMRCLK supplies the external clock input.

The device pin TMRDIR is used to specify the counting direction when a GP timer is in directional-up/down counting mode.

All inputs to the EV module are synchronized with the internal CPU clock. A transition must hold until two rising edges of the CPU clock (that is, two falling edges of CLKOUT if the CPU clock has been chosen to be the source for CLKOUT output) are met before it is recognized by the EV. Therefore, it is recommended that any transition be held for more than two CPU clock cycles.

The device pins are summarized in Table 2–1.

Table 2–1. Event Manager Pins

Pin Name	Description
CAP1/QEP1	Capture Unit 1 input, QEP circuit input 1
CAP2/QEP2	Capture Unit 2 input, QEP circuit input 2
CAP3	Capture Unit 3 input
CAP4	Capture Unit 4 input
PWM1/CMP1	Full Compare Unit 1 compare/PWM output 1
PWM2/CMP2	Full Compare Unit 1 compare/PWM output 2
PWM3/CMP3	Full Compare Unit 2 compare/PWM output 1
PWM4/CMP4	Full Compare Unit 2 compare/PWM output 2
PWM5/CMP5	Full Compare Unit 3 compare/PWM output 1
PWM6/CMP6	Full Compare Unit 3 compare/PWM output 2
PWM7/CMP7	Simple Compare Unit 1 compare/PWM output 1
PWM8/CMP8	Simple Compare Unit 2 compare/PWM output 2
PWM9/CMP9	Simple Compare Unit 3 compare/PWM output 3
T1PWM/T1CMP	GP Timer 1 compare/PWM output
T2PWM/T2CMP	GP Timer 2 compare/PWM output
T3PWM/T3CMP	GP Timer 3 compare/PWM output
TMRCLK	External clock input for GP Timers
TMRDIR	External timer direction input

2.1.3 Power Drive Protection

An external interrupt is generated when the device pin PDPINT (Power Drive Protection Interrupt) is pulled low. This interrupt is provided for the safe operation of systems such as power converters and motor drives. If PDPINT is unmasked, all EV output pins will be put in a high-impedance state by hardware immediately after the PDPINT pin is pulled low. The interrupt flag associated with PDPINT will also be set when such an event happens; however, it must wait until the transition on PDPINT has been qualified and synchronized with the internal CPU clock. The qualification and synchronization causes a delay of three to four CPU clock cycles. If PDPINT is unmasked, the flag will keep the EV outputs in a high-impedance state and generate an interrupt request to the DSP core. Therefore, in order for the outputs to remain in high impedance, PDPINT should be held low for longer than four CPU clock cycles. The setting of the flag does not depend on whether PDPINT is masked. It will happen as long as a qualified transition has occurred on PDPINT pin. PDPINT can be used to inform the monitoring program of motor drive abnormalities such as overvoltage, overcurrent, and excessive temperature rise.

2.1.4 EV Registers

All registers in the EV module are mapped to data memory. Their addresses take up 64 (16-bit) words of the 64K words of data memory address range, which are from 7400h to 743Fh. The registers are treated by software as data memory locations and can be accessed by a wide range of DSP instructions. The undefined bits of the EV registers all return zero when read by user software. See Section 2.2, *Event Manager (EV) Register Addresses*, on page 2-8.

2.1.5 EV Interrupts

The interrupts generated by the EV module are arranged into three groups. There are three registers, EVIFRA, EVIFRB, and EVIFRC, in the EV for the flags of the three groups of interrupts. The three groups of interrupts are connected to three of the CPU interrupt inputs. Each EV interrupt group has an associated interrupt vector register, EVIFVRx (x = A, B, and C), that can be read by user software to get the vector (ID) of the interrupt source. Each interrupt source has a unique interrupt vector ID.

An interrupt flag is set when an interrupt event (for example, match between a compare register and a GP timer) is generated and enabled. There is an interrupt mask register, EVIMRx (x = A, B, and C), associated with each EV interrupt group. An interrupt is masked if its corresponding bit in EVIMRA, EVIMRB, or EVIMRC is 0 and unmasked if the corresponding bit is 1. An interrupt request will be generated by a flag if the flag is set, unmasked, and no other unmasked interrupts in the same group of higher priority are pending.

The set and reset of interrupt flags, however, are independent of whether the corresponding interrupts are masked. The interrupt flag can, therefore, be checked by software to verify the occurrence of an event when the corresponding interrupt is masked. An interrupt flag can be reset to 0 in two ways:

1) User software writes a 1 to the corresponding bit in EVIFRA, EVIFRB, or EVIFRC.
2) User software reads its interrupt vector ID after an interrupt request generated by its group has been taken.

The vector ID of the flag that has the highest priority among the flags that are set in the group will be loaded into the accumulator when the interrupt vector is read after an interrupt request generated by the group has been taken. A zero value will be returned when the interrupt vector register of an interrupt group is read and no interrupt flag in the group is set. This prevents a strayed interrupt from being recognized as an EV interrupt. The details of interrupt event generation are described in the following sections. The details of request generation, service, and priority of all EV interrupts are summarized in Section 2.10, *Event Manager (EV) Interrupts*, on page 2-87.

2.2 Event Manager (EV) Register Addresses

Table 2–2 through Table 2–5 display the addresses of the Event Manager registers.

Table 2–2. Addresses of GP Timer Registers

			Described in	
Address	**Register**	**Name**	**Section**	**Page**
7400h	GPTCON	General purpose timer control register.	2.3.3	2-36
7401h	T1CNT	GP Timer 1 counter register.	2.3	2-11
7402h	T1CMPR	GP Timer 1 compare register.	2.3	2-11
7403h	T1PR	GP Timer 1 period register.	2.3	2-11
7404h	T1CON	GP Timer 1 control register.	2.3.3	2-36
7405h	T2CNT	GP Timer 2 counter register.	2.3	2-11
7406h	T2CMPR	GP Timer 2 compare register.	2.3	2-11
7407h	T2PR	GP Timer 2 period register.	2.3	2-11
7408h	T2CON	GP Timer 2 control register.	2.3.3	2-36
7409h	T3CNT	GP Timer 3 counter register.	2.3	2-11
740Ah	T3CMPR	GP Timer 3 compare register.	2.3	2-11
740Bh	T3PR	GP Timer 3 period register.	2.3	2-11
740Ch	T3CON	GP Timer 3 control register.	2.3.3	2-36

Table 2–3. Addresses of Full and Simple Compare Unit Registers

			Described in	
Address	**Register**	**Name**	**Section**	**Page**
7411h	COMCON	Compare control register.	2.4.3	2-45
7413h	ACTR	Full compare action control register.	2.4.3	2-45
7414h	SACTR	Simple compare action control register.	2.4.3	2-45
7415h	DBTCON	Dead-band timer control register.	2.5.2	2-53
7417h 7418h 7419h	CMPR1 CMPR2 CMPR3	Full compare unit compare registers.	2.4	2-41
741Ah 741Bh 741Ch	SCMPR1 SCMPR2 SCMPR3	Simple compare unit compare registers.	2.4	2-41

Table 2–4. Addresses of Capture Unit and Quadrature Encoder Pulse Decoding Circuit Registers

			Described in	
Address	**Register**	**Name**	**Subsection**	**Page**
7420h	CAPCON	Capture control register.	2.8.3	2-75
7422h	CAPFIFO	Capture FIFO status register.	2.8.3	2-75
7423h 7424h 7425h 7426h	CAP1FIFO CAP2FIFO CAP3FIFO CAP4FIFO	Two-level deep FIFO stacks.	2.8.4	2-79

Table 2–5. Addresses of EV Interrupt Registers

			Described in	
Address	Register	Name	Subsection	Page
742Ch 742Dh 742Eh	EVIMRA EVIMRB EVIMRC	Interrupt mask registers.	2.10.3	2-89
742Fh 7430h 7431h	EVIFRA EVIFRB EVIFRC	Interrupt flag registers.	2.10.3	2-89
7432h 7433h 7434h	EVIVRA EVIVRB EVIVRC	Interrupt vector registers.	2.10.3	2-89

2.3 General Purpose (GP) Timer

There are three general purpose (GP) timers in the EV module. These timers can be used as independent time bases in applications such as:

❑ Generation of sampling period in a control system.

❑ Providing time base for the operation of QEP circuit and capture units.

❑ Providing time base for the operation of full and simple compare units and associated PWM circuits to generate compare/PWM outputs.

GP Timer Functional Blocks

Figure 2–2 shows a block diagram of the GP timer. Each GP timer includes:

❑ One read- and write-able (R/W) 16-bit up and up/down counter, TxCNT (x = 1, 2, 3).

❑ One R/W 16-bit timer compare register (shadowed), TxCMPR (x = 1, 2, 3).

❑ One R/W 16-bit timer period register (shadowed), TxPR (x = 1, 2, 3).

❑ R/W 16-bit control register, TxCON (x = 1, 2, 3).

❑ Programmable prescaler applicable to both internal and external clock inputs.

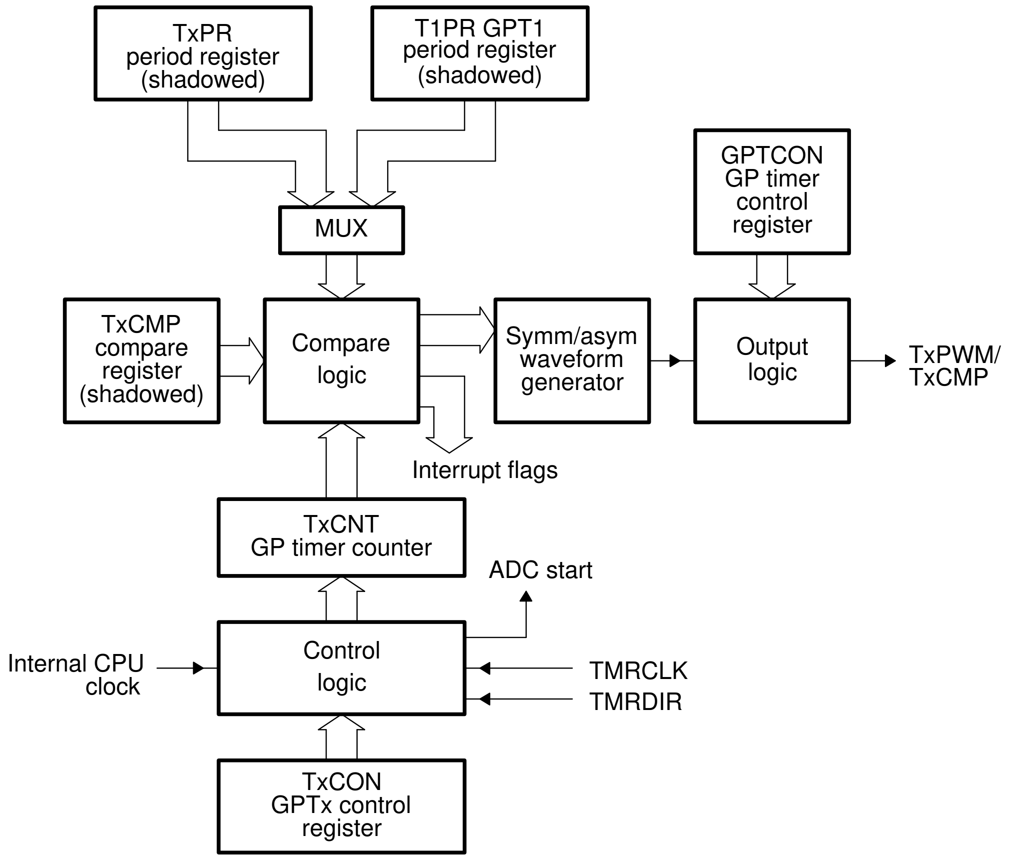
❑ Control and interrupt logic.

❑ One GP timer compare output pin, TxPWM/TxCMP (x = 1, 2, 3).

❑ Output logic.

Another control register, GPTCON specifies the action to be taken by the GP timers on different timer events and indicates the counting directions of all three GP timers. GPTCON is read- and write-able, though writing to the three status bits takes no effect.

Figure 2–2. General Purpose (GP) Timer Block Diagram (x = 1, 2, or 3)



GP Timer Inputs

The inputs to GP timers are:

❑ Internal CPU clock, which comes directly from the core and hence has the same frequency as that of the CPU clock.

❑ External clock, TMRCLK, which has a maximum frequency of one-fourth of that of the CPU clock.

❑ Direction input, TMRDIR, for use by the GP timer in directional-up/down counting mode.

❑ Reset signal, RESET.

In addition, GP Timer 3 takes the overflow of GP Timer 2 as the input clock when GP Timers 2 and 3 are cascaded into a 32-bit timer. When a GP timer is used with the QEP circuit, the QEP circuit generates both the timer's clock and counting direction.

GP Timer Outputs

The outputs of GP timers are:

- ❑ GP timer compare outputs TxPWM/TxCMP, x = 1, 2, 3.
- ❑ ADC start signal to ADC module.
- ❑ Underflow, overflow, compare match, and period match signals to its own compare logic and to full and simple compare units.
- ❑ Counting direction indication bits.

Control of GP Timer Operation

The operation mode of a GP timer is controlled by its control register TxCON. Bits in the TxCON register determine:

- ❑ Which of the 6 counting modes the GP Timer is in.
- ❑ Whether internal or external CPU clock is to be used by the GP Timer.
- ❑ Which of the 6 prescale factors for input clock (ranging from 1 to 1/128) is to be used.
- ❑ On which condition the timer compare register is reloaded.
- ❑ Whether the timer is enabled or disabled.
- ❑ Whether the GP timer compare operation is enabled or disabled.
- ❑ Whether the period register of GP Timer 1 or its own period register determines its period (T2CON and T3CON only).
- ❑ Whether carryover of GP Timer 2 should be used as its clock (T3CON only).

GP Timer Control Register (GPTCON)

The control register GPTCON specifies the action to be taken by the GP Timers on different timer events and indicates their counting directions.

GP Timer Compare Registers

The compare register associated with a GP timer stores the value to be constantly compared with the counter of the GP timer. When a match happens, a transition occurs on the associated compare output according to the bit pattern in GPTCON, the corresponding interrupt flag is set, and an interrupt request to the core is generated if the interrupt is unmasked and no other unmasked interrupts in the same group of higher priority are pending. The compare operation of a GP Timer can be enabled or disabled by appropriate bit in TxCON.

GP Timer Period Register

The value in the period register of a GP timer determines the period of the timer. The operation of a GP timer stops and holds at its current value, resets to 0, or starts counting downward when a match occurs between the period register and the timer counter depending on what counting mode the timer is in.

Double Buffering of GP Timer Compare and Period Registers

The compare and period registers, TxCMPR and TxPR, of a GP timer are shadowed. A new value can be written to any of these registers at any time during a period. However, the new value is written to the associated shadow register. For the compare register, the content in the shadow register is loaded into the working (active) register only when a certain timer event specified by TxCON occurs. For the period register, the working register is reloaded with the value in its shadow register only when the value of the counter register TxCNT is 0. The condition on which a compare register is reloaded can be one of the following:

❏ Immediately after the shadow register is written.

❏ On underflow; that is, when the GP timer counter value is 0.

❏ On underflow or period match; that is, when the counter value is 0 or when the counter value equals the value of the period register.

The double buffering feature of the period and compare registers allows the application code to update the period and compare registers at any time during a period in order to change the timer period and the width of PWM pulse for the following period. On-the-fly change of the timer period value, in the case of PWM generation, means on-the-fly change of PWM carrier frequency.

CAUTION

Initialize period register

The period register of a GP timer should be initialized before its counter is initialized to a nonzero value. Otherwise, the value of the period register will remain unchanged until the next underflow.

Note:

Also, a compare register is transparent (the newly loaded value goes directly into the active register) when the associated compare operation is disabled. This applies to all the compare registers in the EV.

GP Timer Compare Output

The compare output of a GP timer can be specified to be active high, active low, forced high, or forced low, depending on how GPTCON bits are configured. It goes from low to high (high to low) on the first compare match when it is active high (low). It then goes from high to low (low to high) on the second compare match if the GP timer is in up/down counting modes, or on period match if the GP timer is in up counting modes. The timer compare output becomes high (low) right away when it is specified to be forced high (low).

GP Timer Counting Direction

The counting directions of all three GP timers are reflected by respective bits in the GPTCON during all timer operations with:

- ❑ 1 representing the up counting direction.
- ❑ 0 representing the down counting direction.

The input pin TMRDIR determines the direction of counting when a GP timer is in directional-up/down counting mode. When TMRDIR is set high, upward counting is specified; when TMRDIR is pulled low, downward counting is specified.

GP Timer Clock

The source of the GP timer clock can be the internal CPU clock or external clock input, TMRCLK. The frequency of the external clock must be less than or equal to one-fourth of that of the CPU clock. GP Timer 2, 3, or 2 and 3 together as a 32-bit timer can be used with QEP circuits, in directional-up/down counting mode. In this case, the QEP circuits provide both the clock and direction inputs to the timer.

A wide range of prescale factors are provided for the clock input to each GP timer.

32-Bit Timer

The roll-over signal of GP Timer 2 is used as clock input to GP Timer 3 when GP Timers 2 and 3 are cascaded into a 32-bit timer. When this happens, the counter of GP Timer 2 will act as the lower 16-bits of the 32-bit counter. The 32-bit timer so obtained, can only operate in directional-up/down counting mode (see the operating modes of the GP timers) with external or internal clock and external direction inputs. QEP circuits can also be chosen to generate the counting clock and direction for the 32-bit timer. The period registers of GP Timers 2 and 3 are cascaded in this case to provide a 32-bit period register for the 32-bit timer, while the compare operation is based on individual compare registers and occurs individually on 16-bit compare matches. Both underflow and overflow events for the 32-bit timer are 32-bit based.

QEP Based Clock Input

The quadrature encoder pulse (QEP) circuit, when selected, can generate the input clock and counting direction for GP Timer 2, 3, or 2 and 3 together as a 32-bit timer in the directional-up/down counting mode. This input clock can not be scaled by GP timer prescaler circuits (that is, the prescaler of the selected GP timer is always 1 if the QEP circuit is selected as the clock source). Furthermore, the frequency of the clock generated by QEP circuits is four times that of the frequency of each QEP input channel because both the rising and falling edges of both QEP input channels are counted by the selected timer. The frequency of QEP input must be less than or equal to one-fourth of that of the CPU clock.

GP Timer Synchronization

GP Timers 2 and 3 can be individually synchronized with GP Timer 1 by proper configuration of T2CON and T3CON in the following ways:

❑ Start the operation of GP Timer 2 or 3 by the same control bit in T1CON that starts the operation of GP Timer 1.

❑ Initialize the timer counters in GP Timer 2 or 3 with different values before synchronized operation starts.

❑ Specify that GP Timer 2 or 3 uses the period register of GP Timer 1 as its period register (ignoring its own period register).

This allows desired synchronization between GP timer events. Since each GP timer starts counting operation from its current value in the counter register, a GP timer can be programmed to start with a known delay after other GP timers. Note, however, two writes to T1CON are required to synchronize GP Timer 1 with GP Timer 2 or GP Timers 2 and 3.

ADC Start by GP Timer Event

The bits in GPTCON can specify that an ADC start signal be generated on a GP Timer event such as underflow, compare match or period match. This feature provides synchronization between the GP Timer event and the ADC start without any CPU intervention.

GP Timer in Emulation Suspend

GP timer control register bits also define the operation of GP timers during emulation suspend. These bits can be set to allow the operation of GP timers to continue when emulation interrupt occurs making in-circuit emulation possible. They can also be set to specify that the operation of GP timers stops immediately or after completion of the current counting period when emulation interrupt occurs.

Emulation suspend occurs when the CPU clock is stopped by the emulator, for example, when the emulator encounters a break point.

GP Timer Interrupts

There are 12 interrupt flags in EVIFRA and EVIFRB for the three GP timers. Each GP timer can generate four interrupts upon the following events:

- Overflow: TxOFINT (x = 1, 2, or 3)
- Underflow: TxUFINT (x = 1, 2, or 3)
- Compare match: TxCINT (x = 1, 2, or 3)
- Period match: TxPINT (x = 1, 2, or 3)

A timer compare event (match) happens when the content of a GP timer counter is the same as that of the compare register. The corresponding compare interrupt flag is set two CPU clock cycles after the match if the compare operation is enabled.

An overflow event occurs when the value of the timer counter reaches FFFFh. An underflow event occurs when the timer counter reaches 0000h. Similarly, a period event happens when the value of the timer counter is the same as that of the period register. The overflow, underflow, and period interrupt flags of the timer are set two CPU clock cycles after the occurrence of each individual event. Note the definition of overflow and underflow is different from their conventional definition.

2.3.1 GP Timer Counting Operation

Each GP timer has 6 selectable modes of operation:

- Stop/Hold mode
- Single-Up counting mode
- Continuous-Up counting mode
- Directional-Up/Down counting mode
- Single-Up/Down counting mode
- Continuous-Up/Down counting mode

The bit pattern in the corresponding timer control register TxCON determines the counting mode of a GP timer. The timer enabling bit, TxCON[6], enables or disables the counting operation of a timer. When the timer is disabled, the counting operation of the timer stops and the prescaler of the timer is reset to x/1. When the timer is enabled, the timer starts counting according to the counting mode specified by other bits of TxCON.

Stop/Hold Mode

The operation of GP timer stops and holds at its current state. The timer counter, the compare output, and the prescale counter all remain unchanged in this mode.

Single-Up Counting Mode

The GP timer in this mode counts up according to the scaled input clock until the value of the timer counter matches that of the period register. On the next rising edge of the input clock after the match, the GP timer will reset to 0 and disable its counting operation by resetting the timer disabling bit, TxCON[6].

The period interrupt flag of the timer is set two CPU clock cycles after the match between the timer counter and period register. An interrupt request is generated by the flag if it is unmasked and no other unmasked interrupts in the same group of higher priority are pending. An ADC start is sent to the ADC module at the same time the flag is set, if the period interrupt of this timer has been selected by appropriate bits in GPTCON to start ADC.

Two clock cycles after the GP timer becomes 0, the underflow interrupt flag of the timer is set. An interrupt request is generated by the flag if it is unmasked and no other unmasked interrupts in the same group of higher priority are pending. An ADC start is sent to the ADC module at the same time if the underflow interrupt flag of this timer has been selected by appropriate bits in GPTCON to start ADC.

The overflow interrupt flag is set two CPU clock cycles after the value in TxCNT matches FFFFh. An interrupt request is generated by the flag if it is unmasked and no other unmasked interrupts in the same group of higher priority are pending.

The duration of the counting period of this mode is (TxPER) + 1 cycles of the scaled clock input if the timer counter is 0 at the beginning of the period.

The initial value of the GP timer can be any value between 0h and FFFFh. When the initial value is greater than the value in the period register, the timer will count up to FFFFh, reset to 0, and count up again to finish the period as if the initial counter value were 0. When the initial value in the timer counter is the same as that of the period register, the timer will set the period interrupt flag, reset to 0, set the underflow interrupt flag, and immediately end the period. If the initial value of the timer is between 0 and the contents of the period register, the timer will count up to the period value and continue to finish the period as if the initial counter value were the same as that of the period register.

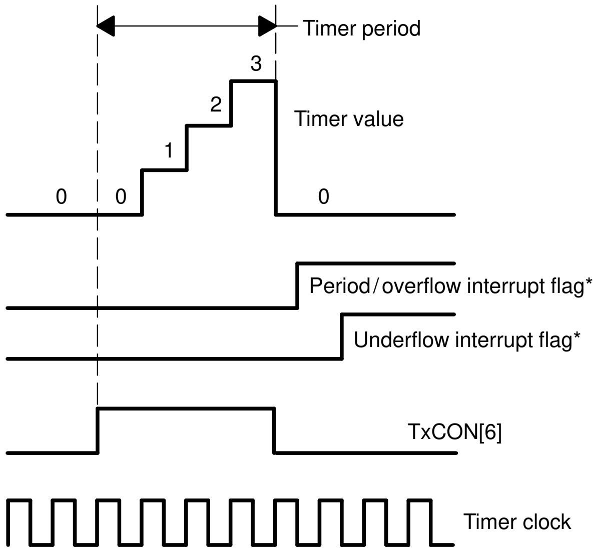
Once the period ends, the operation of the GP timer can only be started again by software writing to the timer enabling bit, TxCON[6].

The counting direction indication bit in GPTCON is 1 for the timer in this mode of operation. Either external or internal CPU clock can be used as input clock to the timer. The TMRDIR pin is ignored by the GP timer in this mode of operation.

Figure 2–3 shows the single-up counting mode of the GP timer.

Note that the GP timer starts counting immediately after TxCON[6] is set. This is true for every counting mode.

Figure 2–3. GP Timer Single-Up Counting Mode (TxPR = 4 – 1 = 3)



*Timing of interrupt flags is the same among all counting modes.

Example 2–1 shows an initialization routine for the single-up counting mode for GP timer 1.

Example 2–1. Initialization Routine for Single-Up Counting Mode

```
;***************************************************************************
; The following section of code initializes GP Timer1 to run in single up *
; count mode. GP Timer compare function is also enabled.                 *
;***************************************************************************
             LDPK   #232                             ; DP => EV Registers
;***************************************************************************
; Configure GPTCON
;***************************************************************************
             SPLK   #0000000001000001b, GPTCON       ; Set GP Timer control
;                    ||||||||||||||||
;                    FEDCBA9876543210
;
* bits 0-1           01:    GP Timer 1 comp output active low
* bits 2-3           00:    GP Timer 2 comp output forced low
* bits 4-5           00:    GP Timer 3 comp output forced low
* bit 6              1:     Enable GP Timer Compare outputs
* bits 7-8           00:    No GP Timer 1 event starts ADC
* bits 9-9           00:    No GP Timer 2 event starts ADC
* bits 10-11         00:    No GP Timer 3 event starts ADC
;***************************************************************************
; Configure T1PER and T1CMP                                               *
;***************************************************************************
             SPLK   #05, T1PER                       ; Set GP Timer period
             SPLK   #03, T1CMP                       ; Set GP Timer compare
;***************************************************************************
; Configure T1CON and start GP Timer 1                                    *
;***************************************************************************
             SPLK   #1000100101000010b, T1CON        ; Set GP Timer 3 control
;                    ||||||||||||||||
;                    FEDCBA9876543210
;
* bit 0              0:     Use own PR
* bit 1              1:     GP Timer compare enabled
* bits 2-3           00:    Load GP Timer comp register on under flow
* bits 4-5           00:    Select internal CLK
* bit 6              1:     Timer (counting operation) enabled
* bit 7              0:     Use own Timer ENABLE
* bits 8-10          001:   Prescaler = /2
* bits 11-13         001:   Single up count
* bit 14             0:     SOFT = 0
* bit 15             1:     FREE = 1
```

Continuous-Up Counting Mode

The operation of the GP timer in this mode is the same as the single-up counting mode repeated each time the timer is reset to 0. The GP timer in this mode will count up according to the scaled input clock until the value in its counter is equal to that of the period register. It then resets to 0 and starts another counting period.

The duration of the timer period is (TxPR) + 1 cycles of the scaled clock input except for the first period. The duration of the first period is the same if the timer counter is 0 when counting starts.

The initial value of the GP timer can be any value between 0h and FFFFh. When the initial value is greater than the value in period register, the timer will count up to FFFFh, reset to 0, and continue the operation as if the initial value were 0. When the initial value in the timer counter is the same as that of the period register, the timer will set the period interrupt flag, reset to 0, set the underflow interrupt flag, and then continue the operation again as if the initial value were 0. If the initial value of the timer is between 0 and the contents of the period register, the timer will count up to the period value and continue to finish the period as if the initial counter value were the same as that of the period register.

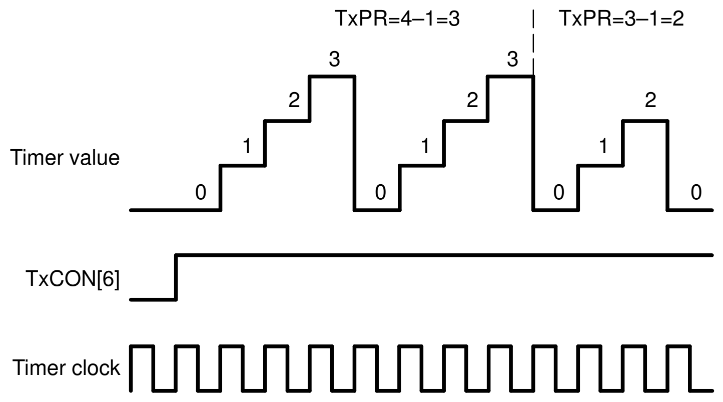
The period, underflow and overflow interrupt flags, interrupts, and associated actions are generated on respective matches in the same manner as they are generated in single-up counting mode.

The counting direction indication bit in GPTCON is 1 for the timer in this mode. Either external or internal CPU clock can be selected as input clock to the timer. TMRDIR input is ignored by the GP timer in this counting mode.

The continuous-up counting mode of the GP timer is particularly useful for the generation of edge-triggered or asynchronous PWM waveforms and sampling periods in many motor and motion control systems.

Figure 2–4 shows the continuous-up counting mode of the GP timer.

Figure 2–4. GP Timer Continuous-Up Counting Mode (TxPR = 3 or 2)



As shown in Figure 2–4, no clock cycle is missed from the time the counter reaches the period register value to the time it starts another counting cycle.

Example 2–2 shows an initialization routine for the continuous-up counting mode for GP timer 1.

Example 2–2. Initialization Routine for Continuous-Up Counting Mode

```
;*****************************************************************************
; The following section of code initializes GP Timer1 to run in continuous up *
; count mode. GP Timer compare function is also enabled. The counter is       *
; initially loaded with FFFEh.                                                 *
;*****************************************************************************
             LDPK   #232                            ; DP => EV Registers
;*************************************************************************
; Configure T1CON but do not start GP Timer 1                            *
;*************************************************************************
             SPLK   #1000000101000010b, T1CON        ; Set GP Timer 3 control
;                    ||||||||||||||||
;                    FEDCBA9876543210
;
* bit 0             0:     Use own PR
* bit 1             1:     GP Timer compare enabled
* bits 2-3          00:    Load GP Timer comp register on under flow
* bits 4-5          00:    Select internal CLK
* bit 6             1:     Timer (counting operation) enabled
* bit 7             0:     Use own Timer ENABLE
* bits 8-10         001:   Prescaler = /2
* bits 11-13        000:   Stop and hold mode
* bit 14            0:     SOFT = 0
* bit 15            1:     FREE = 1
;*************************************************************************
; Configure GPTCON
;*************************************************************************
             SPLK   #0000000001101010b, GPTCON       ; Set GP Timer control
;                    ||||||||||||||||
;                    FEDCBA9876543210
;
* bits 0-1          10:    GP Timer 1 comp output active high
* bits 2-3          10:    GP Timer 2 comp output active high
* bits 4-5          10:    GP Timer 3 comp output active high
* bit 6             1:     Enable GP Timer Compare outputs
* bits 7-8          00:    No GP Timer 1 event starts ADC
* bits 9-10         00:    No GP Timer 2 event starts ADC
* bits 11-12        00:    No GP Timer 3 event starts ADC
;*************************************************************************
; Configure T1PER T1CMP and T1CNT                                        *
;*************************************************************************
             SPLK   #05, T1PER                      ; Set GP Timer period
             SPLK   #03, T1CMP                      ; Set GP Timer compare
             SPLK   #0FFFEh, T1CNT                  ; Set GP Timer counter
```

Example 2–2. Initialization Routine for Continuous-Up Counting Mode (Continued)

```
;***************************************************************************
; Start GP Timer                                                            *
;***************************************************************************
             SPLK   #1001000101000010b, T1CON        ; Set GP Timer 3 control
;                    ||||||||||||||||
;                    FEDCBA9876543210
;
* bit 0             0:     Use own PR
* bit 1             1:     GP Timer compare enabled
* bits 2-3          00:    Load GP Timer comp register on under flow
* bits 4-5          00:    Select internal CLK
* bit 6             1:     Timer (counting operation) enabled
* bit 7             0:     Use own Timer ENABLE
* bits 8-10         001:   Prescaler = /2
* bits 11-13        010:   Continuous up count
* bit 14            0:     SOFT = 0
* bit 15            1:     FREE = 1
```

Directional-Up/Down Counting Mode

The GP timer in directional-up/down counting mode will count up or down according to the scaled clock and TMRDIR inputs. The GP timer will count up until its value reaches that of the period register or FFFFh when the TMRDIR pin is held high. When the timer value equals that of its period register or FFFFh and TMRDIR is held high, the timer will hold at that value. The GP timer will count down until its value becomes 0 when TMRDIR is held low. When the value of the timer is 0 and TMRDIR is held low, the timer will hold at 0.

The initial value of the timer can be any value between 0000h to FFFFh. When the initial value of the timer counter is greater than that of the period register, the timer will count up to FFFFh and hold at FFFFh if TMRDIR is held high, or it will count down to the value of the period register if TMRDIR is held low and continue as if the initial value of the timer were equal to that of the period register. If the initial value of the timer is equal to that of the period register, the timer will hold at that value if TMRDIR is held high, and count down from there if TMRDIR is held low.

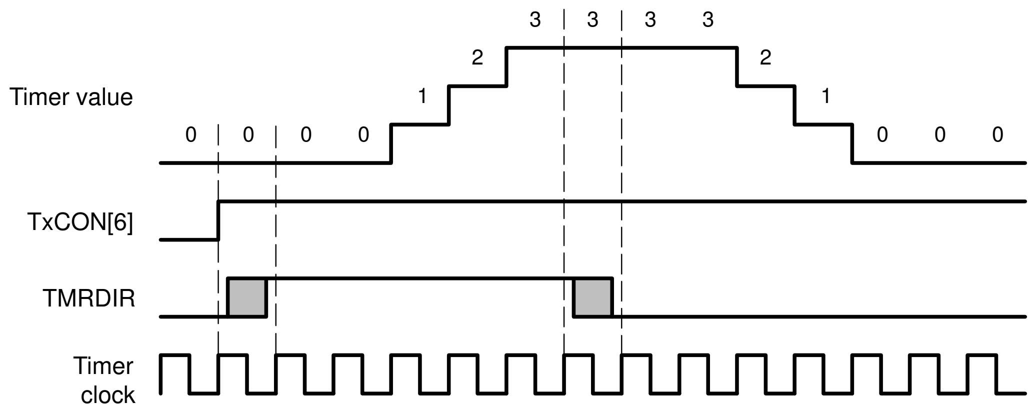
The period, underflow and overflow interrupt flags, interrupts, and associated actions are generated on respective events in the same manner as they are generated in single-up counting mode.

The latency from a change of TMRDIR to a change of counting direction is two CPU clock cycles after the end of the current count (that is, after the end of the current prescale counter period).

The direction of counting is indicated for the timer in this mode by the corresponding direction indication bit in GPTCON: 1 means counting up; 0 means counting down. Either external or Internal CPU clock can be used as input clock for the timer in this mode.

Figure 2–5 shows the directional-up/down counting mode of the GP timer.

Figure 2–5. GP Timer Directional-Up/Down Counting Mode With Prescale Factor 1 and TxPR = 3



The directional-up/down counting mode of the GP timer can be used with the Quadrature Encoder Pulse (QEP) circuits in the EV module. QEP circuits provide both the counting clock and direction for the timer in this case. This mode of operation can also be used to time the occurrence of external events in motion/motor control and power electronics applications. However, no transition will happen on the compare outputs associated with the compare modules (including GP timer compare, full compare, and simple compare) that use the GP timer in this mode as their time base.

Example 2–3 shows an initialization routine for the directional up/down counting mode for GP timer 1.

Example 2–3. Initialization Routine for Directional-Up/Down Counting Mode Using External Clock

```
;*******************************************************************************
; The following section of code initializes GP Timer1 to run in directional   *
; up-down counting mode. GP Timer compare function is also enabled, however   *
; that will have no effect on compare output pin.                              *
;*******************************************************************************
             LDPK   #232                          ; DP => EV Registers
;***************************************************************************
; Configure GPTCON
;***************************************************************************
             SPLK   #0000000001101010b, GPTCON       ; Set GP Timer control
;                    ||||||||||||||||
;                    FEDCBA9876543210
;
* bits 0-1          10:    GP Timer 1 comp output active high
* bits 2-3          10:    GP Timer 2 comp output active high
* bits 4-5          10:    GP Timer 3 comp output active high
* bit 6             1:     Enable GP Timer Compare outputs
* bits 7-8          00:    No GP Timer 1 event starts ADC
* bits 9-10         00:    No GP Timer 2 event starts ADC
* bits 11-12        00:    No GP Timer 3 event starts ADC
;***************************************************************************
; Configure T1PER and T1CMP                                                *
;***************************************************************************
             SPLK   #05, T1PER                    ; Set GP Timer period
             SPLK   #03, T1CMP                    ; Set GP Timer compare
;***************************************************************************
; Start GP Timer                                                           *
;***************************************************************************
             SPLK   #1001100001010010b, T1CON        ; Set GP Timer 3 control
;                    ||||||||||||||||
;                    FEDCBA9876543210
;
* bit 0             0:     Use own PR
* bit 1             1:     GP Timer compare enabled
* bits 2-3          00:    Load GP Timer comp register on under flow
* bits 4-5          10:    Select internal CLK
* bit 6             1:     Timer (counting operation) enabled
* bit 7             0:     Use own Timer ENABLE
* bits 8-10         000:   Prescaler = /1
* bits 11-13        011:   Directional up-down count
* bit 14            0:     SOFT = 0
* bit 15            1:     FREE = 1
```

Single-Up/Down Counting Mode

The GP timer in this mode will count up according to the scaled clock input to the value in its period register. It then changes its counting direction and counts downward until it reaches 0. When the timer reaches 0, it resets TxCON[6] and its prescale counter, stops its counting and holds at its current state.

The period of the GP timer in this mode is 2*(TxPR) cycles of the scaled clock input if the initial value of the timer is 0.

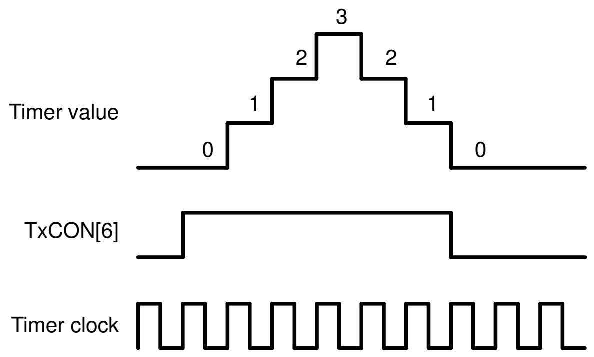
The initial value of the timer counter can be any value between 0h and FFFFh. If the initial timer value is greater than that of the period register, the timer will count up to FFFFh, reset to 0, and continue as if the initial value of the timer were 0. If the initial value of the timer is the same as that of the period register, the timer will count down to 0 and end the period there. If the initial value of the timer is between 0 and the contents of the period register, the timer will count up to the period value and continue to finish the period as if the initial counter value were the same as that of the period register.

The period, underflow and overflow interrupt flags, interrupts and associated actions are generated on respective events in the same manner as they are generated in single-up counting mode. Note, however, that period event happens when a match between the timer counter and period register is made, which is at the middle of a counting period.

Once the operation of the GP timer in this mode has ended, the operation can only be started again by software writing a 1 to TxCON[6]. The direction indication bit in GPTCON is 1 when the counting direction is up and 0 when the counting direction is down. Either external clock or internal CPU clock can be selected as the clock input to the timer. TMRDIR input is ignored by the GP timer in this mode.

Figure 2–6 shows the single-up/down mode of the GP timer. Example 2–4 shows an initialization routine for single-up/down counting mode for GP timer 1.

Figure 2–6. GP Timer Single-Up/Down Counting Mode (TxPR = 3)



Example 2–4. Initialization Routine for Single-Up/Down Counting Mode

```
;*********************************************************************
; The following section of code initializes GP Timer1 to run in single Up   *
; count mode. GP Timer compare function is also enabled.                  *
;***********************************************************************
             LDPK   #232                              ; DP => EV Registers
;***********************************************************************
; Configure GPTCON
;***********************************************************************
             SPLK   #0000000001000001b, GPTCON       ; Set GP Timer control
;                    ||||||||||||||||
;                    FEDCBA9876543210
;
* bits 0-1          01:    GP Timer 1 comp output active low
* bits 2-3          00:    GP Timer 2 comp output forced low
* bits 4-5          00:    GP Timer 3 comp output forced low
* bit 6             1:     Enable GP Timer Compare outputs
* bits 7-8          00:    No GP Timer 1 event starts ADC
* bits 9-9          00:    No GP Timer 2 event starts ADC
* bits 10-11        00:    No GP Timer 3 event starts ADC
;**********************************************************************
; Configure T1PER and T1CMP                                            *
;***********************************************************************
             SPLK   #05, T1PER                        ; Set GP Timer period
             SPLK   #03, T1CMP                        ; Set GP Timer compare
;***********************************************************************
; Configure T1CON and start GP Timer 1                                *
;***********************************************************************
             SPLK   #1000100101000010b, T1CON        ; Set GP Timer 3 control
;                    ||||||||||||||||
;                    FEDCBA9876543210
;
* bit 0             0:     Use own PR
* bit 1             1:     GP Timer compare enabled
* bits 2-3          00:    Load GP Timer comp register on underflow
* bits 4-5          00:    Select internal CLK
* bit 6             1:     Timer (counting operation) enabled
* bit 7             0:     Use own Timer ENABLE
* bits 8-10         001:   Prescaler = /2
* bits 11-13        001:   Single up count
* bit 14            0:     SOFT = 0
* bit 15            1:     FREE = 1
```

Continuous-Up/Down Counting Mode

This mode of operation is the same as the single-up/down counting mode repeated each time the timer is reset to 0. Once this mode of operation is started, no user software or hardware intervention is required to repeat the counting period.

The period of the timer in this mode is 2*(TxPR) cycles of the scaled clock input except for the first period. The duration of the first counting period is the same if the timer counter is 0 when counting starts.

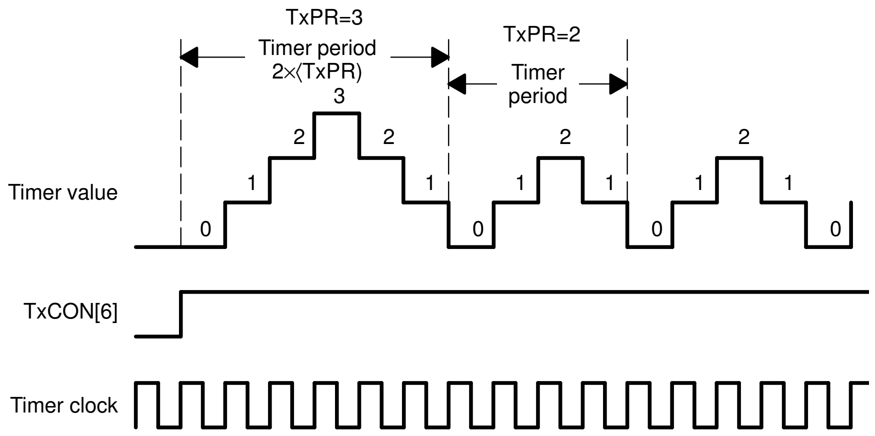
The initial value of the GP timer counter can be any value between 0h and FFFFh. When the initial value is greater than that of the period register, the timer will count up to FFFFh, reset to 0, and continue the operation as if the initial value were 0. When the initial value in the timer counter is the same as that of the period register, the timer will count down to 0 and continue again as if the initial value were 0. If the initial value of the timer is between 0 and the contents of the period register, the timer will count up to the period value and continue to finish the period as if the initial counter value were the same as that of the period register.

The period, underflow and overflow interrupt flags, interrupts and associated actions are generated on respective events in the same manner as they are generated in single-up counting mode.

The counting direction indication bit for this timer in GPTCON is 1 when the timer counts upward and 0 when the timer counts downward. Either external or internal CPU clock can be selected as input clock. TMRDIR input is ignored by the timer in this mode.

Figure 2–7 shows the continuous-up/down counting mode of the GP timer.

Figure 2–7. GP Timer Continuous-Up/Down Counting Mode (TxPR = 3 or 2)



Continuous-up/down counting mode is particularly useful in generating centered or symmetric PWM waveforms found in a broad range of motor/motion control and power electronics applications.

Example 2–5 shows an initialization routine for continuous-up/down counting mode.

Example 2–5. Initialization Routine for Continuous-Up/Down Counting Mode

```
;*******************************************************************
; The following section of code initializes GP Timer1 to run in continuous  Up/down *
; count mode. GP Timer compare function is also enabled.                  *
;*********************************************************************
             LDPK   #232                            ; DP => EV Registers
;*********************************************************************
; Configure GPTCON
;*********************************************************************
             SPLK   #0000000000111111b, GPTCON       ; Set GP Timer control
;                    ||||||||||||||||
;                    FEDCBA9876543210
;
* bits 0-1          01:    GP Timer 1 comp output forced high
* bits 2-3          00:    GP Timer 2 comp output forced high
* bits 4-5          00:    GP Timer 3 comp output forced high
* bit 6             1:     Enable GP Timer Compare outputs
* bits 7-8          00:    No GP Timer 1 event starts ADC
* bits 9-9          00:    No GP Timer 2 event starts ADC
* bits 10-11        00:    No GP Timer 3 event starts ADC
;********************************************************************
; Configure T1PER and T1CMP                                         *
;*********************************************************************
             SPLK   #05, T1PER                      ; Set GP Timer period
             SPLK   #03, T1CMP                      ; Set GP Timer compare
;*********************************************************************
; Configure T1CON and start GP Timer 1                              *
;*********************************************************************
             SPLK   #1010100001000000b, T1CON        ; Set GP Timer 3 control
;                    ||||||||||||||||
;                    FEDCBA9876543210
;
* bit 0             0:     Use own PR
* bit 1             0:     GP Timer compare disabled
* bits 2-3          00:    Load GP Timer comp register on underflow
* bits 4-5          00:    Select internal CLK
* bit 6             1:     Timer (counting operation) enabled
* bit 7             0:     Use own Timer ENABLE
* bits 8-10         000:   Prescaler = /1
* bits 11-13        010:   Continuous up/down count
* bit 14            0:     SOFT = 0
* bit 15            1:     FREE = 1
```

2.3.2 GP Timer Compare Operation

Each GP timer has an associated compare register TxCMPR and a compare/PWM output pin TxPWM/TxCMP. The value of a GP timer counter is constantly compared to that of its associated compare register. A compare match occurs when the value of the timer counter is the same as that of the compare register. Compare operation is enabled by setting TxCON[1] to 1. If it is enabled, the following happens on a compare match:

- ❏ The compare interrupt flag of the timer is set two CPU clock cycles after the match.
- ❏ If the GP timer is not in directional-up/down counting mode, a transition occurs on the associated compare/PWM output according to the bit configuration in GPTCON, one CPU clock cycle after the match.
- ❏ If the compare interrupt flag has been selected by the appropriate GPTCON bits to start ADC, an ADC start signal is generated at the same time the compare interrupt flag is set.

An interrupt request to the core is generated by the compare interrupt flag if it is unmasked and no other unmasked interrupts in the same group of higher priority are pending.

Compare/PWM Transition

The transition on compare/PWM output is controlled by an asymmetric and symmetric waveform generator and the associated output logic, and depends on the following:

- ❏ Bit definition in GPTCON.
- ❏ Counting mode the timer is in.
- ❏ Counting direction when the counting mode is single- or continuous-up/down mode.

Asymmetric/Symmetric Waveform Generator

The asymmetric/symmetric waveform generator generates an asymmetric or symmetric compare/PWM waveform based on the counting mode the GP timer is in.

Asymmetric Waveform Generation

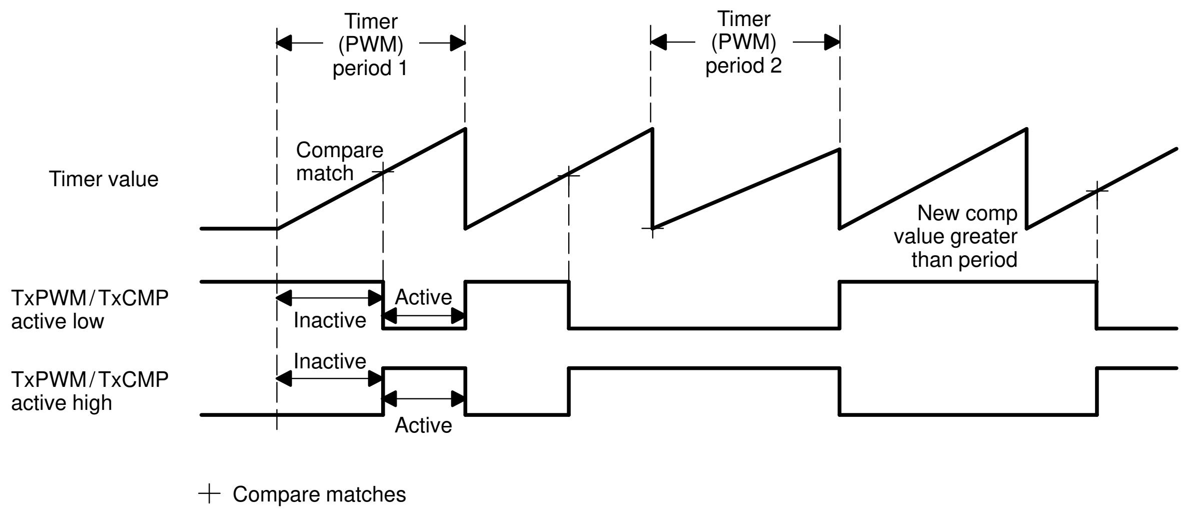
Asymmetric waveform (Figure 2–8) is generated when GP timer is in single- or continuous-up counting modes. When the GP timer is in these two modes, the output of the waveform generator changes according to the following sequence:

- 0 before the counting operation starts
- remains unchanged until the compare match happens
- toggles on compare match
- remains unchanged until the end of the period
- resets to 0 at the end of a period on period match, if the new compare value for the following period is not 0

The output will be 1 for the whole period, if the compare value is 0 at the beginning of a period. The output will not reset to 0, if the new compare value for the following period is 0. This is important because it allows the generation of PWM pulses of 0% to 100% duty cycle without glitches. The output is 0 for the whole period, if the compare value is greater than the value in the period register. The output will be 1 for one cycle of the scaled clock input, if the compare value is the same as that of the period register.

One characteristic of asymmetric compare/PWM waveform is that a change in the value of the compare register only affects one side of the compare/PWM pulse.

Figure 2–8. GP Timer Compare/PWM Output in Up Counting Mode



Symmetric Waveform Generation

Symmetric waveform (Figure 2–9) is generated when the GP timer is in single- or continuous-up/down counting modes. When the GP timer is in these two modes, the state of the output of the waveform generator is determined by the following:

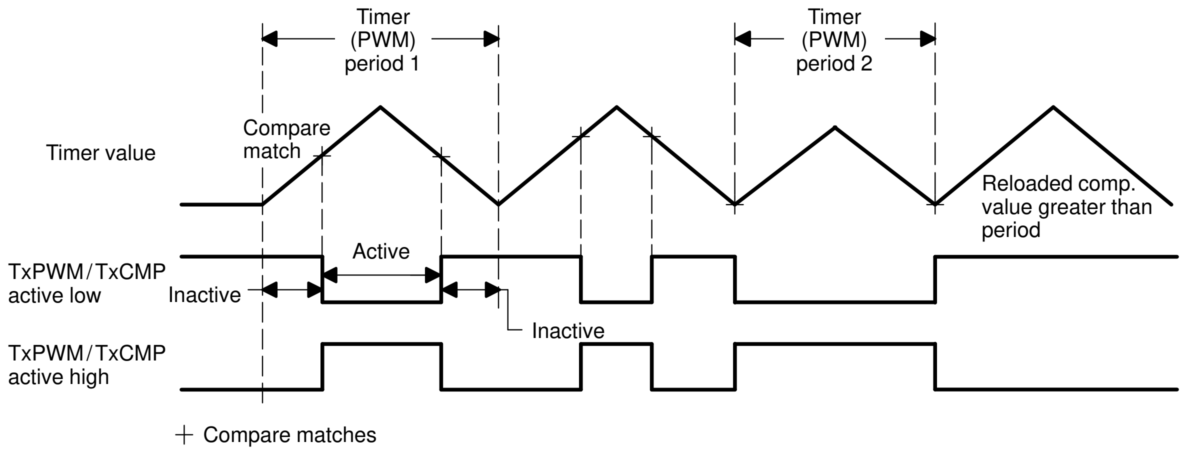
- ❑ 0 before the counting operation starts
- ❑ Remains unchanged until first compare match
- ❑ Toggles on the first compare match
- ❑ Remains unchanged until the second compare match
- ❑ Toggles on the second compare match
- ❑ Remains unchanged until the end of the period
- ❑ Resets to 0 at the end of the period, if there is not a second compare match and the new compare value for the following period is not 0

The output is set to 1 at the beginning of a period and will remain 1 until the second compare match, if the compare value is 0 at the beginning of a period. After the first transition, the output will remain 1 until the end of the period if the compare value is 0 for the second half of the period. When this happens, the output will not reset to 0 if the new compare value for the following period is still 0. This is done again to assure the generation of PWM pulses of 0% to 100% duty cycle without any glitches. The first transition will not happen if the compare value is greater than or equal to that of the period register for the first half of the period. However, the output will still toggle when a compare match happens in the second half of the period. This error in output transition, often as a result of calculation error in the application routine, will be corrected at the end of the period because the output will reset to 0 unless the new compare value for the following period is 0; in which case, the output will remain to be 1, which again puts the output of the waveform generator in the correct state.

Note:

The output logic determines what the polarity is for all the output pins.

Figure 2–9. GP Timer Compare/PWM Output in Up/Down Counting Modes



Output Logic

The output logic further conditions the output of the waveform generator to form the ultimate compare/PWM output to control different kinds of power devices. The compare/PWM output can be specified to be active high, active low, forced low, and forced high by proper configuration of GPTCON bits.

The polarity of the compare/PWM output is the same as that of the output of the associated asymmetric/symmetric waveform generator when the compare/PWM output is specified to be active high.

The polarity of the compare/PWM output is the opposite of that of the output of the associated asymmetric/symmetric waveform generator when the compare/PWM output is specified to be active low.

The compare/PWM output is set to 1 (or 0) immediately after the corresponding bits in GPTCON are set and the bit pattern specifies that the state of compare/PWM output is forced high (or low).

In summary, during a normal counting mode, transition on GP timer compare/ PWM output happens according to Table 2–6 for single-up and continuous-up counting modes and according to Table 2–7 for single-up/down and continuous-up/down counting modes, assuming compare is enabled.

Setting active means setting high for active high and setting low for active low. Setting inactive means the opposite.

The asymmetric/symmetric waveform generation based on the timer counting mode and the output logic are also applicable to both full and simple compare units.

Table 2–6. GP Timer Compare Output in Single-Up and Continuous-Up Counting Modes

Time in a period	**State of Compare Output**
Before compare match	Inactive
On compare match	Set active
On period match	Set inactive

Table 2–7. GP Timer Compare Output in Single-Up/Down and Continuous-Up/Down Counting Modes

Time in a period	**State of Compare Output**
Before 1st compare match	Inactive
On 1st compare match	Set active
On 2nd compare match	Set inactive
After 2nd compare match	Inactive

All GP timer compare/PWM outputs are put in high impedance when any of the following events occurs:

- ❑ GPTCON[6] is set to 0 by software
- ❑ PDPINT is pulled low and is unmasked
- ❑ Any reset event occurs

Compare Output in Directional-Up/Down Mode

When a GP timer is in directional-up/down counting mode, no transition occurs on its compare output. Similarly, no transition happens on the compare outputs associated with full compare units when GP Timer 1 is in directional-up/down counting mode; no transition happens on the compare outputs associated with simple compare units when the GP timer selected as the time base for simple compare units is in directional-up/down counting mode. The setting of compare interrupt flags and the generation of compare interrupt requests, however, do not depend on what counting mode the GP timer is in.

Active/Inactive Time Calculation

For up counting modes, the value in compare register represents the elapsed time between the beginning of a period and the occurrence of the first compare match, that is, the length of the inactive phase. This elapsed time is equal to the period of the scaled input clock multiplied by the value of TxCMPR. Therefore, the length of the active phase, the output pulse width, is given by (TxPR) – (TxCMPR) + 1 cycles of the scaled input clock.

For up/down counting modes, the compare register can have a different value on the way counting down from the value on the way counting up. The length of the active phase, that is, the output pulse width, for up/down counting modes, is thus given by (TxPR) – $(\text{TxCMPR})_{up}$ + (TxPR) – $(\text{TxCMPR})_{dn}$ cycles of the scaled input clock, where $(\text{TxCMPR})_{up}$ is the compare value on the way up and $(\text{TxCMPR})_{dn}$ is the compare value on the way down.

When the value in TxCMPR is 0, the GP timer compare output will be active for the whole period if the timer is in up counting mode. For up/down counting modes, the compare output will be active at the beginning of the period if $(\text{TxCMPR})_{up}$ is 0. The output will remain active until the end of the period if $(\text{TxCMPR})_{dn}$ is also 0.

The length of the active phase (the output pulse width) will be 0 when the value of TxCMPR is greater than that of TxPR for up counting modes. For up/down counting mode, the first transition will be lost when $(\text{TxCMPR})_{up}$ is greater than or equal to (TxPR). Similarly, the second transition will be lost when $(\text{TxCMPR})_{dn}$ is greater than or equal to (TxPR). The GP timer compare output will be inactive for the entire period if both $(\text{TxCMPR})_{up}$ and $\text{TxCMPR})_{dn}$ are greater than or equal to (TxPR) for up/down counting modes.

Figure 2–8 (page 2-31) shows the compare operation of a GP timer in up counting modes. Figure 2–9 (page 2-33) shows the compare operation of a GP timer in up/down counting mode.

2.3.3 GP Timer Control Registers (TxCON and GPTCON)

Addresses of GP timer registers are given in Table 2–2 on page 2-8. The bit definition of the GP timer control register TxCON is shown in Figure 2–10 and of GP timer control register GPTCON is shown in Figure 2–11 (page 2-38).

GP Timer Control Register (TxCON; x = 1, 2, and 3)

Figure 2–10. GP Timer Control Register (TxCON; x = 1, 2, and 3)

15	14	13	12	11	10	9	8
Free	Soft	TMODE2	TMODE1	TMODE0	TPS2	TPS1	TPS0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
TSWT1	TENABLE	TCLKS1	TCLKS0	TCLD1	TCLD0	TECMPR	SELT1PR
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

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

Bits 15–14 **Free, Soft**. Emulation control bits.

00 = Stop immediately on emulation suspend.
01 = Stop after current timer period is completed on emulation suspend.
10 = Operation is not affected by emulation suspend.
11 = Operation is not affected by emulation suspend.

Bits 13–11 **TMODE2–TMODE0**. Count Mode Selection.

000 = Stop/Hold
001 = Single-Up Count Mode
010 = Continuous-Up Count Mode
011 = Directional-Up/Down Count Mode
100 = Single-Up/Down Count Mode
101 = Continuous-Up/Down Count Mode
110 = Reserved. Result is unpredictable.
111 = Reserved. Result is unpredictable.

Bits 10–8 **TPS2–TPS0**. Input Clock Prescaler.

000 = x/1
001 = x/2
010 = x/4
011 = x/8
100 = x/16
101 = x/32
110 = x/64
111 = x/128
x = CPU clock frequency

Bit 7	**TSWT1**. (GP Timer Start with GP Timer 1). Start timer with GP Timer 1 timer enable bit. This bit is reserved in T1CON. 0 = Use own TENABLE bit. 1 = Use TENABLE bit of T1CON to enable and disable operation ignoring own TENABLE bit.
Bit 6	**TENABLE**. Timer enable. In Single-Up and Single-Up/Down count modes this bit is cleared to 0 by the timer after it completes a period of operation. 0 = Disable timer operation, that is the timer is put in hold and the prescaler counter is reset. 1 = Enable timer operations.
Bits 5–4	**TCLKS1, TCLKS0**. Clock Source Select. 00 = Internal 01 = External 10 = Rollover from GP Timer 2. (Only applicable to T3CON when GP Timers 2 and 3 are cascaded into a 32-bit timer; reserved in T1CON and T2CON; ILLEGAL if SELT1PR=1.) 11 = Quadrature Encoder Pulse circuit. (Only applicable to T2CON and T3CON; reserved in T1CON; ILLEGAL if SELT1PR is 1.) ILLEGAL means result is unpredictable.
Bits 3–2	**TCLD1, TCLD0**. Timer Compare (Active) Register Reload Condition. 00 = When counter is 0. 01 = When counter value is 0 or equals period register value. 10 = Immediately 11 = Reserved
Bit 1	**TECMPR**. Timer compare enable. 0 = Disable timer compare operation. 1 = Enable timer compare operation.
Bit 0	**SELT1PR**. Period register select. This bit is reserved in T1CON. 0 = Use own period register. 1 = Use T1PR as period register ignoring own period register.

Note: Synchronization of the GP Timers

Two consecutive writes to T1CON are required to ensure the synchronization of the GP timers when T1CON[6] is used to enable GP Timer 2 or 3:

1) Configure all other bits with T1CON[6] set to 0.

2) Enable GP Timer 1 and, thus, GP Timer 2 or GP Timers 2 and 3, by setting T1CON[6] to 1.

GP Timer Control Register (GPTCON)

Figure 2–11. GP Timer Control Register (GPTCON) — Address 7400h

15	14	13	12–11	10–9	8–7
T3STAT	T2STAT	T1STAT	T3TOADC	T2TOADC	T1TOADC
R–1	R–1	R–1	RW–0	RW–0	RW–0

6	5–4	3–2	1–0
TCOMPOE	T3PIN	T2PIN	T1PIN
RW–0	RW–0	RW–0	RW–0

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

Bit 15 **T3STAT**. GP Timer 3 Status. Read only.

0 = Counting downward
1 = Counting upward

Bit 14 **T2STAT**. GP Timer 2 Status. Read only.

0 = Counting downward
1 = Counting upward

Bit 13 **T1STAT**. GP Timer 1 Status. Read only.

0 = Counting downward
1 = Counting upward

Bits 12–11 **T3TOADC**. ADC start by event of GP Timer 3.

00 = No event starts ADC.
01 = Setting of underflow interrupt flag. Starts ADC.
10 = Setting of period interrupt flag. Starts ADC.
11 = Setting of compare interrupt flag. Starts ADC.

Bits 10–9 **T2TOADC**. ADC start by event of GP Timer 2.

00 = No event starts ADC.
01 = Setting of underflow interrupt flag. Starts ADC.
10 = Setting of period interrupt flag. Starts ADC.
11 = Setting of compare interrupt flag. Starts ADC.

Bits 8–7 **T1TOADC**. ADC start by event of GP Timer 1.

00 = No event starts ADC.
01 = Setting of underflow interrupt flag. Starts ADC.
10 = Setting of period interrupt flag. Starts ADC.
11 = Setting of compare interrupt flag. Starts ADC.

Bit 6	**TCOMPOE**. Compare output enable. Active PDPINT writes zero to this bit.
	0 = Disable all three GP timer compare outputs (all three compare outputs are put in high-impedance state).
	1 = Enable all three GP timer compare outputs.
Bits 5–4	**T3PIN**. Polarity of GP Timer 3 compare output.
	00 = Forced low
	01 = Active low
	10 = Active high
	11 = Forced high
Bits 3–2	**T2PIN**. Polarity of GP Timer 2 compare output.
	00 = Forced low
	01 = Active low
	10 = Active high
	11 = Forced high
Bits 1–0	**T1PIN**. Polarity of GP Timer 1 compare output.
	00 = Forced low
	01 = Active low
	10 = Active high
	11 = Forced high

2.3.4 Generation of Compare and PWM Outputs Using GP Timers

Each GP Timer can independently be used to provide a compare or PWM output channel. Thus up to three compare or PWM outputs may be generated by the GP timers.

Compare Operation

To generate a compare output, an appropriate operation mode of GP timer must be selected. The following procedure should then be followed:

- ❑ Set up TxCMPR according to when the compare event is expected to happen.
- ❑ Set up GPTCON such that a desired transition on compare output will take place on compare match.
- ❑ Load TxPR with desired period value, if needed.
- ❑ Load TxCNT with desired initial counter value, if necessary.
- ❑ Set up TxCON to specify the counting mode and clock source and start the operation.

PWM Operation

To generate a PWM output with a GP timer, a continuous- up or up/down counting mode can be selected. Edge-triggered or asymmetric PWM wave forms are generated when a continuous-up count mode is selected. Centered or symmetric PWM wave forms are generated a when continuous-up/down mode is selected. To set up the GP timer for this operation, do the following:

❑ Set up TxPR according to the desired PWM (carrier) period.

❑ Set up TxCON to specify the counting mode and clock source and start the operation.

❑ Load TxCMPR with values corresponding to the on-line calculated widths (duty cycles) of PWM pulses.

The period value is obtained by dividing the desired PWM period by the period of the GP timer input clock, and subtracting one from the resulting number when the continuous-up counting mode is selected to generate asymmetric PWM wave forms. When the continuous-up/down counting mode is selected to generate symmetric PWM wave forms, this value is obtained by dividing the desired PWM period by two times the period of the GP timer input clock.

The GP Timer can be initialized the same way as in the previous example. During run time, the GP Timer compare register is constantly updated with newly determined compare values corresponding to the newly determined duty cycles.

Figure 2–8 (page 2-31) and Figure 2–9 (page 2-33) are examples of asymmetric and symmetric PWM waveforms that can be generated by GP timer.

2.3.5 GP Timer Reset

When any RESET event occurs, the following happens:

❑ All GP timer register bits, except for the counting direction indication bits in GPTCON, are reset to 0; thus, the operation of all GP timers is disabled. The counting direction indication bits are all set to 1.

❑ All timer interrupt flags are reset to 0.

❑ All timer interrupt mask bits are reset to 0; thus, all GP timer interrupts are masked.

❑ All GP timer compare outputs are put in high-impedance state.

2.4 Compare Units

There are three full compare units (Full Compare Units 1, 2, and 3) and three simple compare units (Simple Compare Units 1, 2, and 3) in the EV module. Each full compare unit has two associated compare/PWM outputs. Each simple compare unit has one associated compare/PWM output. The time base for full compare units is provided by GP Timer 1. The time base for simple compare units can be GP Timer 1 or 2.

2.4.1 Simple Compare Units

The three simple compare units include:

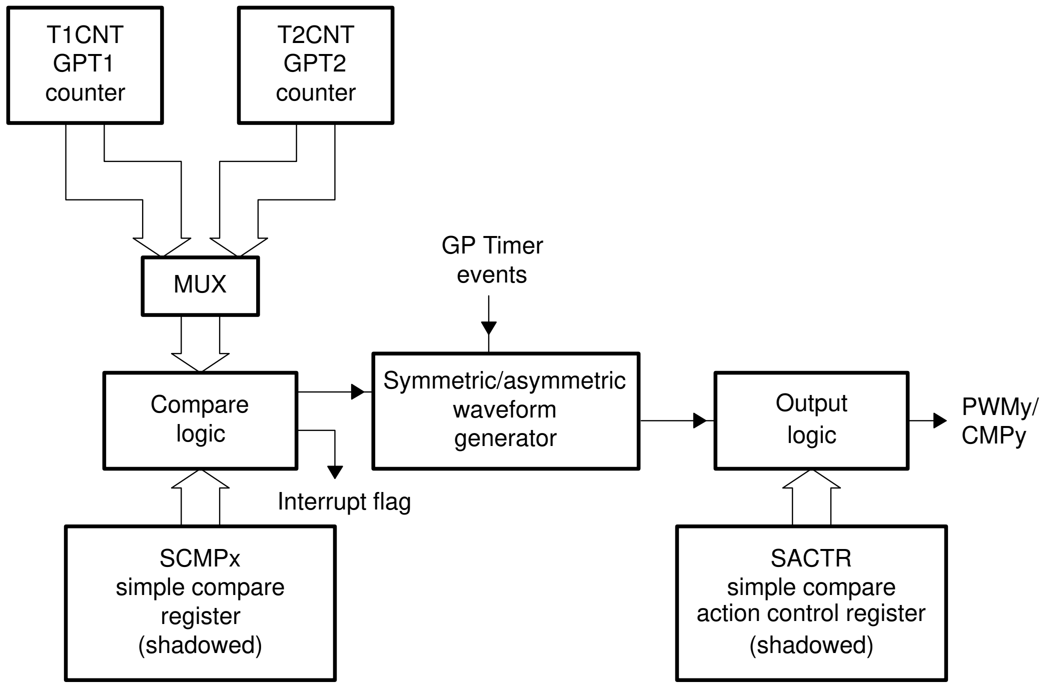
- ❑ Three 16-bit compare registers (SCMPRx, x = 1, 2, 3) all with an associated shadow register, (R/W)
- ❑ One compare control register (COMCON), shared with full compare units, (R/W)
- ❑ One 16-bit action control register (SACTR), with associated shadow register, (R/W)
- ❑ Three asymmetric/symmetric waveform generators
- ❑ Three compare/PWM (3-state) outputs, PWMy/CMPy, y = 7, 8, 9, one for each simple compare unit, with programmable polarity
- ❑ Compare and interrupt logic

The operation of the three simple compare units are the same as the GP timer compare operation except that:

- ❑ The time base for the simple compare units can be GP Timer 1 or 2.
- ❑ The enabling and disabling of simple compare operation, the enabling and disabling of simple compare outputs, the condition when (active) simple compare register is updated, and the selection of time base for simple compare operation are controlled by appropriate bits in COMCON.
- ❑ The behavior of compare outputs of simple compare units is individually defined by corresponding bits in simple compare action control register SACTR.

Figure 2–12 shows a block diagram of simple compare units.

Figure 2–12. Simple Compare Unit Block Diagram (x = 1, 2, or 3; y = 7, 8, or 9)



The timing of simple compare outputs is the same as that of the GP timer compare outputs. There is an interrupt flag for each simple compare unit. The setting of simple compare interrupt flags and the generation of simple compare interrupt requests are also the same as that of the GP timer compare operation.

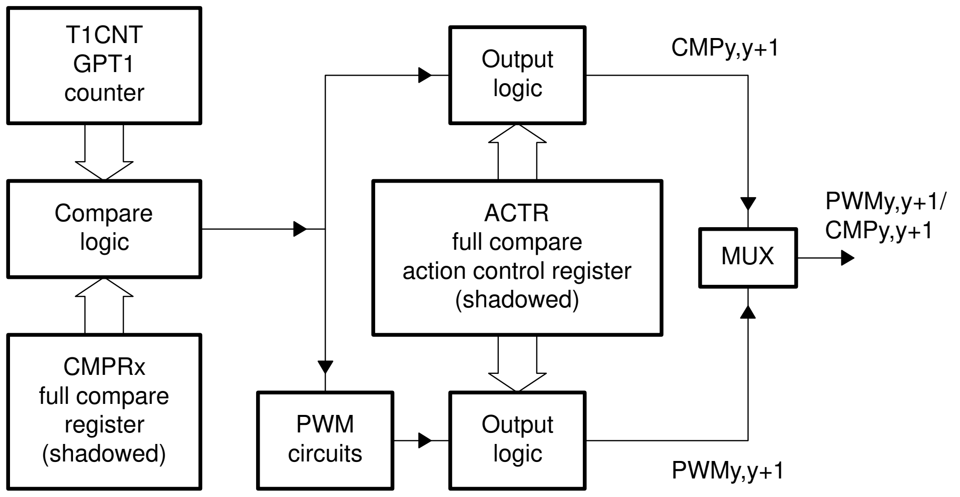
2.4.2 Full Compare Units

The three full compare units include:

- ❑ Three 16-bit compare registers (CMPRx, x = 1, 2, 3), all with an associated shadow register, (R/W)
- ❑ One 16-bit compare control register (COMCON), (R/W)
- ❑ One 16-bit action control register (ACTR), with associated shadow register, (R/W)
- ❑ Six compare/PWM (3-state) output pins, PWMy/CMPy, y = 1, 2, 3, 4, 5, 6
- ❑ Control and interrupt logic

The functional block diagram of a full compare unit is shown in Figure 2–13.

Figure 2–13. Full Compare Unit Block Diagram (x = 1, 2, 3; y = 1, 3, 5)



The time base for full compare units and the associated PWM circuits is provided by GP Timer 1. GP Timer 1 can be in any of its six counting modes when compare operation is enabled. However, no transition happens on compare outputs when GP Timer 1 is in directional-up/down counting mode.

Full Compare Inputs/Outputs

The inputs to a full compare unit include:

- ❑ Control signals from control registers
- ❑ GP Timer 1 (T1CNT) and its underflow and period match signals
- ❑ RESET

The output of a full compare unit is a compare match signal. If the compare operation is enabled, this match signal will set the interrupt flag and cause transitions on the two output pins associated with the full compare unit.

Full Compare Operation Modes

The operation mode of full compare units is determined by bits in COMCON. These bits determine:

- ❑ Whether the compare operation is enabled.
- ❑ Whether the full compare outputs are enabled.
- ❑ Condition on which full compare registers are updated with values in their shadow registers.

- ❑ Condition on which the action control register is updated with value in its shadow register.
- ❑ Whether space vector PWM mode is enabled.
- ❑ Whether each full compare unit is in compare mode or PWM mode.

The three full compare units can all operate in either of two operating modes: compare mode or PWM mode. Bits in COMCON determine which operating mode each full compare unit is in.

Compare Mode

When compare mode is selected and compare is enabled, the value of the GP Timer 1 counter is continuously compared with that of the compare register. When a match is made, a transition will appear on the two outputs of the compare unit according to the bits in action control register (ACTR). Bits in ACTR can individually specify each output to Hold, Reset (go low), Set (go high), or Toggle on a compare match. The compare interrupt flag associated with a full compare unit is set when a compare match is made between GP Timer 1 and the compare register of this compare unit, and compare is enabled. An interrupt request to the core is generated by the flag if no other unmasked interrupts in the same group of higher priority are pending. The timing of output transitions, setting of interrupt flags, and generation of interrupt requests are the same as that of GP timer compare operation. The outputs of the full compare units in compare mode are subject to modification by the output logic.

PWM Mode

Each full compare unit can be individually put in PWM mode. The operation of full compare units in this mode is similar to that of the GP Timer compare operation with the exception that the full compare units are controlled by different control registers and that the full compare/PWM outputs are subject to modification by dead band units and space vector PWM logic. The PWM mode of operation is further described in the following sections.

Register Setup for Full Compare Operation

The register setup sequence for full compare operation requires:

- ❑ Setting up T1PR.
- ❑ Setting up ACTR.
- ❑ Initializing CMPRx.
- ❑ Setting up COMCON.
- ❑ Setting up T1CON.

Note that in many cases, COMCON requires two writes to ensure the polarity of PWM outputs.

2.4.3 Compare Units Registers

The addresses of registers associated with full and simple compare units and associated PWM circuits are shown in Table 2–3 (page 2-9) and further discussed in the following subsections.

Compare Control Register (COMCON)

The operation of full and simple compare units are controlled by the 16-bit compare control register (COMCON). The bit definition of COMCON is summarized in Figure 2–14. COMCON is read- and write-able and data memory mapped.

Figure 2–14. Compare Control Register (COMCON) — Address 7411h

15	14	13	12	11	10	9	8
CENABLE	CLD1	CLD0	SVENABLE	ACTRLD1	ACTRLD0	FCOMPOE	SCOMPOE
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
SELTMR	SCLD1	SCLD0	SACTRLD1	SACTRLD0	SELCMP3	SELCMP2	SELCMP1
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

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

Bit 15 **CENABLE**. Compare enable.

0 = Disable compare operation. All shadowed registers (CMPRx, SCMPRx, ACTR, SACTR) become transparent.
1 = Enable compare operation.

Bits14–13 **CLD1, CLD0**. Full compare register CMPRx reload condition.

00 = When T1CNT = 0 (that is, on underflow).
01 = When T1CNT = 0 or T1CNT = T1PR (that is, on underflow or period match).
10 = Immediately
11 = Reserved. Result is unpredictable.

Bit 12 **SVENABLE**. Space vector PWM mode enable. In space vector PWM mode, all six full compare outputs are PWM outputs. SVENABLE = 1 overrides COMCON bits 0 ,1, and 2.

0 = Disable space vector PWM mode.
1 = Enable space vector PWM mode.

Bits11–10 **ACTRLD1, ACTRLD0**. Full compare action register ACTR reload condition.

00 = When T1CNT = 0 (that is, on underflow).
01 = When T1CNT = 0 or T1CNT = T1PR (that is, on underflow or period match).
10 = Immediately
11 = Reserved. Result is unpredictable.

Bit 9 **FCOMPOE**. Full compare output enable. Active PDPINT clears this bit to 0.

0 = Full compare output pins are in high-impedance state, that is they are disabled.
1 = Full compare output pins are not in high-impedance state, that is they are enabled.

Bit 8 **SCOMPOE**. Simple compare output enable. Active PDPINT clears this bit to 0.

0 = Simple compare output pins are in high-impedance state, that is they are disabled.
1 = Simple compare output pins are not in high-impedance state, that is they are enabled.

Bit 7 **SELTMR**. Simple compare time base select.

0 = GP Timer 1
1 = GP Timer 2

Bits 6–5 **SCLD1, SCLD0**. Simple compare register SCMPRx reload condition.

00 = When TyCNT = 0 (y = 1 or 2 according to SELTMR).
01 = When TyCNT = 0 or TyCNT = TyPR.
10 = Immediately
11 = Reserved

Bits 4–3 **SACTRLD1, SACTRLD0**. Simple compare action register SACTR reload condition.

00 = When TyCNT = 0 (y = 1 or 2 according to SELTMR).
01 = When TyCNT = 0 or TyCNT = TyPR.
10 = Immediately
11 = Reserved

Bit 2 **SELCMP3**. Mode select for PWM6/CMP6 and PWM5/CMP5 (for Full Compare Unit 3).

0 = Compare mode
1 = PWM mode

Bit 1 **SELCMP2**. Mode select for PWM4/CMP4 and PWM3/CMP3 (for Full Compare Unit 2).

0 = Compare mode
1 = PWM mode

Bit 0 **SELCMP1**. Mode select for PWM2/CMP2 and PWM1/CMP1 (for Full Compare Unit 1).

0 = Compare mode
1 = PWM mode

Note:

Two consecutive writes to COMCON are required to ensure the proper operation of the Full Compare Units in PWM mode:

1) Enable PWM mode without enabling compare operation.
2) Enable compare operation by setting COMCON[15] to 1 without changing any other bits.

See Example 2–6 for a code example.

Example 2–6. Full Compare Units in PWM Mode

```
;***********************************************************************
; Configure COMCON register                                            *
;***********************************************************************
       SPLK   #0100101101010111B, COMCON ; COMCON needs to be written twice for
       SPLK   #1100101101010111B, COMCON ; proper operation.
               ||||||||||||||||
               FEDCBA9876543210
; bit 15      : 1    : Enable PWM
; bit 14–13   :10    : Load compare immediate
; bit 12      : 0    : Disable SV
; bit 11–10   :10    : Load ACTR immediate
; bit 9       : 1    : PWM specified by ACTR
; bit 8       : 1    : SPWM specified by SACTR
; bit 7       : 0    : Simple compare are associated with T2
; bit 6–5     :10    : Load SCMPR immediate
; bit 4–3     :10    : Load SACTR immediate
; bit 2       : 1    : CMP6/5 enabled
; bit 1       : 1    : CMP4/3 enabled
; bit 0       : 1    : CMP2/1 enabled
```

Full Compare Action Control Register (ACTR)

Bits in the full compare action control register (ACTR) control the action that takes place on each of the 6 compare output pins (PWMx/CMPx, x = 1–6) on a compare event, in both compare and PWM operation modes, if the compare operation is enabled by COMCON[15]. ACTR is double buffered. The condition on which the ACTR is reloaded is specified by bits in COMCON. ACTR also contains the SVRDIR, D2, D1, and D0 bits needed for space vector PWM operation. The bit configuration of ACTR is described in Figure 2–15.

Figure 2–15. Full Compare Action Control Register (ACTR) — Address 7413h

15	14	13	12	11	10	9	8
SVRDIR	D2	D1	D0	CMP6ACT1	CMP6ACT0	CMP5ACT1	CMP5ACT0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
CMP4ACT1	CMP4ACT0	CMP3ACT1	CMP3ACT0	CMP2ACT1	CMP2ACT0	CMP1ACT1	CMP1ACT0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

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

Bit 15 **SVRDIR**. Space vector PWM rotation direction. Used only in space vector PWM output generation.

0 = Positive (CCW)
1 = Negative (CW)

Bits 14–12 **D2–D0**. Basic space vector bits. Used only in space vector PWM output generation.

Bits 11–10 **CMP6ACT1, CMP6ACT0**. Action on full compare output pin 6, PWM6/CMP6.

	Compare mode	**PWM mode**
00	Hold	Forced low
01	Reset	Active low
10	Set	Active high
11	Toggle	Forced high

Bits 9–8 **CMP5ACT1, CMP5ACT0**. Action on full compare output pin 5, PWM5/CMP5.

	Compare mode	**PWM mode**
00	Hold	Forced low
01	Reset	Active low
10	Set	Active high
11	Toggle	Forced high

Bits 7–6 **CMP4ACT1, CMP4ACT0**. Action on full compare output pin 4, PWM4/CMP4.

	Compare mode	PWM mode
00	Hold	Forced low
01	Reset	Active low
10	Set	Active high
11	Toggle	Forced high

Bits 5–4 **CMP3ACT1, CMP3ACT0**. Action on full compare output pin 3, PWM3/CMP3.

	Compare mode	PWM mode
00	Hold	Forced low
01	Reset	Active low
10	Set	Active high
11	Toggle	Forced high

Bits 3–2 **CMP2ACT1, CMP2ACT0**. Action on full compare output pin 2, PWM2/CMP2.

	Compare mode	PWM mode
00	Hold	Forced low
01	Reset	Active low
10	Set	Active high
11	Toggle	Forced high

Bits 1–0 **CMP1ACT1, CMP1ACT0**. Action on full compare output pin 1, PWM1/CMP1.

	Compare mode	PWM mode
00	Hold	Forced low
01	Reset	Active low
10	Set	Active high
11	Toggle	Forced high

Simple Compare Action Control Register (SACTR)

The action of simple compare output pins on compare event is defined by the 16-bit simple compare action control register (SACTR). The bit configuration of SACTR is shown in Figure 2–16. SACTR is double buffered. The condition on which the register is reloaded is defined by bits in COMCON.

Figure 2–16. Simple Compare Action Control Register (SACTR) — Address 7414h

15–8
Reserved

7–6	5	4	3	2	1	0
Reserved	SCMP3ACT1	SCMP3ACT0	SCMP2ACT1	SCMP2ACT0	SCMP1ACT1	SCMP1ACT0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

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

Bits 15–6 **Reserved**. Reads are zero and writes have no effect.

Bits 5–4 **SCMP3ACT1, SCMP3ACT0**. Action on simple compare output pin 3, the PWM9/CMP9.

00 = Forced low
01 = Active low
10 = Active high
11 = Forced high

Bits 3–2 **SCMP2ACT1, SCMP2ACT0**. Action on simple compare output pin 2, the PWM8/CMP8.

00 = Forced low
01 = Active Low
10 = Active high
11 = Forced high

Bits 1–0 **SCMP1ACT1, SCMP1ACT0**. Action on simple compare output pin 1, the PWM7/CMP7.

00 = Forced low
01 = Active low
10 = Active high
11 = Forced high

2.4.4 Compare Unit Interrupts

There is a maskable interrupt flag for each compare unit in EVIFRA and EVIFRC. An interrupt flag of a compare unit is set two CPU clock cycles after a compare match, if compare operation is enabled. An interrupt request is generated to the CPU by the flag, if it is unmasked and no other unmasked interrupts in the same group of higher priority are pending.

2.4.5 Compare Unit Reset

When any reset event occurs, all registers associated with the compare units are reset to 0 and all compare output pins are put in a high-impedance state.

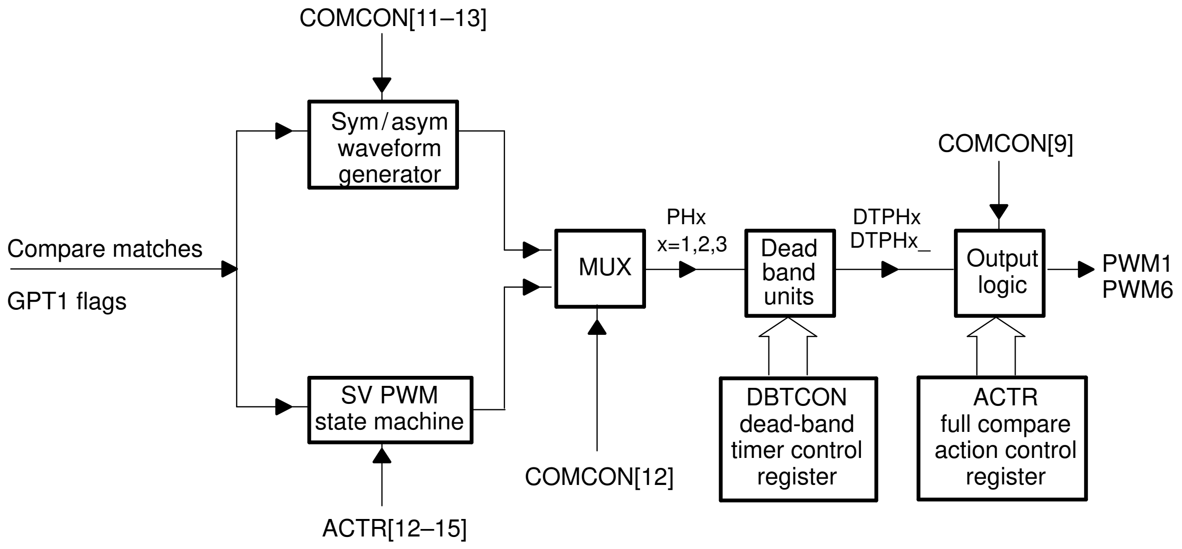
2.5 PWM Circuits Associated with Full Compare Units

The PWM circuits associated with full compare units make it possible to generate 6 PWM output channels with programmable dead-band and output polarity. The PWM Circuits functional block diagram is shown in Figure 2–17. It includes the following functional units:

- ❑ Asymmetric/Symmetric Waveform Generators
- ❑ Programmable Dead-Band Unit (DBU)
- ❑ Output Logic
- ❑ Space Vector (SV) PWM State Machine

The asymmetric/symmetric waveform generators are the same as that of the GP timer and simple compare units; thus, they will not be discussed. Dead-band units and output logic are discussed in the following subsections. Space vector PWM state machine and space vector PWM technique are described later in this chapter.

Figure 2–17. PWM Circuits Block Diagram



The PWM circuits are designed to minimize the CPU overhead and user intervention in generating pulse width modulated wave forms used in motor control and motion control applications. PWM generation with full compare units and associated PWM circuits are controlled by the following control registers: T1CON, COMCON, ACTR, and DBTCON.

2.5.1 PWM Generation Capability of Event Manager

The PWM waveform generation capability of the event manager is summarized as follows:

- ❑ 9 independent PWM outputs
- ❑ Programmable dead-band for the PWM output pairs associated with full compare units, from 0 to 2048 CPU clock cycles, 102.4 μs if CPU clock cycle is 50 ns.
- ❑ Minimum dead band duration of one CPU clock cycle
- ❑ Minimum PWM pulsewidth and pulsewidth increment/decrement of one CPU clock cycle
- ❑ 16-bit maximum PWM resolution
- ❑ On-the-fly change of PWM carrier frequency (double buffered period registers)
- ❑ On-the-fly change of PWM pulsewidths (double buffered compare registers)
- ❑ Power Drive Protection Interrupt
- ❑ Programmable generation of asymmetric, symmetric, and space vector PWM waveforms
- ❑ Minimum CPU overhead because of the autoreloading of compare and period registers

2.5.2 Programmable Dead-Band Unit

The programmable dead-band unit features:

- ❑ One 16-bit dead-band control register, DBTCON (R/W)
- ❑ One input clock prescaler: x/1, x/2, x/4, x/8
- ❑ CPU clock input
- ❑ Three 8-bit down counting timers
- ❑ Control logic

Dead-Band Timer Control Register (DBTCON)

The operation of the dead-band unit is controlled by the dead-band timer control register (DBTCON). The bit description of DBTCON is given in Figure 2–18.

Figure 2–18. Dead-Band Timer Control Register (DBTCON) — Address 7415h

15	14	13	12	11	10	9	8
DBT7	DBT6	DBT5	DBT4	DBT3	DBT2	DBT1	DBT0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

7	6	5	4	3	2–0
EDBT3	EDBT2	EDBT1	DBTPS1	DBTPS0	Reserved
RW–0	RW–0	RW–0	RW–0	RW–0	

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

Bits 15–8 **DBT7 (MSB)–DBT0 (LSB)**. Dead-band timer period. These bits define the period value of the three 8-bit dead-band timers.

Bit 7 **EDBT3**. Dead-band Timer 3 enable (for pins PWM5/CMP5 and PWM6/CMP6 of Full Compare Unit 3).

0 = Disable
1 = Enable

Bit 6 **EDBT2**. Dead-band Timer 2 enable (for pins PWM3/CMP3 and PWM4/CMP4 of Full Compare Unit 2).

0 = Disable
1 = Enable

Bit 5 **EDBT1**. Dead-band Timer 1 enable (for pins PWM1/CMP1 and PWM2/CMP2 of Full Compare Unit 1).

0 = Disable
1 = Enable

Bits 4–3 **DBTPS1, DBTPS0**. Dead-band Timer prescaler.

00 = x/1
01 = x/2
10 = x/4
11 = x/8
x = CPU clock frequency

Bits 2–0 **Reserved**. Reads are zero and writes have no effect.

Inputs and Outputs of Dead-Band Unit

The inputs to the dead-band unit are PH1, PH2, and PH3 from the asymmetric/symmetric waveform generators of full compare units 1, 2, and 3, respectively.

The outputs of the dead-band unit are DTPH1, DTPH1_, DTPH2, DTPH2_, DTPH3, and DTPH3_, corresponding to PH1, PH2, and PH3, respectively.

Dead Band Generation

For each input signal PHx, two output signals, DTPHx and DTPHx_, are generated. When dead-band is not enabled for the compare unit and its associated outputs, the two signals will be exactly the same. When the Dead-band Unit is enabled for the compare unit, the transition edges of the two signals will be separated by a time interval called dead-band. This time interval is determined by the DBTCON bits. Assume the value in DBTCON[15–8] is *m* and the value in DBTCON[4–3] corresponds to prescaler *x/p.* Then the dead-band value is (*p** *m*) CPU clock cycles.

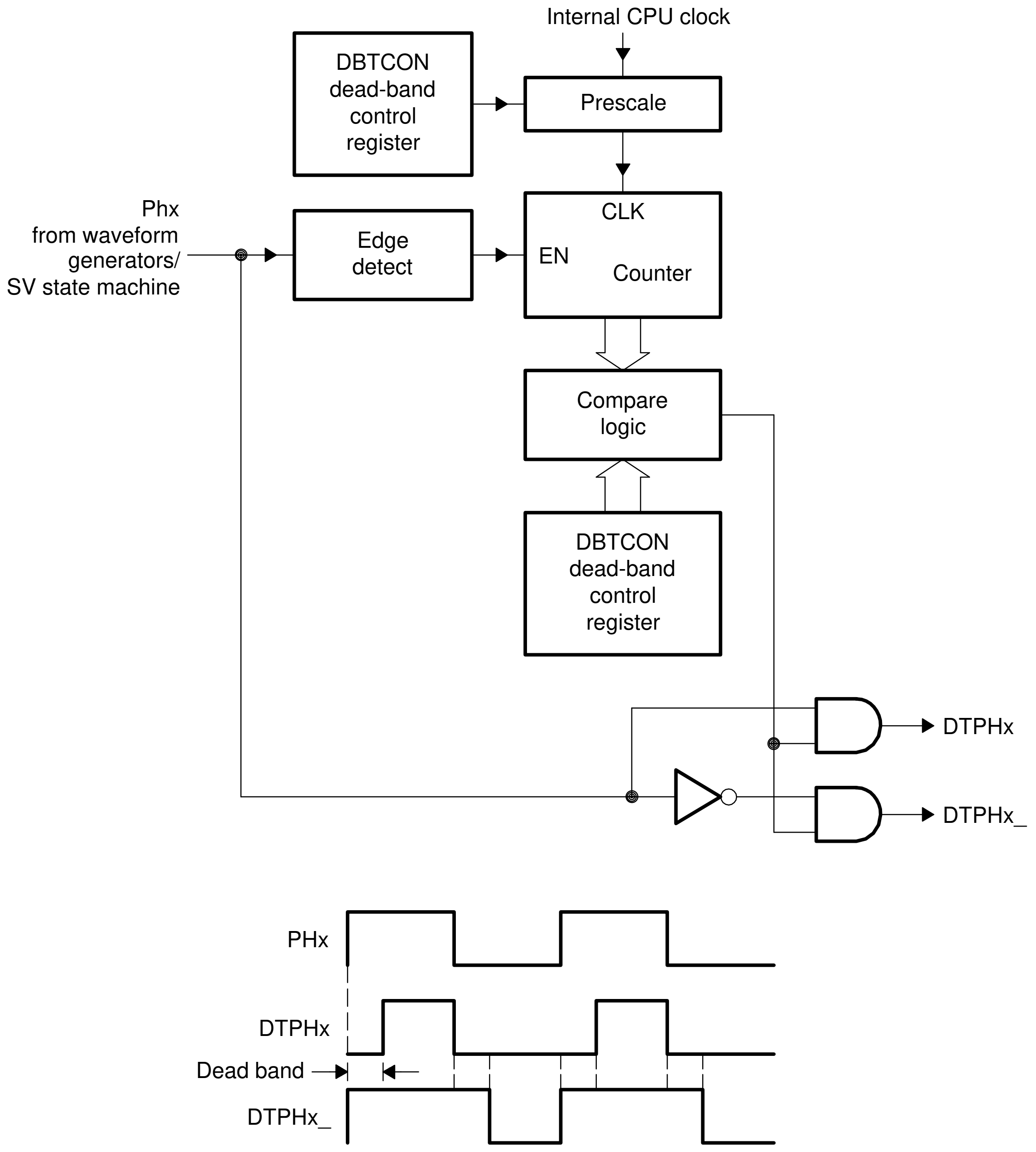
Table 2–8 shows the dead band generated by typical bit combinations in DBTCON. The values are based on a 50 ns device. Figure 2–19 shows the block diagram of the dead-band logic for one full compare unit.

Table 2–8. Dead Band Generation Examples

DBT7–DBT0 (*m*) (DBTCON[15–8])	**DBTPS1–DBTPS0 (*p*) (DBTCON[4–3])**			
	11 (P = 8)	**10 (P = 4)**	**01 (P = 2)**	**00 (P = 1)**
00h	0	0	0	0
01h	0.40	0.20	0.10	0.05
02h	0.80	0.40	0.20	0.10
03h	1.20	0.60	0.30	0.15
04h	1.60	0.80	0.40	0.20
05h	2.00	1.00	0.50	0.25
06h	2.40	1.20	0.60	0.30
07h	2.80	1.40	0.70	0.35
08h	3.20	1.60	0.80	0.40

Note: Table values are given in µs.

Figure 2–19. Dead-Band Unit Block Diagram (x = 1, 2, or 3)


Internal CPU clock
DBTCON dead-band control register
Prescale
Phx from waveform generators/ SV state machine
Edge detect
CLK
EN
Counter
Compare logic
DBTCON dead-band control register
DTPHx
DTPHx_
PHx
DTPHx
Dead band
DTPHx_

Other Important Features of Dead-Band Units

The dead-band unit is designed to assure no overlap between the turn-on period of the upper and lower devices controlled by the two compare/PWM outputs associated with each full compare unit under any operating situation including the situations when the user has loaded a dead-band value greater than that of the duty cycle and when the duty cycle is 100% or 0%. As a result, the compare/PWM outputs associated with a full compare unit do not reset to an inactive state at the end of a period when dead band is enabled for the full compare unit.

Dead-band is disabled when a full compare unit is in compare mode of operation.

Example 2–7 shows an initialization routine for full compare units for symmetric PWM waveform with dead band unit activated.

Example 2–7. Initialization Routine for Dead Band Generation

```
;*******************************************************************************
; The following section of codes initializes symmetric PWM with dead band
;*******************************************************************************
             LDPK   #232                      ; DP  => EV registers
;*******************************************************************************
; Configure ACTR
;*******************************************************************************
             SPLK   #0000011001100110b, ACTR   ; GP Timer control
;                    ||||||||||||||||
;                    FEDCBA9876543210
;
* bit  15     0      SV rotation direction (for here not applicable)
* bits 12-14  000    Space vector bits (for here not applicable)
* bits 10-11  01     PWM6 active low
* bits 8-9    10     PWM5 active high
* bits 6-7    01     PWM4 active low
* bits 4-5    10     PWM3 active high
* bits 2-3    01     PWM2 active low
* bits 0-1    10     PWM1 active high
;*******************************************************************************
; Configure DBTCON
;*******************************************************************************
             SPLK   #0000010111100000b, DBTCON ; Timer control
;                    ||||||||||||||||
;                    FEDCBA9876543210
* bits 8-15   101    : 5 cycles of dead band
* bit 7       1      : Enable DB for PWM 5/6
* bit 6       1      : Enable DB for PWM 3/4
* bit 5       1      : Enable DB for PWM 1/2
* bits 3-4    00     : Timer prescaler for DB unit /1
* bits 0-2    000    : Reserved
;*******************************************************************************
; Initialize compare registers
;*******************************************************************************
             SPLK   #0008h, CMPR1
             SPLK   #000Ch, CMPR2
             SPLK   #0011h, CMPR3
;*******************************************************************************
; Initialize T1PER  register
;*******************************************************************************
             SPLK   #0014h, T1PER
```

2.5.3 Output Logic

The output logic circuit determines the polarity and/or the action that must be taken on a compare match for outputs PWMx/CMPx, for x = 1–9. The outputs associated with each full compare unit can be specified to be active low, active high, forced low, or forced high when the full compare unit is in PWM mode. They can be specified to hold, set, reset or toggle when the compare unit is in compare mode. The polarity and/or the action of full and simple compare/PWM outputs can be programmed by proper configuration of bits in ACTR and SACTR. The six full compare/PWM and three simple compare/PWM output pins can all be put in a high-impedance state by any of the following:

- Software resetting of the COMCON[9] and COMCON[8] bits, respectively.
- Hardware pulling PDPINT low when PDPINT is unmasked.
- The occurrence of any reset event.

Active PDPINT (when enabled) and system reset override bits in COMCON, ACTR, and SACTR.

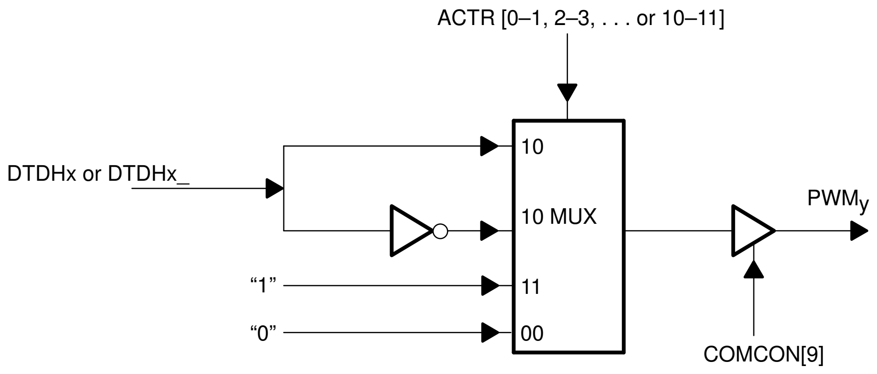
Figure 2–20 shows a block diagram of the output logic circuit (OLC). The inputs of Output Logic for full and simple compare units are:

- DTPH1, DTPH1_, DTPH2, DTPH2_, DTPH3, and DTPH3_ from the dead-band unit and full compare match signals
- Outputs from simple compare units asymmetric/symmetric waveform generator
- Bits of ACTR and SACTR
- PDPINT and RESET

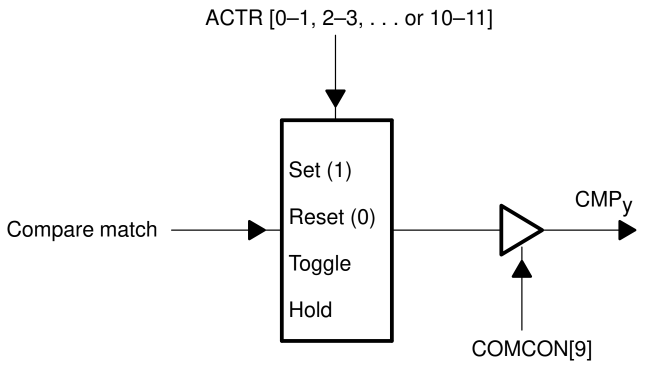
The outputs of Output Logic for full and simple compare units are:

- PWMx/CMPx, x = 1–9

Figure 2–20. Output Logic Block Diagram (x = 1, 2, or 3; y = 1, 2, 3, 4, 5, or 6)



Output logic for PWM mode



Output logic for compare mode

2.6 PWM Waveform Generation with Compare Units and PWM Circuits

2.6.1 PWM Signals

A pulsewidth-modulated (PWM) signal is a sequence of pulses with changing pulsewidths. The pulses are spread over a number of fixed-length periods so that there is one pulse in each period. The fixed period is called the PWM (carrier) period. Its inverse is called the PWM (carrier) frequency. The widths of the PWM pulses are determined or modulated from pulse to pulse according to another sequence of desired values, the modulating signal.

In a motor control system, PWM signals are used to control the on and off time of switching power devices that deliver the desired current and energy to the motor windings (see Figure 2–23 on page 2-66). The shape and frequency of the phase currents and the amount of energy delivered to the motor windings control the required speed and torque of the motor. In this case, the command voltage or current to be applied to the motor is the modulating signal. The frequency of the modulating signal is typically much lower than the PWM carrier frequency.

PWM Signal Generation

To generate a PWM signal, an appropriate timer is needed to repeat a counting period that is the same as the PWM period. A compare register is used to hold the modulating values. The value of the compare register is constantly compared with the value of the timer counter. When the values match, a transition (from low to high, or high to low) happens on the associated output. When a second match is made between the values, or when the end of a timer period is reached, another transition (from high to low, or low to high) happens on the associated output. In this way, an output pulse is generated whose on (or off) duration is proportional to the value in the compare register. This process is repeated for each timer period with different (modulating) values in the compare register. As a result, a PWM signal is generated at the associated output.

Dead Band

In many motion/motor and power electronics applications, two (an upper and/ or lower) power devices are placed in series on one power converter leg, and the turn-on periods of the two devices must not overlap with each other in order to avoid the shoot-through fault. Thus a pair of nonoverlapping PWM outputs is often required to turn on and off the two devices properly. A dead time (dead-band) is often inserted between the turning off of one transistor and the turning on of the other transistor. This delay allows complete turning-off of one transistor before the turning-on of the other transistor. The required time delay is specified by the turning-on and turning-off characteristics of the power transistors and the load characteristics in a specific application.

2.6.2 Generation of PWM Outputs with Event Manager

Each of the three full compare units together with GP Timer 1, the dead-band unit and the output logic in the event manager module can be used to generate a pair of PWM outputs with programmable dead-band and output polarity on two dedicated device pins. There are six such dedicated PWM output pins associated with the three full compare units in the EV module. These six dedicated output pins can be used to conveniently control 3-phase AC induction or brushless DC motors. The flexibility of output behavior control by the full compare action control register (ACTR) also makes it easy to control switched reluctance and synchronous reluctance motors in a wide range of applications. The PWM circuits can also be used to conveniently control other types of motors such as DC brush, and stepper motors in single or multi-axis control applications. The three simple compare units with GP Timer 1 or 2 can be used to generate another three PWM outputs in case dead-band is not necessary or is generated by circuits off the chip. Each GP timer compare unit, if desired, can generate a PWM output based on its own timer.

Asymmetric and Symmetrical PWM Generation

Both asymmetric and symmetric PWM waveforms can be generated by every compare unit on the EV module. In addition, the three full compare units together can be used to generate 3-phase symmetric space vector PWM outputs. PWM generation with GP timer compare units has been described in the GP timer sections. PWM generation with simple compare units is similar to GP timer compare units except that different control registers are used and that either GP Timer 1 or 2 can be chosen as the time base. Generation of PWM outputs with full compare units is discussed in this section.

2.6.3 Register Setup for PWM Generation

All three kinds of PWM waveform generation with full compare units and associated circuits require configuration of the same Event Manager registers. The setup process for PWM generation includes the following steps:

- Setup and load ACTR.
- Setup and load DBTCON, if dead-band is to be used.
- Initialize CMPRx.
- Setup and load COMCON without enabling compare operation.
- Setup and load COMCON to enable compare operation.
- Setup and load T1CON to start the operation.
- Rewrite CMPRx with newly determined values.

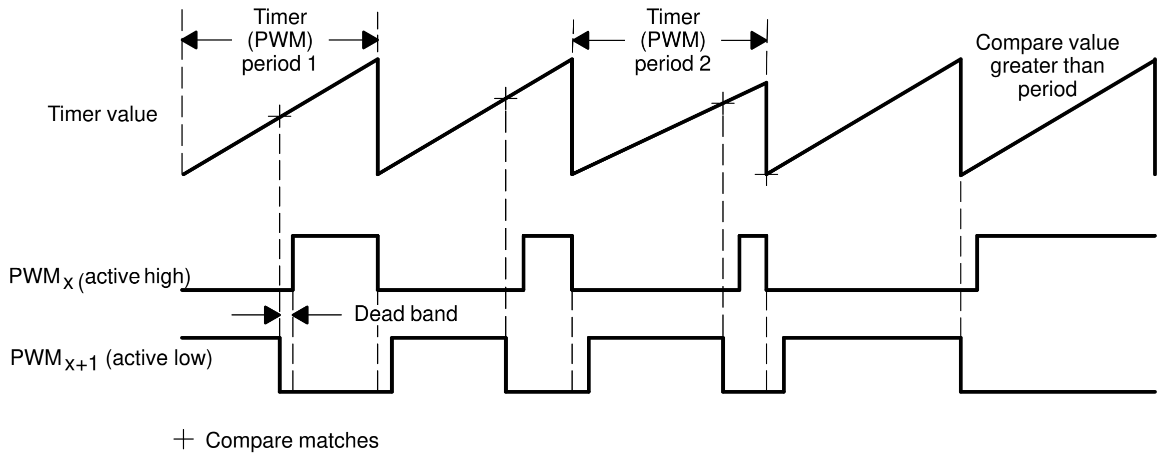
Note:

Before T1CON is written to start the operation, COMCON must be written twice to ensure that all full compare outputs come up in the correct (inactive) states.

2.6.4 Asymmetric PWM Wave Form Generation

The edge-triggered or asymmetric PWM signal is characterized by modulated pulses which are not centered with respect to the PWM period, as shown in Figure 2–21. The width of each pulse can only be changed from one side of the pulse.

Figure 2–21. Asymmetric PWM waveform generation with full compare unit and PWM Circuits (x = 1, 3, or 5)



To generate an Asymmetric PWM signal, GP Timer 1 is put in continuous-up counting mode. Its period register is loaded with a value corresponding to the desired PWM carrier period. The COMCON is configured to enable the compare operation, set the selected output pins to be PWM outputs and enable the outputs. If dead-band is enabled, a desired value corresponding to the dead-band should be written by software to the 8 MSBs of DBTCON as period for the 8-bit dead-band timers. One dead-band value is used for all PWM output channels.

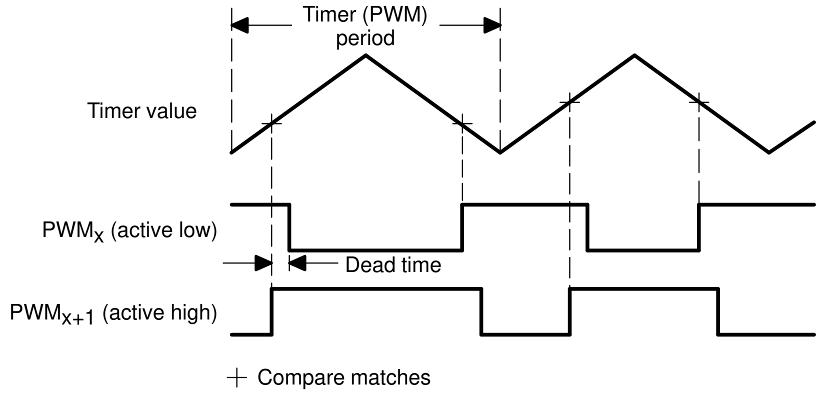
By proper configuration of ACTR with software, a normal PWM signal can be generated on one output associated with a full compare unit while the other is held low (or off) or high (or on), at the beginning, middle, or end of a PWM period. Such software controlled flexibility of PWM outputs is particularly useful in switched reluctance motor control applications.

After GP Timer 1 is started, the compare registers are rewritten every PWM period with newly determined compare values to adjust the width (the duty cycle) of PWM outputs that control the switch-on and off duration of the power devices. Since the compare registers are shadowed, new value can be written to them at any time during a period. For the same reason, new values can be written to the action and period registers at any time during a period to change PWM period or to force changes in PWM output definition.

2.6.5 Symmetric PWM Waveform Generation

A centered or symmetric PWM signal is characterized by modulated pulses which are centered with respect to each PWM period. The advantage of a symmetric PWM signal over an asymmetric PWM signal is that it has two inactive zones of same duration — at the beginning and at the end of each PWM period. This symmetry has been shown to cause less harmonics than an asymmetric PWM signal in the phase currents of an AC motor such as induction and DC brushless motors when sinusoidal modulation is used. Figure 2–22 shows two examples of symmetric PWM waveforms.

Figure 2–22. Symmetric PWM waveform generation with full compare units and PWM Circuits (x = 1, 3, or 5)



The generation of a symmetric PWM waveform with full compare unit is similar to the generation of an asymmetric PWM waveform. The only exception is that GP Timer 1 now needs to be put in continuous-up/down counting mode.

There are usually two compare matches in a PWM period in symmetric PWM waveform generation, one during the upward counting before period match and another during downward counting after period match. A new compare value can become effective after the period match (reload on period), making it possible to advance or delay the second edge of a PWM pulse. An application of this feature is PWM waveform modification to compensate for current errors caused by the dead-band in AC motor control.

Again, since the compare registers are shadowed, a new value can be written to them at any time during a period. For the same reason, new values can be written to the action and period registers at any time during a period to change PWM period or to force changes in PWM output definition.

2.7 Space Vector PWM

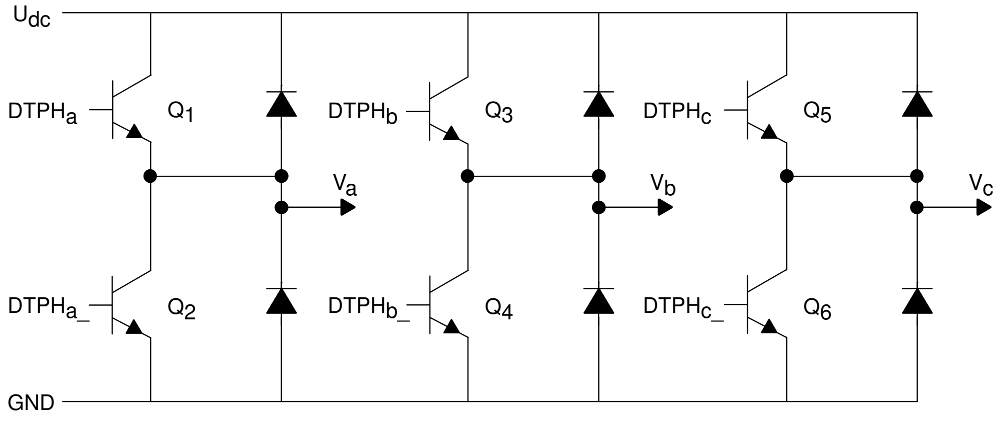
2.7.1 Space Vector PWM Theory Overview

Space vector PWM refers to a special switching scheme of the six power transistors of a 3-phase power converter. It generates minimum harmonic distortion to the currents in the windings of a 3-phase AC motor. It also provides more efficient use of supply voltage in comparison with sinusoidal modulation method.

3-Phase Power Inverter

The structure of a typical 3-phase power inverter is shown in Figure 2–23, where V_a, V_b, and V_c are the voltages applied to the motor windings. The six power transistors are controlled by $DTPH_x$ and $DTPH_{x\_}$ (x = a, b, and c). When an upper transistor is switched on ($DTPH_x = 1$), the lower transistor is switched off ($DTPH_{x\_} = 0$). Thus, the on and off states of the upper transistors (Q1, Q3, and Q5) or, equivalently, the state of DTPHx (x = a, b, and c) are sufficient to evaluate the applied motor voltage U_{out}.

Figure 2–23. 3-Phase Power Inverter Schematic Diagram



Switching Patterns of Power Inverter and the Basic Space Vectors

When an upper transistor of a leg is on, the voltage V_x (x = a, b, or c) applied by the leg to the corresponding motor winding is equal to the voltage supply U_{dc}. When it is off, the voltage applied is zero. The on and off switching of the upper transistors ($DTPH_x$, x = a, b, or c) have eight possible combinations. The eight combinations and the derived motor line-to-line and phase voltage in terms of DC supply voltage U_{dc} are shown in Table 2–9, where a, b, and c represent the values of $DTPH_a$, $DTPH_b$, and $DTPH_c$, respectively.

Table 2–9. Switching Patterns of A 3-Phase Power Inverter

a	b	c	$V_{a0}(U_{dc})$	$V_{b0}(U_{dc})$	$V_{c0}(U_{dc})$	$V_{ab}(U_{dc})$	$V_{bc}(U_{dc})$	$V_{ca}(U_{dc})$
0	0	0	0	0	0	0	0	0
0	0	1	–1/3	–1/3	2/3	0	–1	1
0	1	0	–1/3	2/3	–1/3	–1	1	0
0	1	1	–2/3	1/3	1/3	–1	0	1
1	0	0	2/3	–1/3	–1/3	1	0	–1
1	0	1	1/3	–2/3	1/3	1	–1	0
1	1	0	1/3	1/3	–2/3	0	1	–1
1	1	1	0	0	0	0	0	0

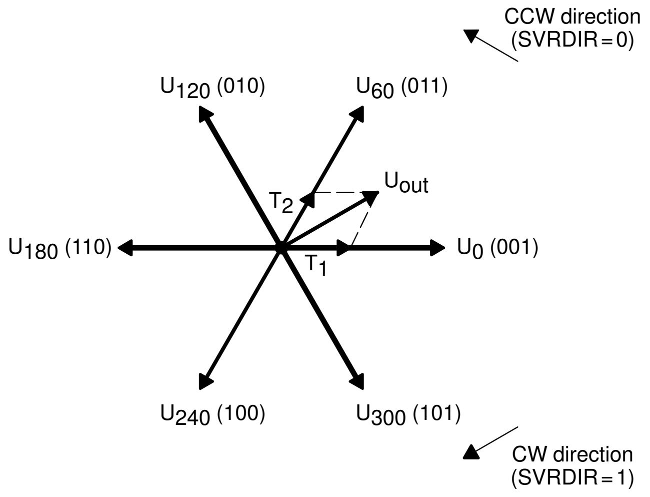
Note: 0 = off and 1 = on

Mapping the phase voltages corresponding to the eight combinations onto the d–q plane by performing a d–q transformation (which is equivalent to an orthogonal projection of the 3-vectors (a b c) onto the two dimensional plane perpendicular to the vector (1,1,1), the d–q plane), results in six nonzero vectors and two zero vectors. The nonzero vectors form the axes of a hexagonal. The angle between two adjacent vectors is 60 degrees. The two zero vectors are at the origin. These 8 vectors are called the basic space vectors and are denoted by U_0, U_{60}, U_{120}, U_{180}, U_{240}, U_{300}, O_{000}, and O_{111}. The same transformation can be applied to the demanded voltage vector U_{out} to be applied to a motor. Figure 2–24 shows the projected vectors and the projected desired motor voltage vector U_{out}.

The d axis and q axis of a d–q plane correspond here to the horizontal and vertical geometrical axes of the stator of an AC machine.

The objective of space vector PWM method is to approximate the motor voltage vector U_{out} by a combination of these eight switching patterns of the six power transistors.

Figure 2–24. Basic Space Vectors and Switching Patterns



The binary representations of two adjacent basic vectors are different in only one bit. That is, only one of the upper transistors switches when the switching pattern switches from U_x to U_{x+60} or from U_{x+60} to U_x. Also, the zero vectors O_{000} and O_{111} apply no voltage to the motor.

Approximation of Motor Voltage with Basic Space Vectors

The projected motor voltage vector U_{out}, at any given time, falls in one of the six sectors. Thus, for any PWM period, it can be approximated by vector sum of two vector components lying on the two adjacent basic vectors:

$$U_{out} = \frac{T_1\,U_x}{T_p} + \frac{T_2\,U_{x+60}}{T_p} + \frac{T_0\,(O_{000} \text{ or } O_{111})}{T_p}$$

where T_0 is given by $T_p – T_1 – T_2$ and T_p is the PWM carrier period. The third term on the right side of the equation above doesn’t affect the vector sum U_{out}. The generation of U_{out} is beyond the scope of this context. See publications on space vector PWM and motor control theory, for example, *The Field Orientation Principle in Control of Induction Motors* by Andrzej M. Trzynadlowski, for more details.

The above approximation means that the upper transistors must have the on and off pattern corresponding to U_x and U_{x+60} for the time duration of T_1 and T_2, respectively, in order to apply voltage U_{out} to the motor. The including of zero basic vectors helps to balance the turn on and off periods of the transistors and thus their power dissipation.

2.7.2 Space Vector PWM Waveform Generation with Event Manager

Software

The EV module has built-in hardware to greatly simplify the generation of symmetric space vector PWM waveforms. To generate space vector PWM outputs, the user software must:

- ❑ Configure ACTR to define the polarity of full compare output pins.
- ❑ Configure COMCON to enable compare operation and space vector PWM mode, and set the reload condition for ACTR and CMPRx to be underflow.
- ❑ Put GP Timer 1 in continuous-up/down counting mode to start the operation.

Note:

Enabling space vector PWM mode automatically sets all full compare output pins as PWM outputs.

The user software then needs to determine the voltage U_{out} to be applied to the motor phases in the two dimensional d–q plane, decompose U_{out} and determine for each PWM period:

- ❑ The two adjacent vectors, U_x and U_{x+60}.
- ❑ The parameters T_1, T_2, and T_0.
- ❑ Write the switching pattern corresponding to U_x in ACTR[14–12] and 1 in ACTR[15], or the switching pattern of U_{x+60} in ACTR[14–12] and 0 in ACTR[15].
- ❑ Put (1/2 T1) in CMPR1 and (1/2 T1 + 1/2 T2) in CMPR2.

Space Vector PWM Hardware

The space vector PWM hardware in the EV module does the following to complete a space vector PWM period.

- ❑ At the beginning of each period, sets the PWM outputs to the (new) pattern U_y defined by ACTR[14–12].
- ❑ On the first compare match during up counting between CMPR1 and GP Timer 1 at (1/2 T1), switches the PWM outputs to the pattern of U_{y+60} if ACTR[15] is 1, or to the pattern of U_y if ACTR[15] is 0. ($U_{0-60} = U_{300}$, $U_{360+60} = U_{60}$).

- ❑ On the second compare match during up counting between CMPR2 and GP Timer 1 at (1/2 T1 + 1/2 T2), switches the PWM outputs to the pattern (000) or (111), whichever differs from the second pattern by one bit.
- ❑ On the first compare match during down counting between CMPR2 and GP Timer 1 at (1/2 T1 + 1/2 T2), switches the PWM outputs back to the second output pattern.
- ❑ On the second compare match during down count between CMPR1 and GP Timer 1 at (1/2 T1), switches the PWM outputs back to the first pattern.

Space Vector PWM Waveforms

The space vector PWM waveforms generated are symmetric with respect to the middle of each PWM period. Therefore, it is called symmetric space vector PWM generation method. Figure 2–25 shows examples of the symmetric space vector PWM waveforms.

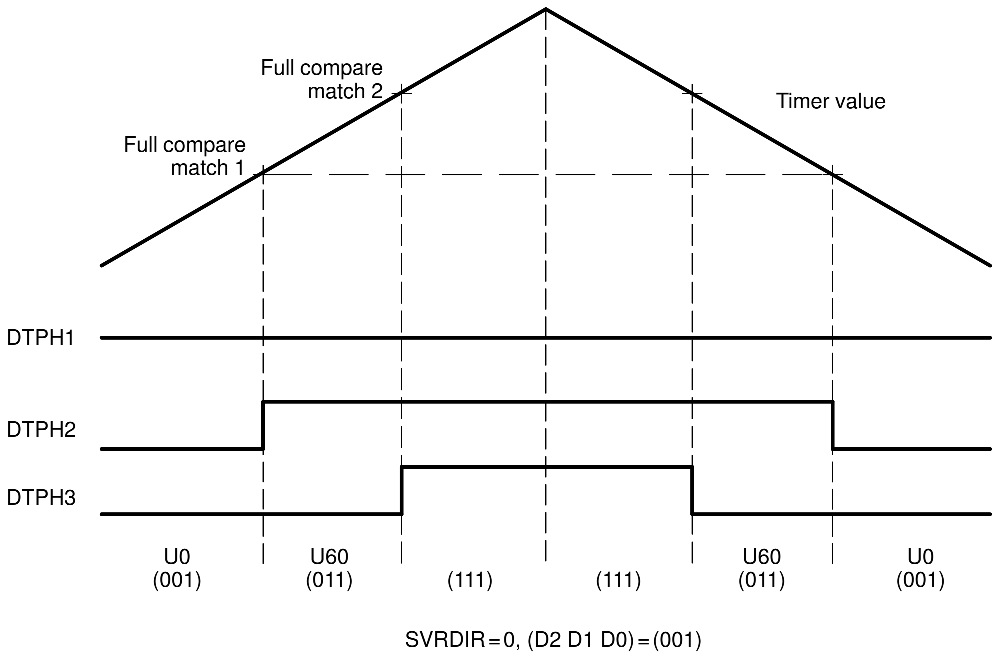
Unused Full Compare Register

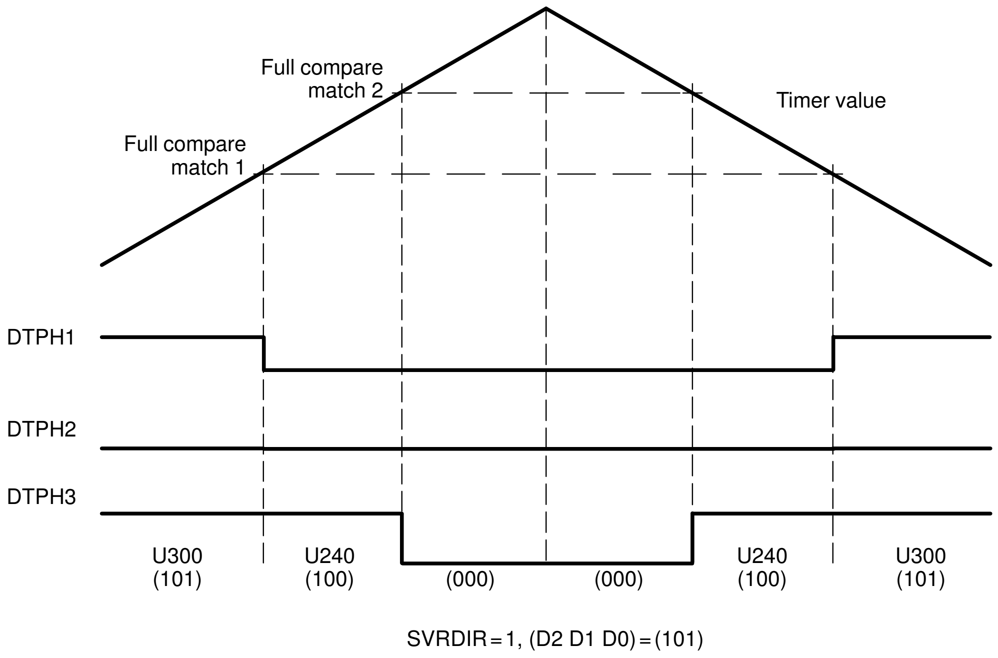
Only two full compare registers are used in space vector PWM outputs generation. The third full compare register, however, is still constantly compared with GP Timer 1. When a compare match happens, the corresponding compare interrupt flag will still be set and an interrupt request will be generated if the flag is unmasked and no other unmasked interrupts in the same group of higher priority are pending. Therefore, the compare register that is not used in the space vector PWM outputs generation can still be used to time events happening in a specific application. Also, because of the extra delay introduced by the state machine, the full compare output transitions are delayed by two CPU clock cycles in space vector PWM mode.

2.7.3 Space Vector PWM Boundary Conditions

All three full compare outputs become inactive when both full compare registers (CMPR1 and CMPR2) are loaded with a zero value in space vector PWM mode. In general, it is the user’s responsibility to assure that (CMPR1) $\leq$ (CMPR2) $\leq$ (T1PR) in the space vector PWM mode. Otherwise, unpredicted behavior may result.

Figure 2–25. Symmetric Space Vector PWM Waveforms

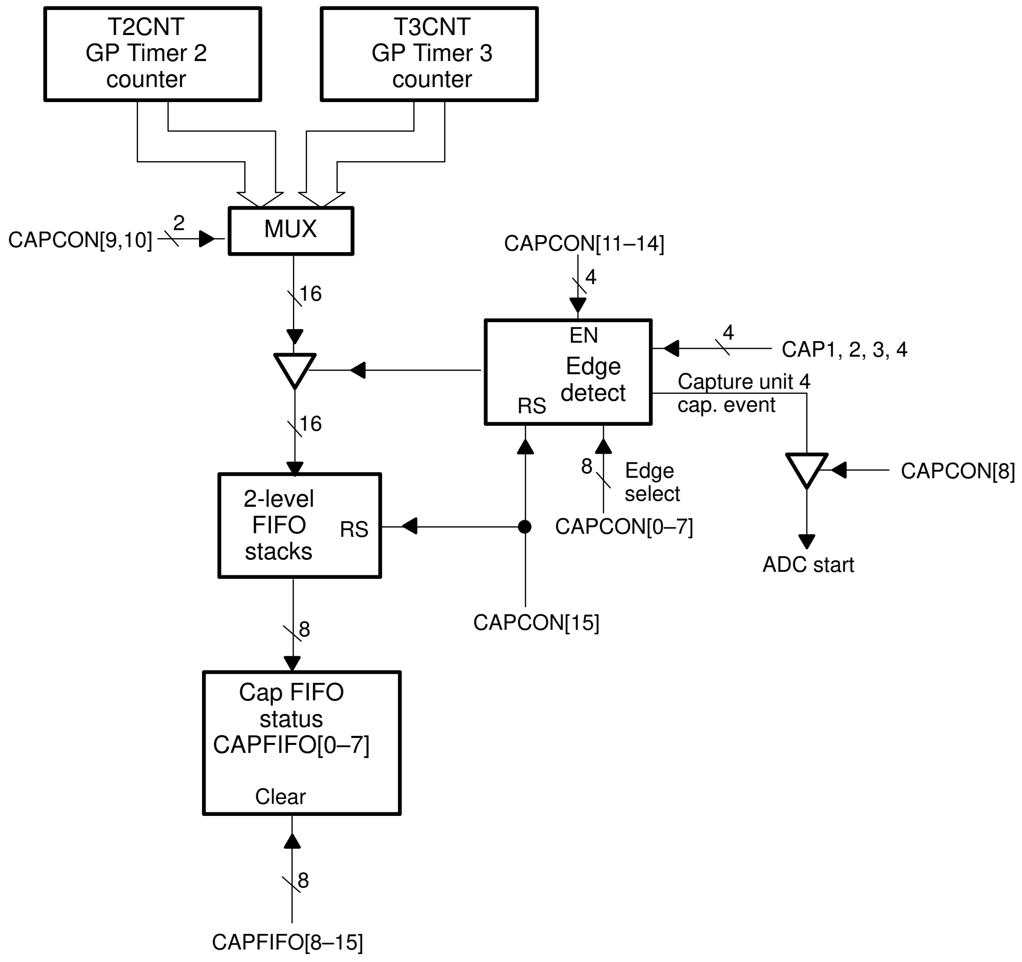




2.8 Capture Units

Capture units enable logging of transitions on capture input pins. There are four capture units, Capture Units 1, 2, 3, and 4. Each capture unit is associated with a capture input pin. Each capture unit can choose GP Timer 2 or 3 as its time base. The value of GP Timer 2 or 3 is captured and stored in the corresponding 2-level-deep FIFO stack when a specified transition is detected on a capture input pin, CAPx. Figure 2–26 shows the block diagram of a capture unit.

Figure 2–26. Capture Units Block Diagram



2.8.1 Features

Capture units have the following features:

- ❏ One 16-bit capture control register, CAPCON (R/W)
- ❏ One 16-bit capture FIFO status register, CAPFIFO (8 MSBs are read only, 8 LSBs are write only)
- ❏ Selection of GP Timer 2 or 3 as the time base
- ❏ Four 16-bit 2-level-deep FIFO stacks, one for each capture unit
- ❏ Four Schmitt-triggered capture input pins CAP1, CAP2, CAP3, and CAP4, one input pin per each capture unit (All inputs are synchronized with the CPU clock. In order for a transition to be captured, the input must hold at its current level to meet two rising edges of the CPU clock. The input pins CAP1 and CAP2 can also be used as QEP inputs to QEP circuit.)
- ❏ User-specified transition (rising edge, falling edge, or both edges) detection
- ❏ Four maskable flags, one for each capture unit

2.8.2 Operation of Capture Units

Capture Unit Time Base Selection

Either GP Timer 2 or 3 can be selected by Capture Units 1 and 2 or by Capture Units 3 and 4. In this way, two different GP timers can be used at the same time, one for Capture Units 1 and 2, and the other for Capture Units 3 and 4.

Capture operation does not affect the operation of any GP Timer or the compare/PWM operations associated with any GP Timer.

Capture Operation

After a capture unit is enabled, a specified transition on the associated input pin causes the counter value of the selected GP timer to be locked into the corresponding FIFO stack. At the same time, the corresponding interrupt flag is set, and, if the flag is unmasked and no other unmasked interrupts in the same group of higher priority are pending, an interrupt request is generated to the CPU core. The corresponding status bits in CAPFIFO are adjusted to reflect the new status of the FIFO stack each time a new counter value is captured into a FIFO stack. The latency from the time a transition happens on a capture input to the time the counter value of selected GP Timer is locked is 3.5–4.5 CPU clock cycles.

All capture unit registers are cleared to 0 when RESET input goes low.

Capture Unit Setup

The following register setup is performed for a capture unit to function properly:

1) Initialize the CAPFIFO. Clear the appropriate status bits.
2) Set the selected GP timer in one of its operating mode.
3) Set the associated GP timer compare register or GP timer period register if necessary.
4) Set up CAPCON.

2.8.3 Capture Unit Registers

The operation of capture units is controlled by two 16-bit control registers CAPCON and CAPFIFO. T2CON and T3CON registers are also used since the time base for capture circuits is provided by GP Timer 2 or 3. The CAPCON is also used to control the operation of QEP circuit. As all other EV registers, these registers are all data memory mapped and hence can be treated by user software as data memory locations. Table 2–4 on page 2-9 shows the addresses of these registers.

Capture Control Register (CAPCON)

Figure 2–27. Capture Control Register (CAPCON) — Address 7420h

15	14–13	12	11	10	9	8
CAPRES	CAPQEPN	CAP3EN	CAP4EN	CAP34TSEL	CAP12TSEL	CAP4TOADC
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

7–6	5–4	3–2	1–0
CAP1EDGE	CAP2EDGE	CAP3EDGE	CAP4EDGE
RW–0	RW–0	RW–0	RW–0

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

Bit 15 **CAPRES**. Capture reset.

0 = Clear all registers of capture units and QEP circuit to 0.
1 = No action

Bits 14–13 **CAPQEPN**. Capture Units 1 and 2 and QEP circuit control.

00 = Disable Capture Units 1 and 2, and QEP circuit. FIFO stacks retain their contents.
01 = Enable Capture Units 1 and 2. Disable QEP circuit.
10 = Reserved
11 = Enable QEP circuit. Disable Capture Units 1 and 2. Bits 4–7 and 9 are ignored.

Bit 12 **CAP3EN**. Capture Unit 3 control.

0 = Disable Capture Unit 3. FIFO stack of Capture Unit 3 retains its contents.
1 = Enable Capture Unit 3

Bit 11 **CAP4EN**. Capture Unit 4 control.

0 = Disable Capture Unit 4. FIFO stack of Capture Unit 4 retains its contents.
1 = Enable Capture Unit 4

Bit 10	**CAP34TSEL**. GP timer selection for Capture Units 3 and 4.
	0 = Select GP Timer 2
	1 = Select GP Timer 3
Bit 9	**CAP12TSEL**. GP timer selection for Capture Units 1 and 2.
	0 = Select GP Timer 2
	1 = Select GP Timer 3
Bit 8	**CAP4TOADC**. Capture Unit 4 event starts ADC.
	0 = No action
	1 = Start ADC when the CAP4INT flag is set.
Bits 7–6	**CAP1EDGE**. Edge detection control for Capture Unit 1.
	00 = No detection
	01 = Detect rising edge
	10 = Detect falling edge
	11 = Detect both edges
Bits 5–4	**CAP2EDGE**. Edge detection control for Capture Unit 2.
	00 = No detection
	01 = Detect rising edge
	10 = Detect falling edge
	11 = Detect both edges
Bits 3–2	**CAP3EDGE**. Edge detection control for Capture Unit 3.
	00 = No detection
	01 = Detect rising edge
	10 = Detect falling edge
	11 = Detect both edges
Bits 1–0	**CAP4EDGE**. Edge detection control for Capture Unit 4.
	00 = No detection
	01 = Detect rising edge
	10 = Detect falling edge
	11 = Detect both edges

Capture FIFO Status Register (CAPFIFO)

CAPFIFO contains the status bits for each of the four FIFO stacks of capture units. The bit description of CAPFIFO is given in Figure 2–28. The 8 least significant bits of CAPFIFO have a one-to-one correspondence with the 8 most significant bits of CAPFIFO. A one written to CAPFIFO[x], x = 0, 1, ... or 7, clears bit CAPFIFO[x+8]. CAPFIFO[x], for x = 0, 1, ... 7, are write-only and CAPFIFO[y], for y = 8, 9, ... 15, are read-only.

Figure 2–28. Capture FIFO Status Register (CAPFIFO) — Address 7422h

15–14	13–12	11–10	9–8
CAP4FIFO	CAP3FIFO	CAP2FIFO	CAP1FIFO
R–0	R–0	R–0	R–0

7	6	5	4	3	2	1	0
CAPFIFO15	CAPFIFO14	CAPFIFO13	CAPFIFO12	CAPFIFO11	CAPFIFO10	CAPFIFO9	CAPFIFO8
W–0	W–0	W–0	W–0	W–0	W–0	W–0	W–0

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

Bits 15–14 **CAP4FIFO**. CAP4FIFO Status. Read only.

00 = Empty
01 = Has one entry.
10 = Has two entries.
11 = Had two entries and captured another one; first entry has been lost.

Bits 13–12 **CAP3FIFO**. CAP3FIFO Status. Read only.

00 = Empty
01 = Has one entry.
10 = Has two entries.
11 = Had two entries and captured another one; first entry has been lost.

Bits 11–10 **CAP2FIFO**. CAP2FIFO Status. Read only.

00 = Empty
01 = Has one entry.
10 = Has two entries.
11 = Had two entries and captured another one; first entry has been lost.

Bits 9–8 **CAP1FIFO**. CAP1FIFO Status. Read only.

00 = Empty
01 = Has one entry.
10 = Has two entries.
11 = Had two entries and captured another one; first entry has been lost.

Bit 7	**CAPFIFO15**. CAP1FIFO bit 15 Clear. Write only. 0 = Do nothing 1 = Clear bit 15
Bit 6	**CAPFIFO14**. CAP1FIFO bit 14 Clear. Write only. 0 = Do nothing 1 = Clear bit 14
Bit 5	**CAPFIFO13**. CAP1FIFO bit 13 Clear. Write only. 0 = Do nothing 1 = Clear bit 13
Bit 4	**CAPFIFO12**. CAP1FIFO bit 12 Clear. Write only. 0 = Do nothing 1 = Clear bit 12
Bit 3	**CAPFIFO11**. CAP1FIFO bit 11 Clear. Write only. 0 = Do nothing 1 = Clear bit 11
Bit 2	**CAPFIFO10**. CAP1FIFO bit 10 Clear. Write only. 0 = Do nothing 1 = Clear bit 10
Bit 1	**CAPFIFO9**. CAP1FIFO bit 9 Clear. Write only. 0 = Do nothing 1 = Clear bit 9
Bit 0	**CAPFIFO8**. CAP1FIFO bit 8 Clear. Write only. 0 = Do nothing 1 = Clear bit 8

2.8.4 Capture Unit FIFO Stacks

FIFO Stack Associated with Each Capture Unit

Each capture unit has a dedicated 2-level-deep FIFO stack. The top-level register of any of the FIFO stacks is a read-only register that always contains the older counter value captured by the corresponding capture unit. Therefore, a read access to the FIFO stack of a capture unit always reads out the older counter value captured in the stack. When the older counter value in the top register of the FIFO stack is read, the newer counter value in the bottom register of the stack, if any, will be pushed into the top register.

First Capture

The counter value of selected GP timer captured by a capture unit when a specified transition happens on its input pin will be written into the top register of the stack if the stack is empty. At the same time, the corresponding status bits are set to (01). The status bits are reset to (00) if a read access is made to the FIFO stack before another capture is made.

Second Capture

If another capture occurs before the previously captured counter value is read, the newly captured counter value goes to the bottom register. In the mean time, the corresponding status bits are set to (10). When the FIFO stack is read before another capture happens, the older counter value in the top register will be read out, the newer counter value in the bottom register will be pushed up into the top register and the corresponding status bits will be set to (01).

Third Capture

If a capture happens when there are already two counter values captured in the FIFO stack, the oldest counter value in the top register of the stack will be pushed out and lost, the counter value in the bottom register of the stack will be pushed up into the top register, the newly captured counter value will be written into the bottom register and the status bits are set to (11) to indicate one or more older captured counter values have been lost.

Example 2–8 shows an initialization routine for capture units.

Example 2–8. Initialization Routine for Capture Units

```
;*********************************************************
; Initialize counter registers                          *
;*********************************************************
             SPLK   #00000H,T1CNT; GP Timer 1 counter
             SPLK   #00000H,T2CNT; GP Timer 2 counter
             SPLK   #00000H,T3CNT; GP Timer 3 counter
;*********************************************************
; Initialize period registers                                  *
;*********************************************************
             SPLK   #00011H,T1PER; GP Timer 1 period
             SPLK   #07fffH,T2PER; GP Timer 2 period
             SPLK   #00011H,T3PER; GP Timer 3 period
;*********************************************************
; Initialize compare registers                          *
;*********************************************************
             SPLK   #00004H,T1CMP; GP Timer 1 Comp Value
             SPLK   #00006H,T2CMP; GP Timer 2 Comp Value
             SPLK   #00008H,T3CMP; GP Timer 3 Comp Value
;*********************************************************
; Configure GPTCON                                             *
;*********************************************************
             SPLK   #0000000001010101b,GPTCON
                                 ; Set GP Timer control
* bits 12-11        00:    No GP Timer 3 event starts ADC
* bits 10-9         00:    No GP Timer 2 event starts ADC
* bits 8-7          00:    No GP Timer 1 event starts ADC
* bit 6             1:     Enable GP Timer Compare outputs
* bits 5-4          01:    GP Timer 3 comp output active low
* bits 3-2          01:    GP Timer 2 comp output active low
* bits 1-0          01:    GP Timer 1 comp output active low
;*********************************************************
; Clear all EV interrupts before operation starts       *
;*********************************************************
             SPLK   #0ffffh,IFRA ; Clear all Group A interrupt flags
             SPLK   #0ffffh,IFRB ; Clear all Group B interrupt flags
             SPLK   #0ffffh,IFRC ; Clear all Group C interrupt flags
;*********************************************************
; Configure T2CON and start GP Timer 2                  *
;*********************************************************
             SPLK   #1001000001000010b,T2CON
                                 ; Set GP Timer 2 control
* bit 15            1:     FREE = 1
* bit 14            0:     SOFT = 0
* bits 13-11        010:   continuous-up count mode
* bits 10-8         000:   Prescaler = /1
* bit 7             0:     Use own Timer ENABLE
* bit 6             1:     Timer (counting operation) enabled
* bits 5-4          00:    Select internal CLK
```

Example 2–8. Initialization Routine for Capture Units (Continued)

```
* bits 3-2          00:    Load GP Timer comp register on underflow
* bit 1             1:     GP Timer compare enabled
* bit 0             0:     Use own PR
             NOP
             NOP
             NOP
             NOP
;*****************************************************
; Configure CAPCON and enable capture operation             *
;*****************************************************
             SPLK   #1011110001010101b,CAPCON
                                 ; Capture control
* bit 15            1:     No action
* bit 14-13         01:    Enable Capture Us 1&2 and disable QEP
* bit 12            1:     Enable Capture U 3
* bit 11            1:     Enable Capture U 4
* bit 10            1:     GP Timer 3 is time base for Cap Us 3&4
* bit 9             0:     GP Timer 2 is time base for Cap Us 1&2
* bit 8             0:     No Capture U 4 event starts ADC
* bits 7-6          01:    Capture U 1 detects rising edge
* bits 5-4          01:    Capture U 2 detects rising edge
* bits 3-2          01:    Capture U 3 detects rising edge
* bits 1-0          01:    Capture U 4 detects rising edge
;*****************************************************
; Configure T3CON                                           *
;*****************************************************
             SPLK   #1001000011000010b,T3CON
                                 ; Set GP Timer 3 control
* bit 15            1:     FREE = 1
* bit 14            0:     SOFT = 0
* bits 13-12        010:   continuous-up count mode
* bits 10-8         000:   Prescaler = /1
* bit 7             1:     Use Timer ENABLE of GP Timer 1
* bit 6             1:     Timer (counting operation) enabled
* bits 5-4          00:    Select internal CLK
* bits 3-2          00:    Load GP Timer comp register on underflow
* bit 1             1:     GP Timer compare enabled
* bit 0             0:     Use own PR
;*****************************************************
; Configure T1CON and start GP Timers 1&3                   *
;*****************************************************
             SPLK   #1001000001000010b,T1CON
                                 ; Set GP Timer 1 control
                                 ; Start GP Timers 1&2
* bit 15            1:     FREE = 1
* bit 14            0:     SOFT = 0
* bits 13-12        010:   continuous-up count mode
* bits 10-8         000:   Prescaler = /1
* bit 7             0:     reserved
```

Example 2–8. Initialization Routine for Capture Units (Continued)

```
* bit 6             1:     Timer (counting operation) enabled
* bits 5-4          00:    Select internal CLK
* bits 3-2          00:    Load GP Timer comp register on underflow
* bit 1             1:     GP Timer compare enabled
* bit 0             0:     reserved
;***********************************************************
; Read captured values and calculate the difference        *
;***********************************************************
             LAR    AR2,#01eh            ; Loop 16 times
             MAR    *,AR1                ; ARP<=AR1 point to result log
LOOP
             LACC   CAPFIFO              ; Read Capture FIFO status
             AND    #0000001000000000b
                                         ; Got >=2 entries in FIFO 1
             BZ     LOOP                 ; No, bypass
             LACC   FIFO1                ; Read Capture FIFO 1 1st entry
             SACL   *+                   ; Save
             LACC   FIFO1                ; 1st entry-2nd entry
             SACL   *-                   ; Save
             SUB    *+                   ; Calculate diff
             MAR    *+,AR3               ; Adjust pointer and point to diff log
             SACL   *+,AR2               ; Save diff and point to loop control
DLOOP        B      DLOOP
```

2.8.5 Capture Interrupt

When a capture is made by a capture unit, the corresponding interrupt flag is set, and, if unmasked and no other unmasked interrupts in the same group of higher priority are pending, an interrupt request will be generated to the DSP core. The captured counter value can thus be read from an interrupt service routine if interrupt is used. If interrupt is not desired, either the interrupt flag or the status bits can be polled to know a capture has been made and thus the captured counter value can be read.

2.9 Quadrature Encoder Pulse (QEP) Circuit

The Event Manager module has a Quadrature Encoder Pulse (QEP) circuit. The QEP circuit, when enabled, decodes and counts the quadrature encoded input pulses on pins CAP1/QEP1 and CAP2/QEP2. The QEP circuit can be used to interface with an optical encoder to get position and speed information of a rotating machine.

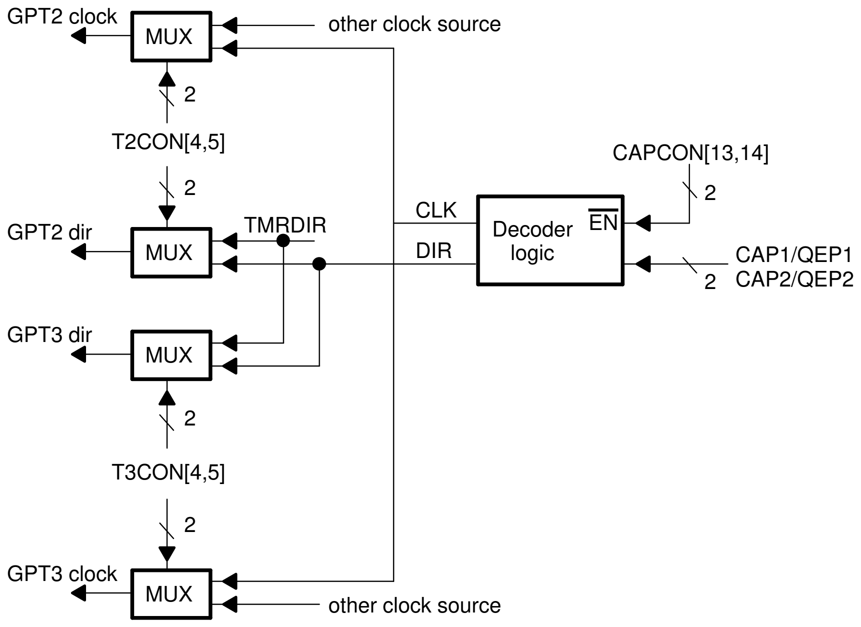
2.9.1 QEP Pins

The two QEP input pins are shared between Capture Units 1 and 2, and the QEP circuit. Proper configuration of CAPCON bits are needed to enable the QEP circuit and disable Capture Units 1 and 2, thus assigning the two associated input pins for use by QEP circuit.

2.9.2 QEP Circuit Time Base

The time base for the QEP circuit can be provided by GP Timer 2, 3, or 2 and 3 together as a 32-bit timer. The selection is made by configuration of T2CON or T3CON bits. The selected GP timer or 32-bit timer must be set in directional-up/down count mode with QEP circuit as the clock source. Figure 2–29 shows the block diagram of the QEP circuit.

Figure 2–29. Quadrature Encoder Pulse (QEP) Circuit Block Diagram



2.9.3 QEP Decoding

Quadrature encoded pulses are two sequences of pulses with variable frequency and a fixed phase shift of a quarter of a period (90 degrees). When generated by an optical encoder on a motor shaft, the direction of rotation of the motor can be determined by detecting which of the two sequences is the leading sequence, and the angular position and speed can be determined by the pulse count and pulse frequency.

QEP Circuit

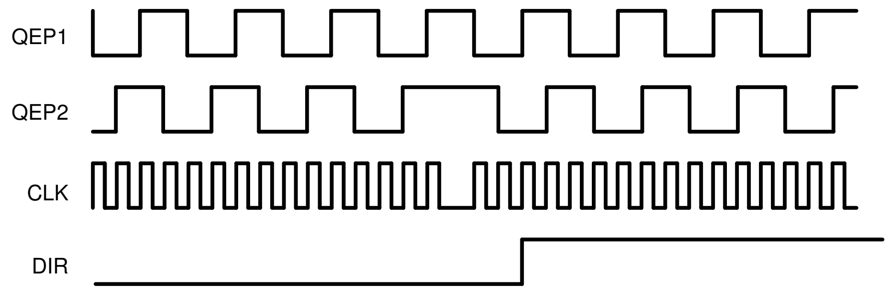
The direction detection logic of the QEP circuit in the EV module determines which one of the sequences is the leading sequence. It then generates a direction signal as the direction input to the selected timer. The selected timer counts up if CAP1/QEP1 input is the leading sequence, and counts down if CAP2/QEP2 is the leading sequence.

Both edges of the pulses of the two quadrature encoded inputs are counted by the QEP circuit. Therefore, the frequency of the generated clock to the GP timer is four times that of each input sequence. This generated clock is connected to the clock input of the selected GP timer or 32-bit timer.

Quadrature Encoded Pulse Decoding Example

Figure 2–30 shows an example of quadrature encoded pulses and the determined up and down counting direction and clock.

Figure 2–30. Quadrature Encoded Pulses and Decoded Timer Clock and Direction



2.9.4 QEP Counting

GP Time Used with QEP Circuit

The selected GP timer always starts counting from its current value in the counter. A desired value can be loaded to the selected GP timer counter prior to the enabling of the QEP operation. When the QEP circuit is selected as the clock source, the selected timer will ignore the TMRDIR and TMRCLK input pins.

GP Timer Counting Mode in QEP Operation

It is important to note that the directional-up/down counting mode of selected GP timer with QEP circuit as clock is different from the normal directional-up/down counting mode. When the timer selected for QEP operation counts up to the period value, the GP timer will not stop, but instead it will continue counting up until the determined counting direction changes. If the initial value of the GP timer counter is greater than that of the period register, the timer will count up to FFFFh (or FFFF FFFFh, where a 32-bit timer is used) and rollover to 0 if the determined counting direction is upward. When the timer counts down to 0, the GP timer will rollover to FFFFh (or FFFF FFFFh, where a 32-bit timer is used) if the determined counting direction is down.

GP Timer Interrupt and Associated Compare Outputs in QEP Operation

Period, underflow, overflow and compare interrupt flags for the GP timer with QEP circuit as clock are generated on respective matches, though no transition will happen on the compare output of the selected GP Timer or any other compare unit that uses this timer as its time base. An interrupt request can be generated to the CPU by an interrupt flag if the interrupt is unmasked and no other unmasked interrupts in the same group of higher priority are pending.

2.9.5 Register Setup for QEP Circuit

To start the operation of QEP circuit:

1) Load the selected GP timer counter, period, and compare registers with desired values, if necessary.
2) Configure T2CON or T3CON to set GP Timer 2, 3, or 2 and 3 in directional-up/down or 32-bit mode with QEP circuits as clock source and enable selected timer.
3) Configure CAPCON to enable the QEP circuit.

Example 2–9 shows an initialization routine for QEP operation.

Example 2–9. Register Setup for QEP Circuit

```
;*********************************************************************
; The following section of codes initializes QEP operation          *
;*********************************************************************
             LDPK   #232                      ; DP => EV Registers
;***********************************************************************
; Configure GPTCON
;***********************************************************************
             SPLK   #1110001011110000b, CAPCON ; Set GP Timer control
;                    ||||||||||||||||
;                   FEDCBA9876543210
;
* bits 0-1          00:    No detection for Capture 4
* bits 2-3          00:    No detection for Capture 3
* bits 4-5          11:    Capture 2 detects both edges
* bits 6-7          11:    Capture 1 detects both edges
* bit 8             0:     No Capture 4 event starts ADC
* bit 9             1:     GP Timer 3 is time base for Capture 1 & 2
* bit 10            0:     GP Timer 2 is time base for Capture 3 & 4
* bit 11            0:     Disable Capture 4
* bit 12            0:     Disable Capture 3
* bits 13-14        11:    Enable QEP
* bit 15            1:     No action
;**********************************************************************
; Configure counter register T3CNT                                   *
;**********************************************************************
             SPLK   #0000h, T3CNT
;**********************************************************************
; Configure period register T3PER                                    *
;**********************************************************************
             SPLK   #00FFh, T3PER
;***********************************************************************
; Configure T3CON and start GP Timer 3                                 *
;***********************************************************************
             SPLK   #1001100001110000b, T1CON  ; Set GP Timer 3 control
;                    ||||||||||||||||
;                   FEDCBA9876543210
;
* bit 0             0:     Use own PR
* bit 1             0:     GP Timer compare disabled
* bits 2-3          00:    Load GP Timer comp register on underflow
* bits 4-5          11:    Select QEP
* bit 6             1:     Timer (counting operation) enabled
* bit 7             0:     Use own Timer ENABLE
* bits 8-10         000:   Prescaler = /1(During QEP,prescaler is always 1)
* bits 11-13        011:   Directional up/down mode for QEP
* bit 14            0:     SOFT = 0
* bit 15            1:     FREE = 1
```

2.10 Event Manager (EV) Interrupts

2.10.1 Organization of TMS320C24x Interrupts

'C2xx Core Interrupt Organization

The 'C2xx core (the CPU) has six maskable (INT1–INT6) and one nonmaskable (NMI) external interrupts. Interrupt NMI has the highest priority and INT1 has the next highest priority. The priority decreases from INT1 to INT6, with INT6 having the lowest priority. Each interrupt corresponds to a bit in the core interrupt flag register (IFR) and each maskable interrupt corresponds to a bit in the core interrupt mask register (IMR). A maskable interrupt is masked (will not generate an interrupt to the core) when the corresponding bit in IMR is 0. In addition, the core has a global interrupt mask bit (INTM) in status register ST0. When INTM is set to 1, all maskable interrupts are masked.

'C2xx Core Interrupt Handling

When a transition from high to low occurs on an interrupt input to the core, the corresponding interrupt flag bit in IFR is set to 1. An interrupt to the core is generated by this flag if it is unmasked, global interrupts are allowed (INTM = 0), and no other unmasked interrupts of higher priority are pending (that is, no other unmasked interrupt flags of higher priority are set). The flag is cleared by hardware once the interrupt request is taken by the core. An interrupt flag can also be cleared by user software writing a 1 to the bit.

2.10.2 EV Interrupt Request and Service

Interrupt Groups

Event Manager interrupt events are organized into 3 groups, EV interrupt group A, B, and C. EV interrupt group A generates interrupt requests to the core on INT2. EV interrupt groups B and C generate interrupt requests to the core on INT3 and INT4, respectively. Table 2–10 shows all EV interrupts and their priority and grouping. There is an interrupt flag register and an interrupt mask register for each EV interrupt group: EVIFRA, EVIFRB, and EVIFRC, and EVIMRA, EVIMRB, and EVIMRC. A flag in EVIFRx (x = A, B, or C) is masked (will not generate an interrupt request to the core) if the corresponding bit in EVIMRx is 0.

There is an 8-bit interrupt vector register (EVIVRx, x = A, B, or C) associated with each EV interrupt group. The corresponding interrupt vector register can be read from an interrupt service routine (ISR), when an interrupt request generated by the interrupt group is accepted by the core. The value (vector) in the interrupt vector register identifies which pending interrupt in the group has the highest priority.

Table 2–10. Event Manager Interrupts

Group	Interrupt	Priority within group	Vector (ID)	Description/Source
A	PDPINT	1 (highest)	0020h	Power Drive Protection Interrupt
	CMP1INT	2	0021h	Full Compare Unit 1 compare interrupt
	CMP2INT	3	0022h	Full Compare Unit 2 compare interrupt
	CMP3INT	4	0023h	Full Compare Unit 3 compare interrupt
	SCMP1INT	5	0024h	Simple Compare Unit 1 compare interrupt
	SCMP2INT	6	0025h	Simple Compare Unit 2 compare interrupt
	SCMP3INT	7	0026h	Simple Compare Unit 3 compare interrupt
	T1PINT	8	0027h	GP Timer 1 period interrupt
	T1CINT	9	0028h	GP Timer 1 compare interrupt
	T1UFINT	10	0029h	GP Timer 1 underflow interrupt
	T1OFINT	11 (lowest)	002Ah	GP Timer 1 overflow interrupt
B	T2PINT	1 (highest)	002Bh	GP Timer 2 period interrupt
	T2CINT	2	002Ch	GP Timer 2 compare interrupt
	T2UFINT	3	002Dh	GP Timer 2 underflow interrupt
	T2OFINT	4	002Eh	GP Timer 2 overflow interrupt
	T3PINT	5	002Fh	GP Timer 3 period interrupt
	T3CINT	6	0030h	GP Timer 3 compare interrupt
	T3UFINT	7	0031h	GP Timer 3 underflow interrupt
	T3OFINT	8 (lowest)	0032h	GP Timer 3 overflow interrupt
C	CAP1INT	1 (highest)	0033h	Capture Unit 1 interrupt
	CAP2INT	2	0034h	Capture Unit 2 interrupt
	CAP3INT	3	0035h	Capture Unit 3 interrupt
	CAP4INT	4 (lowest)	0036h	Capture Unit 4 interrupt

Interrupt Generation

When an interrupt event occurs in the EV module, the corresponding interrupt flag in one of the EV interrupt flag registers is set to 1. An interrupt request is generated on the corresponding INTx, if the flag is locally unmasked (the corresponding bit in EVIMRx is set to 1) and no other unmasked interrupts in the same group of higher priority are pending.

Interrupt Vector

The interrupt vector (ID) corresponding to the interrupt flag that has the highest priority among the flags that are set is loaded into the accumulator when the interrupt vector of an EV interrupt group is read after an interrupt request to the core has been generated by the group. The flag is cleared when its interrupt vector has been read. An interrupt flag can also be cleared using software to directly write a 1 to the bit.

A 0 is returned when the EV interrupt vector register of a group is read and no interrupt flags in the group are set. This is done to avoid strayed interrupts being identified as EV interrupt.

Interrupt Handling

After an EV interrupt request is taken, the EVIVRx must be read into the accumulator and shifted to left by one or more bits. Next, an offset address (start of an interrupt entrance table) is added to the accumulator. A BACC instruction is used to branch to the right entry in the table. Another branch from the table branches to the ISR for a specific source. This process causes a typical interrupt latency of 20 CPU cycles (25 if minimum context saving is required) from the time an interrupt is generated to when the first instruction in the ISR for the specific source is reached. This latency can be reduced to a minimum of 8 CPU cycles if only one interrupt is allowed in an EV interrupt group. If memory space is not a concern, the latency can be reduced to 16 CPU cycles without the requirement of allowing only one interrupt per EV interrupt group.

2.10.3 EV Interrupt Flag Registers

Addresses of EV interrupt registers are shown in Table 2–5 on page 2-10. The registers are all treated as 16-bit memory mapped registers. The unused bits all return zero when read by software. Writing to unused bits has no effect. Since EVIFRx are readable registers, occurrence of an interrupt event can be monitored by software polling the appropriate bit in EVIFRx when the interrupt is masked.

EV Interrupt Flag Register A (EVIFRA)

Figure 2–31. EV Interrupt Flag Register A (EVIFRA) — Address 742Fh

15–11	10	9	8
Reserved	T1OFINT	T1UFINT	T1CINT
	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
T1PINT	SCMP3INT	SCMP2INT	SCMP1INT	CMP3INT	CMP2INT	CMP1INT	PDPINT
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

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

Bits 15–11 **Reserved**. Reads return zero and writes have no effect.

Bit 10 **T1OFINT**. GP Timer 1 overflow interrupt.

Read: 0 = Flag is reset.
1 = Flag is set.
Write: 0 = No effect
1 = Reset flag

Bit 9 **T1UFINT**. GP Timer 1 underflow interrupt.

Read: 0 = Flag is reset.
1 = Flag is set.
Write: 0 = No effect
1 = Reset flag

Bit 8 **T1CINT**. GP Timer 1 compare interrupt.

Read: 0 = Flag is reset.
1 = Flag is set.
Write: 0 = No effect
1 = Reset flag

Bit 7 **T1PINT**. GP Timer 1 period interrupt.

Read: 0 = Flag is reset.
1 = Flag is set.
Write: 0 = No effect
1 = Reset flag

Bit 6 **SCMP3INT**. Simple compare 3 interrupt.

Read: 0 = Flag is reset.
1 = Flag is set.
Write: 0 = No effect
1 = Reset flag

Bit 5 **SCMP2INT**. Simple compare 2 interrupt.

Read:	0	= Flag is reset.
	1	= Flag is set.
Write:	0	= No effect
	1	= Reset flag

Bit 4 **SCMP1INT**. Simple compare 1 interrupt.

Read:	0	= Flag is reset.
	1	= Flag is set.
Write:	0	= No effect
	1	= Reset flag

Bit 3 **CMP3INT**. Full compare 3 interrupt.

Read:	0	= Flag is reset.
	1	= Flag is set.
Write:	0	= No effect
	1	= Reset flag

Bit 2 **CMP2INT**. Full compare 2 interrupt.

Read:	0	= Flag is reset.
	1	= Flag is set.
Write:	0	= No effect
	1	= Reset flag

Bit 1 **CMP1INT**. Full compare 1 interrupt.

Read:	0	= Flag is reset.
	1	= Flag is set.
Write:	0	= No effect
	1	= Reset flag

Bit 0 **PDPINT**. Power drive protection interrupt.

Read:	0	= Flag is reset.
	1	= Flag is set.
Write:	0	= No effect
	1	= Reset flag

EV Interrupt Flag Register B (EVIFRB)

Figure 2–32. EV Interrupt Flag Register B (EVIFRB) — Address 7430h

15–8	7	6	5	4	3	2	1	0
Reserved	T3OFINT	T3UFINT	T3CINT	T3PINT	T2OFINT	T2UFINT	T2CINT	T2PINT
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

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

Bits 15–8 **Reserved**. Reads return zero and writes have no effect.

Bit 7 **T3OFINT**. GP Timer 3 overflow interrupt.

Read: 0 = Flag is reset.
1 = Flag is set.
Write: 0 = No effect
1 = Reset flag

Bit 6 **T3UFINT**. GP Timer 3 underflow interrupt.

Read: 0 = Flag is reset.
1 = Flag is set.
Write: 0 = No effect
1 = Reset flag

Bit 5 **T3CINT**. GP Timer 3 compare interrupt.

Read: 0 = Flag is reset.
1 = Flag is set.
Write: 0 = No effect
1 = Reset flag

Bit 4 **T3PINT**. GP Timer 3 period interrupt.

Read: 0 = Flag is reset.
1 = Flag is set.
Write: 0 = No effect
1 = Reset flag

Bit 3 **T2OFINT**. GP Timer 2 overflow interrupt.

Read: 0 = Flag is reset.
1 = Flag is set.
Write: 0 = No effect
1 = Reset flag

Bit 2 **T2UFINT**. GP Timer 2 underflow interrupt.

Read:	0 = Flag is reset.
	1 = Flag is set.
Write:	0 = No effect
	1 = Reset flag

Bit 1 **T2CINT**. GP Timer 2 compare interrupt.

Read:	0 = Flag is reset.
	1 = Flag is set.
Write:	0 = No effect
	1 = Reset flag

Bit 0 **T2PINT**. GP Timer 2 period interrupt.

Read:	0 = Flag is reset.
	1 = Flag is set.
Write:	0 = No effect
	1 = Reset flag

EV Interrupt Flag Register C (EVIFRC)

Figure 2–33. EV Interrupt Flag Register C (EVIFRC) — Address 7431h

15–4	3	2	1	0
Reserved	CAP4INT	CAP3INT	CAP2INT	CAP1INT
	RW–0	RW–0	RW–0	RW–0

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

Bits 15–4 **Reserved**. Reads return zero and writes have no effect.

Bit 3 **CAP4INT**. Capture 4 interrupt.

Read:	0 = Flag is reset.
	1 = Flag is set.
Write:	0 = No effect
	1 = Reset flag

Bit 2 **CAP3INT**. Capture 3 interrupt.

Read:	0 = Flag is reset.
	1 = Flag is set.
Write:	0 = No effect
	1 = Reset flag

Bit 1 **CAP2INT**. Capture 2 interrupt.

Read:	0	= Flag is reset.
	1	= Flag is set.
Write:	0	= No effect
	1	= Reset flag

Bit 0 **CAP1INT**. Capture 1 interrupt.

Read:	0	= Flag is reset.
	1	= Flag is set.
Write:	0	= No effect
	1	= Reset flag

EV Interrupt Mask Register A (EVIMRA)

Figure 2–34. EV Interrupt Mask Register A (EVIMRA) — Address 742Ch

15–11	10	9	8
Reserved	T1OFINT ENABLE	T1UFINT ENABLE	T1CINT ENABLE
	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
T1PINT ENABLE	SCMP3INT ENABLE	SCMP2INT ENABLE	SCMP1INT ENABLE	CMP3INT ENABLE	CMP2INT ENABLE	CMP1INT ENABLE	PDPINT ENABLE
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

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

Bits 15–11 **Reserved**. Reads return zero and writes have no effect.

Bit 10 **T1OFINT ENABLE**

0 = Disable
1 = Enable

Bit 9 **T1UFINT ENABLE**

0 = Disable
1 = Enable

Bit 8 **T1CINT ENABLE**

0 = Disable
1 = Enable

Bit 7	**T1PINT ENABLE**
	0 = Disable
	1 = Enable
Bit 6	**SCMP3INT ENABLE**
	0 = Disable
	1 = Enable
Bit 5	**SCMP2INT ENABLE**
	0 = Disable
	1 = Enable
Bit 4	**SCMP1INT ENABLE**
	0 = Disable
	1 = Enable
Bit 3	**CMP3INT ENABLE**
	0 = Disable
	1 = Enable
Bit 2	**CMP2INT ENABLE**
	0 = Disable
	1 = Enable
Bit 1	**CMP1INT ENABLE**
	0 = Disable
	1 = Enable
Bit 0	**PDPINT ENABLE**
	0 = Disable
	1 = Enable

EV Interrupt Mask Register B (EVIMRB)

Figure 2–35. EV Interrupt Mask Register B — Address 742Dh

15–8	7	6	5	4	3	2	1	0
Reserved	T3OFINT ENABLE	T3UFINT ENABLE	T3CINT ENABLE	T3PINT ENABLE	T2OFINT ENABLE	T2UFINT ENABLE	T2CINT ENABLE	T2PINT ENABLE
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

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

Bits 15–8 **Reserved**. Reads return zero and writes have no effect.

Bit 7 **T3OFINT ENABLE**

0 = Disable
1 = Enable

Bit 6 **T3UFINT ENABLE**

0 = Disable
1 = Enable

Bit 5 **T3CINT ENABLE**

0 = Disable
1 = Enable

Bit 4 **T3PINT ENABLE**

0 = Disable
1 = Enable

Bit 3 **T2OFINT ENABLE**

0 = Disable
1 = Enable

Bit 2 **T2UFINT ENABLE**

0 = Disable
1 = Enable

Bit 1 **T2CINT ENABLE**

0 = Disable
1 = Enable

Bit 0 **T2PINT ENABLE**

0 = Disable
1 = Enable

EV Interrupt Mask Register C (EVIMRC)

Figure 2–36. EV Interrupt Mask Register C — Address 742Eh

15–4	3	2	1	0
Reserved	CAP4INT ENABLE	CAP3INT ENABLE	CAP2INT ENABLE	CAP1INT ENABLE
	RW–0	RW–0	RW–0	RW–0

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

Bits 15–4 **Reserved**. Reads return zero and writes have no effect.

Bit 3 **CAP4INT ENABLE**

0 = Disable
1 = Enable

Bit 2 **CAP3INT ENABLE**

0 = Disable
1 = Enable

Bit 1 **CAP2INT ENABLE**

0 = Disable
1 = Enable

Bit 0 **CAP1INT ENABLE**

0 = Disable
1 = Enable

EV Interrupt Vector Register A (EVIVRA)

Figure 2–37. EV Interrupt Vector Register A (EVIVRA) — Address 7432h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0
R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0

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

Bits 15–6 **Reserved.** Reads return zero and writes have no effect.

Bits 5–0 **D5–D0**. Vector (ID) of the interrupt flag that has the highest priority among the set interrupt flags of EVIFRA; 0 if no interrupt flag is set in EVIFRA.

EV Interrupt Vector Register B (EVIVRB)

Figure 2–38. EV Interrupt Vector Register B (EVIVRB) — Address 7433h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0
R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0

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

Bits 15–6 **Reserved**. Reads return zero and writes have no effect.

Bits 5–0 **D5–D0**. Vector (ID) of the interrupt flag that has the highest priority among the set interrupt flags of EVIFRB; 0 if no interrupt flag is set in EVIFRB.

EV Interrupt Vector Register C (EVIVRC)

Figure 2–39. EV Interrupt Vector Register C (EVIVRC) — Address 7434h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0
R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0

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

Bits 15–6 **Reserved**. Reads return zero and writes have no effect.

Bits 5–0 **D5–D0**. Vector (ID) of the interrupt flag that has the highest priority among the set interrupt flags of EVIFRC; 0 if no interrupt flag is set in EVIFRC.

Chapter 3

Dual 10-Bit Analog to Digital Converter (ADC) Module

This chapter contains a general description of the Dual 10-Bit Analog to Digital Converter (ADC) module.

Topic **Page**

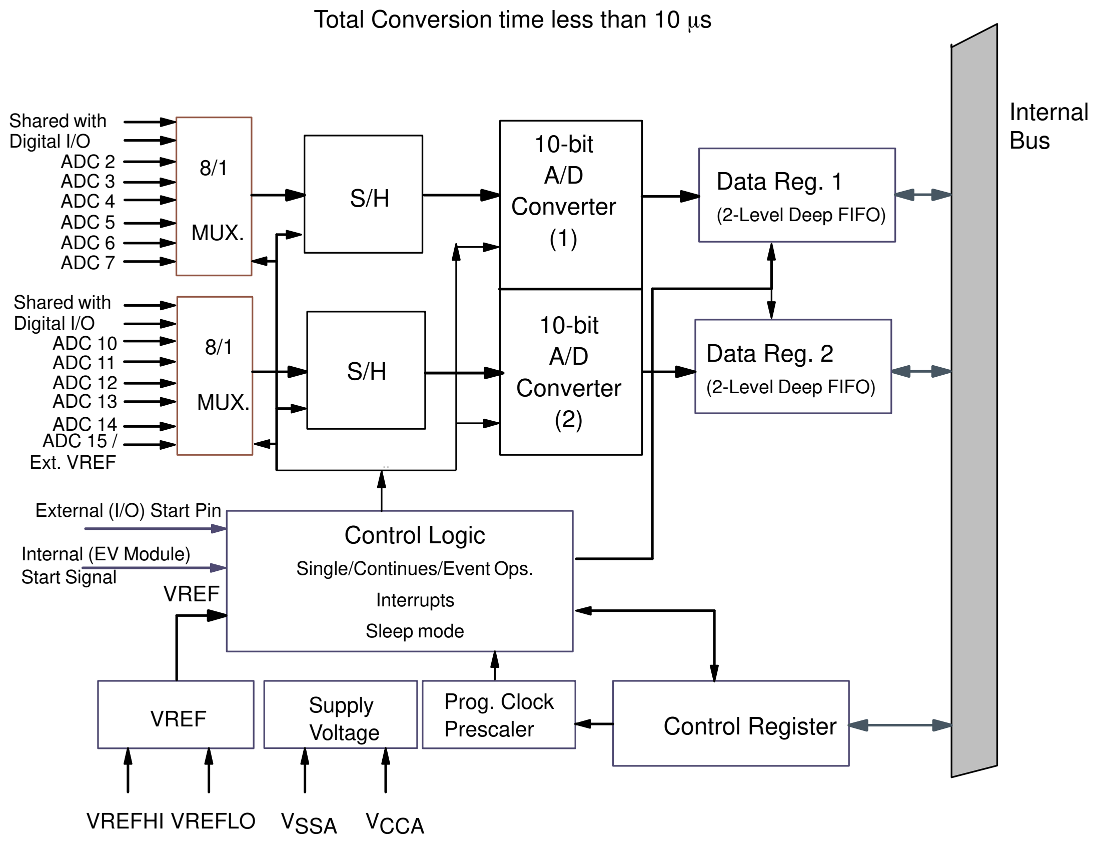
3.1 Dual 10-Bit ADC Overview

The analog-to-digital converter (ADC) is a 10 bit string/capacitor converter with internal sample-and-hold circuitry. The ADC module consists of two 10-bit analog to digital converters with 2 built in sample and hold circuits. A total of 16 analog input channels are available on the ’C24x. Eight analog inputs are provided for each ADC unit via an 8 to 1 analog multiplexer. Maximum total conversion time for each ADC unit is 6.6 µs. The reference voltage of the ADC module has to be supplied from an external source. The upper and lower references can be set to any voltage less than or equal to 5 VDC by connecting VREFHI and VREFLO to the appropriate references. V_{CCA} and V_{SSA} pins have to be connected to 5 VDC and analog ground respectively.

The ADC module, shown in Figure 3–1, consists of the following:

- 8 analog inputs for each ADC module, giving a total of 16 analog inputs (ADC0–ADC15).
- Simultaneous measurement of two analog inputs using two ADC units.
- Single conversion, continuous conversion.
- Conversion can be started by software, internal event, and/or external event.
- VREFHI and VREFLO (high- and low-voltage) reference inputs.
- Analog to digital conversion block.
- Two level deep digital result registers that contain the digital values of completed conversions.
- Two programmable ADC module control registers.
- Programmable prescaler select.
- Interrupt or polled operation.

Figure 3–1. Analog to Digital Converter (ADC) Module Block Diagram



3.2 ADC Operation

The digital result of the conversion process for the 10-bit ADC is approximated by the following equation:

$$\textit{Digital Result} = 1023 \times \frac{\textit{Input Voltage}}{\textit{Reference Voltage}}$$

3.2.1 ADC Module Pin Description

The ADC module provides 20 pins that can interface to external circuitry. Sixteen of these pins, ADC0–ADC15, are for the analog inputs. The other two pins, VREFHI and VREFLO, are the analog reference-voltage pins.

The analog supply pins, V_{CCA} and V_{SSA}, are separate from any digital voltage supply pins. Analog power lines connected to V_{CCA} and V_{SSA} should be as short as possible with the two lines properly decoupled. All other standard noise reduction techniques should be used to ensure accurate conversion.

Analog voltage input pins ADC0 through ADC7 belong to the first ADC module and ADC8 through ADC15 belong to the second ADC module. Analog inputs ADC0 and ADC1 of the first module and analog inputs ADC8 and ADC9 of the second module are multiplexed with digital I/O. By properly programming the system module, these four analog input pins (ADC0, ADC1, ADC8, and ADC9) can be used as digital I/O. The accuracy of these four pins are lower than that of the dedicated analog input pins (ADC2 – ADC7, and ADC10 – ADC15).

3.2.2 ADC Module Operational Modes

Functions of the ADC module include:

- ❑ Two input channels (one for each ADC unit) can be sampled and converted simultaneously.
- ❑ Each ADC unit can perform single or continuous S/H and conversion operations.
- ❑ Two 2-Level deep FIFO result registers for ADC unit 1 and 2.
- ❑ ADC module (both A/D converters) can start operation by software instruction, or external signal transition on a device pin, or by the Event Manager events on each of the GP Timer/Compare outputs and the Capture 4.

- ❑ Certain bits of the ADC control registers are double-buffered with shadow registers and can be written to at any time without effecting the ongoing conversion process. The newly written bit values first go to a shadow register instead of the active register. This new bit configuration is then automatically loaded from the shadow register to the active register only after the completion of the present conversion process. The next conversion process will then be determined by the new bit configuration.
- ❑ At the end of each conversion, an interrupt flag is set and an interrupt is generated if it is unmasked/enabled.
- ❑ If a third conversion is completed without reading the FIFO, the data from the first conversion will be lost.

3.2.3 Analog Signal Sampling/Conversion

The individual ADC module performs input sampling in one ADC prescaled clock cycle and conversion in five ADC prescaled clock cycles for a total sample/conversion in six ADC clock cycles. The architecture of the ADC module requires the sample/conversion time to be 6 μs or greater to ensure an accurate conversion. This relationship between the number of ADC clock cycles required (six) and the minimum 6 μs must be met at all system clock (SYSCLK) frequencies to ensure accurate conversion. Since the system clock may operate at frequencies which would violate this relationship, a prescaler has been provided with the ADC modules, which allows the module to maintain optimal performance as the DSP clock varies between applications. Select the ADC prescaler value such that the total ADC sample/conversion time is greater than or equal to 6 μs. Therefore, the prescaler value should satisfy the following formula:

$$\textit{SYSCLK Period} \times \textit{Prescaler Value} \times 6 \geq 6\,\mu s$$

Table 3–1. Sample Clock Frequencies and Appropriate Prescaler Values

ADCTRL2 Bits			Prescale Value
Bit 2	**Bit 1**	**Bit 0**	
0	0	0	4
0	0	1	6
0	1	0	8
0	1	1	10
1	0	0	12
1	0	1	16
1	1	0	20
1	1	1	32

3.3 ADC Registers

This section provides a bit-by-bit description of the control registers used to program the A/D converter (ADC). Table 3–2 lists the addresses of the ADC registers.

Table 3–2. Addresses of ADC Registers

			Described in	
Address	**Register**	**Name**	**Subsection**	**Page**
7032h	ADCTRL1	ADC Control Register 1	3.3.1	3-6
7034h	ADCTRL2	ADC Control Register 2	3.3.2	3-9
7036h	ADCFIFO1	ADC 2-Level Deep Data Register FIFO for ADC 1	3.3.3	3-10
7038h	ADCFIFO2	ADC 2-Level Deep Data Register FIFO for ADC 2	3.3.3	3-10

3.3.1 ADC Control Register 1 (ADCTRL1)

ADC control register 1 controls conversion start, ADC module enable/disable function, interrupt enable, and end of conversion.

Figure 3–2. ADC Control Register 1 (ADCTRL1) — Address 7032h

15	14	13	12	11	10	9	8
Suspend–soft	Suspend–free	ADCIMSTART	ADC1EN	ADC2EN	ADCCONRUN	ADCINTEN	ADCINTFLAG
RW–0	RW–0	RW–0	SRW–0	SRW–0	SRW–0	SRW–0	RW–0

7	6–4	3–1	0
ADCEOC	ADC2CHSEL	ADC1CHSEL	ADCSOC
R–0	SRW–0	SRW–0	SRW–0

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

Bit 15 **Suspend–soft**. Applicable only during emulation. This bit is not shadowed.

0 = Stop immediately when suspend–free (bit 14) = 0.
1 = Complete conversion before halting emulator.

Bit 14 **Suspend–free**. Applicable only during emulation. This bit is not shadowed.

0 = Operation is determined by suspend–soft (bit 15).
1 = Keep running on emulator suspend.

Bit 13 **ADCIMSTART**. ADC start converting immediately. This bit is not shadowed.

0 = No action
1 = Immediate start of conversion.

Bit 12 **ADC1EN**. Enable/Disable bit for ADC1. This bit is shadowed. This bit can be written while a previous conversion is still going on. However, the effect of writing to this bit will take place from the next conversion.

0 = ADC1 disabled. (No sample/hold/conversion can take place; data register FIFO1 will not change.)
1 = ADC1 is enabled.

Bit 11 **ADC2EN**. Enable/Disable bit for ADC2. This bit is shadowed. This bit can be written while a previous conversion is still going on. However, the effect of writing to this bit will take place from the next conversion.

0 = ADC2 disabled. (No sample/hold/conversion can take place; data register FIFO1 will not change.)
1 = ADC2 is enabled.

Bit 10 **ADCCONRUN**. This bit sets the ADC unit for continuous conversion mode. This bit is shadowed. This bit can be written while a previous conversion is still going on. However, the effect of writing to this bit will take place from the next conversion.

0 = No action
1 = Continuous conversion

Bit 9 **ADCINTEN**. Enable interrupts. If the ADCINTEN bit is set, an interrupt is requested when the ADCINTFLAG is set. This bit is cleared on reset. This bit is shadowed. This bit can be written while a previous conversion is still going on. However, the effect of writing to this bit will take place from the next conversion.

Bit 8 **ADCINTFLAG**. ADC interrupt flag bit. This bit indicates whether an interrupt event has occurred. Writing a 1 to ADCINTFLAG will clear this bit. This bit is not shadowed.

0 = No interrupt event occurred.
1 = An interrupt event occurred.

Bit 7 **ADCEOC**. This bit indicates the status of the ADC conversion. This bit is not shadowed.

0 = End of conversion
1 = Conversion is in progress.

Bits 6–4 **ADC2CHSEL**. Selects channels for ADC2. This bit is shadowed. This bit can be written while a previous conversion is still going on. However, the effect of writing to this bit will take place from the next conversion.

000	= Channel 9
001	= Channel 10
010	= Channel 11
011	= Channel 12
100	= Channel 13
101	= Channel 14
110	= Channel 15
111	= Channel 16

Bits 3–1 **ADC1CHSEL**. Selects channels for ADC1. This bit is shadowed. This bit can be written while a previous conversion is still going on. However, the effect of writing to this bit will take place from the next conversion.

000	= Channel 1
001	= Channel 2
010	= Channel 3
011	= Channel 4
100	= Channel 5
101	= Channel 6
110	= Channel 7
111	= Channel 8

Bit 0 **ADCSOC**. ADC start of conversion (SOC) bit. This bit is shadowed. This bit can be written while a previous conversion is still going on. However, the effect of writing to this bit will take place from the next conversion.

0	= No action
1	= Start converting

Note:

Either channel 1 or channel 2 must be enabled before a conversion will start.

3.3.2 ADC Control Register 2 (ADCTRL2)

ADC control register 2 selects ADC input clock prescaler, conversion mode, emulation operation and also shows ADC FIFO status.

Figure 3–3. ADC Control Register 2 (ADCTRL2) — Address 7034h

15–11	10	9	8
Reserved	ADCEVSOC	ADCEXTSOC	Reserved
	SRW–0	SRW–0	

7–6	5	4–3	2–0
ADCFIFO1	Reserved	ADCFIFO2	ADCPSCALE
R–0		R–0	SRW–0

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

Bits 15–11 **Reserved**. Reads are indeterminate and writes have no effect.

Bit 10 **ADCEVSOC**. Event manager SOC mask bit. When set, the ADC conversion can be synchronized with an event manager signal. Event manager can start a conversion depending on a compare match. This bit is shadowed and functions on any other previously described shadowed bit.

0 = Mask ADCEVSOC (Disable conversion start by EV)
1 = Enable conversion start by EV.

Bit 9 **ADCEXTSOC**. External signal mask bit. When set, the ADC conversion can be synchronized with an external signal. This bit is shadowed.

Note:

If EXT_SOC is more than seven CPU clocks long (2 mode), a second conversion will start after the first conversion.

Bit 8 **Reserved**. Reads are indeterminate and writes have no effect.

Bits 7–6 **ADCFIFO1.** Date register FIFO1 status. These two bits indicate ADC1 FIFO status. Two conversion results can be stored before performing any read operations. However, after two conversions if the third conversion is made, the oldest result will be lost. These bits are not shadowed.

00 = FIFO1 is empty.
01 = FIFO1 has one entry.
10 = FIFO1 has two entries.
11 = FIFO1 had two entries and another entry was received; first entry has been lost.

Bit 5 **Reserved**. Reads are indeterminate and writes have no effect.

Bits 4–3 **ADCFIFO2**. Date register FIFO2 status. These two bits indicate ADC2 FIFO status. Two conversion results can be stored before performing any read operations. However, after two conversions if the third conversion is made, the oldest result will be lost. These bits are not shadowed.

00	=	FIFO2 is empty.
01	=	FIFO2 has one entry.
10	=	FIFO2 has two entries.
11	=	FIFO2 had two entries and another entry was received; first entry has been lost.

Bits 2–0 **ADCPSCALE**. ADC input clock prescaler. These bits define the ADC clock prescaler. Sample prescaler times are explained in subsection 3.2.3, *Analog Signal Sampling/Conversion*, on page 3-5. Refer to the following table for prescaler values.

ADCPSCALE Bits			**Prescale Value**
Bit 2	**Bit 1**	**Bit 0**	
0	0	0	4
0	0	1	6
0	1	0	8
0	1	1	10
1	0	0	12
1	0	1	16
1	1	0	20
1	1	1	32

3.3.3 ADC Digital Result Registers

The digital result registers contain a 10-bit digital result following conversion of the analog input. These are read-only registers. On reset, they are cleared. The results are stored in a two-level FIFO. This provides the flexibility of converting two variables before reading them from the data registers. However, if a third conversion is made when there are two unread values in the FIFO, the first converted value will be lost.

Figure 3–4. ADC Data Registers FIFO1 (ADCFIFO1) — Address 7036h and FIFO2 (ADCFIFO2) — Address 7038h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0	0
MSB									LSB						

Bits 15–6 **D9–D0**. Actual 10-bit converted data.

Bits 5–0 **Reserved**. Always read as 0.

Chapter 4

Serial Communications Interface (SCI) Module

The Serial Communications Interface (SCI) module is part of the PRISM Library. The architecture, functions, and programming of the SCI module are described in this chapter.

Note: 8-Bit Peripheral

This module is interfaced to the 16-bit peripheral bus as an 8-bit peripheral. Therefore, reads from bits 15–8 are undefined; writes to bits 15–8 have no effect.

Topic **Page**

4.1 SCI Overview

The programmable SCI supports digital communications between the CPU and other asynchronous peripherals that use the standard NRZ (nonreturn-to-zero) format. The SCI's receiver and transmitter are double buffered, and each has its own separate enable and interrupt bits. Both may be operated independently or simultaneously in the full-duplex mode.

To ensure data integrity, the SCI checks received data for break detection, parity, overrun, and framing errors. The speed of bit rate (baud) is programmable to over 65 000 different speeds through a 16-bit baud-select register.

Note: Register Bit Notation

For convenience, references to a bit in a register are abbreviated using the register name followed by a period and the number of the bit. For example, the notation for bit 0 of SCI port control register 1 (SCIPC1) is SCIPC1.0.

4.1.1 SCI Physical Description

The SCI module, shown in Figure 4–1, key features are:

❑ Two I/O pins:
  - ■ SCIRXD (SCI receive data input)
  - ■ SCITXD (SCI transmit data output)

❑ Programmable bit rates to over 65 000 different speeds through a 16-bit baud select register
  - ■ Range at 10-MHz SYSCLK: 19.07 bps to 625.0 kbps
  - ■ Number of bit rates: 64K

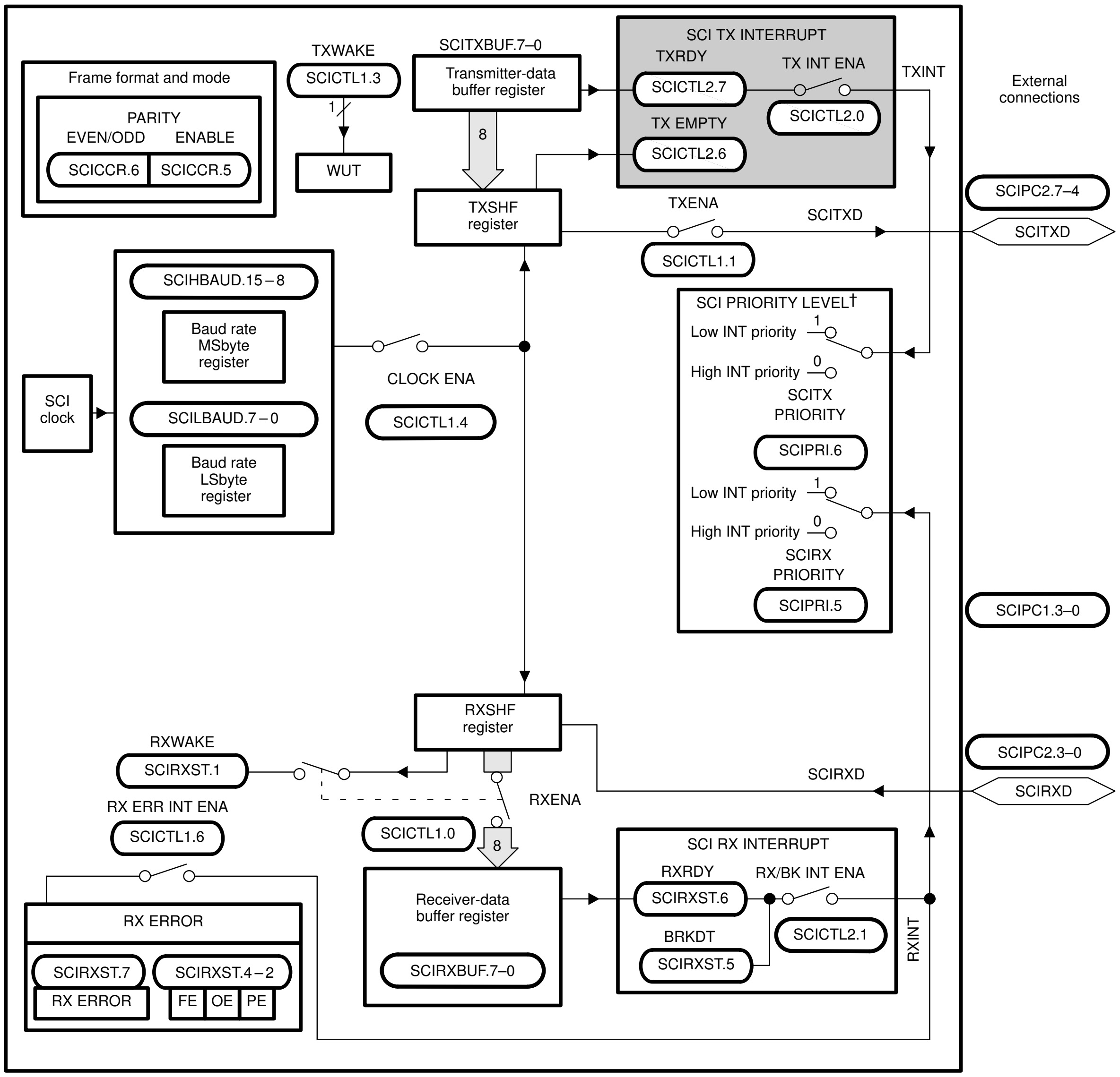
❑ Programmable data word length from one to eight bits

❑ Programmable stop bits of either one or two bits in length

❑ Internally generated serial clock

❑ Four error detection flags:
  - ■ Parity error
  - ■ Overrun error
  - ■ Framing error
  - ■ Break detect

Figure 4–1. SCI Block Diagram



❑ Two wake-up multiprocessor modes that can be used with either communications format:
  - Idle-line wake-up
  - Address-bit wake-up

❑ Half- or full-duplex operation

❑ Double-buffered receive and transmit functions

- Transmitter and receiver operation can be operated through interrupts or through polled operation because of status flags:
  - Transmitter: TXRDY flag (transmitter buffer register is ready to receive another character) and TX EMPTY flag (transmit shift register is empty)
  - Receiver: RXRDY flag (receive buffer register ready to receive another character), BRKDT flag (break condition occurred), and RX ERROR monitoring four interrupt conditions
- Separate enable bits for transmitter and receiver interrupts (except break)
- NRZ (nonreturn-to-zero) format

4.1.2 Architecture

The major elements used in full duplex are shown in Figure 4–1 (page 4-3) and include:

- A transmitter (TX) and its major registers:
  - SCITXBUF — transmitter data buffer register. Contains data (loaded by the CPU) to be transmitted.
  - TXSHF — transmitter shift register. Loads data from SCITXBUF and shifts data onto the SCITXD pin, one bit at a time.
- A receiver (RX) and its major registers:
  - RXSHF — receiver shift register. Shifts data in from SCIRXD pin, one bit at a time.
  - SCIRXBUF — receiver data buffer register. Contains data to be read by the CPU. Data from a remote processor is loaded into RXSHF and then into SCIRXBUF and SCIRXEMU.
- A programmable baud generator.
- Data memory-mapped control and status registers.

The SCI receiver and transmitter can operate independently or simultaneously.

4.1.3 SCI Control Registers

The SCI control registers and the function of each location is shown in Table 4–1. This is a general representation; see Section 4.6, *SCI Control Registers*, on page 4-18 for more detail.

Table 4–1. Addresses of SCI Registers

				Described in	
Address	**Register**	**Name**	**Description**	**Subsection**	**Page**
7050h	SCICCR	SCI communication control register	Defines the character format, protocol, and communications mode used by the SCI.	4.6.1	4-19
7051h	SCICTL1	SCI control register 1	Controls the RX/TX and receiver error interrupt enable, TXWAKE and SLEEP functions, internal clock enable, and the SCI software reset.	4.6.2	4-21
7052h	SCIHBAUD	SCI baud register, high bits	Stores the data (MSbyte) required to generate the bit rate.	4.6.3	4-24
7053h	SCILBAUD	SCI baud register, low bits	Stores the data (LSbyte) required to generate the bit rate.	4.6.3	4-24
7054h	SCICTL2	SCI control register 2	Contains the transmitter interrupt enable, the receiver-buffer/break interrupt enable, the transmitter ready flag, and the transmitter empty flag.	4.6.4	4-25
7055h	SCIRXST	SCI receiver status register	Contains seven receiver status flags.	4.6.5	4-26
7056h	SCIRXEMU	SCI emulation data buffer register	Contains data received for screen updates, principally used by the emulator.	4.6.6	4-28
7057h	SCIRXBUF	SCI receiver data buffer register	Contains the current data from the receiver shift register.	4.6.6	4-28
7058h	—	Reserved	Reserved		
7059h	SCITXBUF	SCI transmit data buffer register	Stores data bits to be transmitted by the SCITX.	4.6.7	4-29
705Ah	—	Reserved	Reserved		
705Bh	—	Reserved	Reserved		
705Ch	—	Reserved	Reserved		
705Dh	—	Reserved	Reserved		
705Eh	SCIPC2	SCI port control register 2	Controls the SCIRXD and SCITXD pin functions.	4.6.8	4-30
705Fh	SCIPRI	SCI priority control register	Contains the receiver and transmitter interrupt priority select bits and the emulator suspend enable bit.	4.6.9	4-32

4.1.4 Multiprocessor and Asynchronous Communications Modes

The SCI has two multiprocessor protocols, the **idle-line** multiprocessor mode (see subsection 4.3.1 on page 4-9) and the **address-bit** multiprocessor mode (see subsection 4.3.2 on page 4-11). These protocols allow efficient data transfer between multiple processors.

The SCI offers the universal asynchronous receiver/transmitter (UART) communications mode for interfacing with many popular peripherals. The asynchronous mode (see Section 4.4, *SCI Communication Format*, on page 4-13) requires two lines to interface with many standard devices such as terminals and printers that use RS-232-C formats. Data transmission characteristics include:

- ❑ One start bit
- ❑ One to eight data bits
- ❑ An even/odd parity bit or no parity bit
- ❑ One or two stop bits

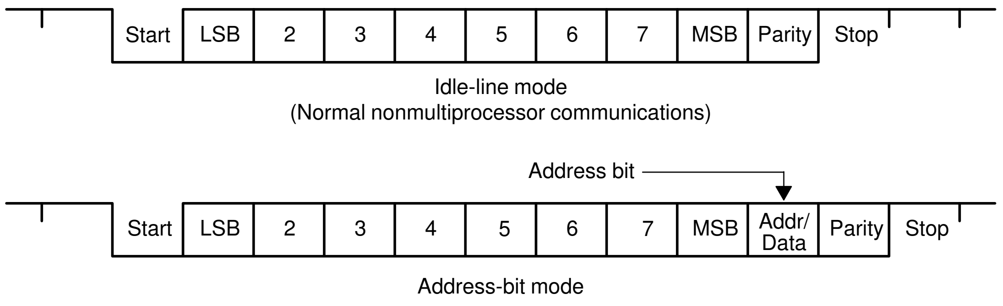
4.2 SCI Programmable Data Format

SCI data, both receive and transmit, is in NRZ (nonreturn-to-zero) format. The NRZ data format, shown in Figure 4–2, consists of:

❑ One start bit
❑ One to eight data bits
❑ An even/odd parity bit (optional)
❑ One or two stop bits
❑ An extra bit to distinguish addresses from data (address-bit mode only)

The basic unit of data is called a character and is one to eight bits in length. Each character of data is formatted with a start bit, one or two stop bits, and optional parity and address bits. A character of data with its formatting information is called a frame and is shown in Figure 4–2.

Figure 4–2. Typical SCI Data Frame Formats



To program the data format, use the SCI communication control register (SCICCR) described in subsection 4.6.1 on page 4-19. The bits used to program the data format are listed in Table 4–2.

Table 4–2. Programming the Data Format Using SCICCR

Bit Name	Designation	Functions
SCI CHAR2–0	SCICCR.2–0	Selects the character (data) length (one to eight bits). Bit values are shown in Table 4–4 (page 4-20).
PARITY ENABLE	SCICCR.5	Enables the parity function if set to 1, or disables the parity function if cleared to 0.
EVEN/ODD PARITY	SCICCR.6	If parity is enabled, selects odd parity if cleared to 0 or even parity if set to 1.
STOP BITS	SCICCR.7	Determines the number of stop bits transmitted—one stop bit if cleared to 0 or two stop bits if set to 1.

4.3 SCI Multiprocessor Communication

The multiprocessor communication format allows one processor to efficiently send blocks of data to other processors on the same serial link. On one serial line, there should be only one transfer at a time. In other words, there can be only one talker on a serial line at a time.

The **first byte** of a block of information that the talker sends contains an **address byte**, that is read by all listeners. Only listeners with the correct address can be interrupted by the data bytes that follow the address byte. The listeners with an incorrect address remain uninterrupted until the next address byte.

All processors on the serial link set their SCI's SLEEP bit (SCICTL1.2) to 1 so that they are interrupted only when the address byte is detected. When a processor reads a block address that corresponds to the CPU's device address as set by software (of your application), your program must clear the SLEEP bit to enable the SCI to generate an interrupt on receipt of each data byte.

Although the receiver still operates when the SLEEP bit is 1, it does not set RXRDY, RXINT, or any of the receive error status bits to 1 unless the address byte is detected and the address bit in the received frame is a 1. The SCI does not alter the SLEEP bit; your software must alter the SLEEP bit.

A processor recognizes an address byte according to the multiprocessor mode. For example:

❑ The **idle-line mode** (subsection 4.3.1 on page 4-9) leaves a quiet space before the address byte. This mode does not have an extra address/data bit and is more efficient than the address-bit mode for handling blocks that contain more than 10 bytes of data. The idle-line mode should be used for typical nonmultiprocessor SCI communication.

❑ The **address-bit mode** (subsection 4.3.2 on page 4-11) adds an extra bit (address bit) into every byte to distinguish addresses from data. This mode is more efficient in handling many small blocks of data because, unlike the idle mode, it does not have to wait between blocks of data. However, at high transmit speed, the program is not fast enough to avoid a 10-bit idle in the transmission stream.

The multiprocessor mode is software selectable via the ADDR/IDLE MODE bit (SCICCR.3). Both modes use the TXWAKE flag bit (SCICTL1.3), RXWAKE flag bit (SCIRXST.1), and the SLEEP flag bits (SCICTL1.2) to control the SCI transmitter and receiver features of these modes.

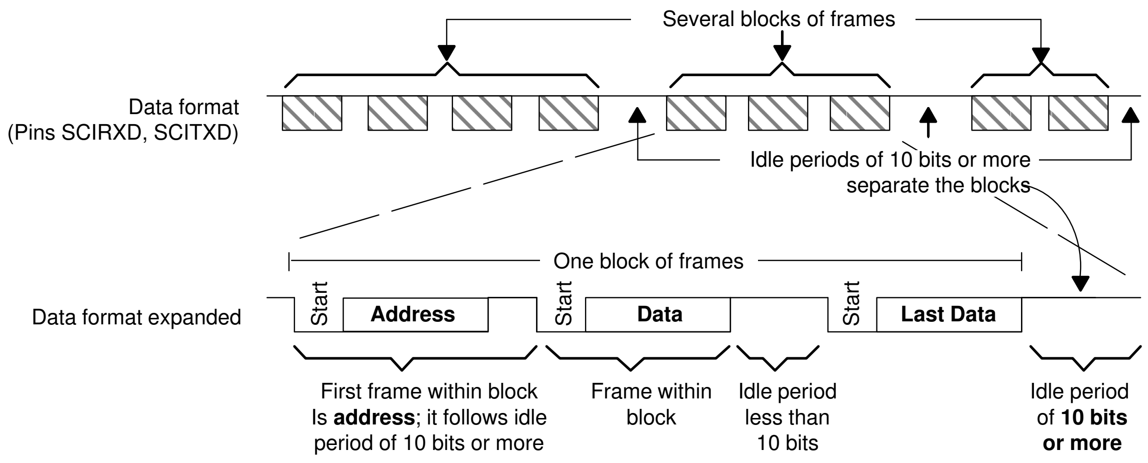
In both multiprocessor modes, the receipt sequence is:

1) At the receipt of an address block, the SCI port wakes up and requests an interrupt (bit RX/BK INT ENA—SCICTL2.1—must be enabled to request an interrupt). It reads the first frame of the block, which contains the destination address.
2) A software routine is entered through the interrupt and checks the RXWAKE flag bit. If the RXWAKE bit is 1, the incoming byte is an address (otherwise, the byte is data) and this address byte is checked against its device address byte stored in memory.
3) If the check shows that the block is addressed to the DSP controller, the CPU clears the SLEEP bit and reads the rest of the block; if not, the software routine exits with the SLEEP bit still set and does not receive interrupts until the next block start.

4.3.1 Idle-Line Multiprocessor Mode

In the Idle-line multiprocessor protocol (ADDR/IDLE MODE bit=0), blocks are separated by having a longer idle time between the blocks than between frames in the blocks. An **idle time** of ten or more high-level bits after a frame indicates the start of a new block (the time of a single bit is calculated directly from the baud value (bits per second). The idle-line multiprocessor communication format is shown in Figure 4–3 (ADDR/IDLE MODE bit is SCICCR.3).

Figure 4–3. Idle-Line Multiprocessor Communication Format



The steps followed by the idle-line mode:

1) SCI wakes up after receipt of the block-start signal.
2) The processor now recognizes the next SCI interrupt.
3) The service routine compares the received address (sent by a remote transmitter) to its own.
4) If the CPU is being addressed, the service routine clears the SLEEP bit and receives the rest of the data block.
5) If the CPU is not being addressed, the SLEEP bit remains set. This lets the CPU continue to execute its main program without being interrupted by the SCI port until the next detection of a block start.

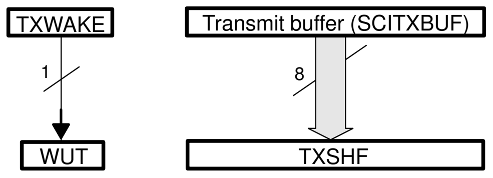
There are two ways to send a block start signal:

❑ **Method 1:** Deliberately leave an idle time of ten bits or more by delaying the time between the transmission of the last frame of data in the previous block and the transmission of the address frame of the new block.

❑ **Method 2:** The SCI port first sets the TXWAKE bit (SCICTL1.3) to 1 before writing to the SCITXBUF. This sends an idle time of exactly 11 bits. In this method, the serial communications line is not idle any longer than necessary.

Associated with the TXWAKE bit is the wake-up temporary (WUT) flag. WUT is an internal flag, double-buffered with TXWAKE. When TXSHF is loaded from SCITXBUF, WUT is loaded from TXWAKE, and the TXWAKE bit is cleared to 0. This arrangement is shown in Figure 4–4. (The SCI block diagram, Figure 4–1 on page 4-3, shows this in additional detail.)

Figure 4–4. Double-Buffered WUT and TXSHF



Note: WUT = Wake Up Temporary

To send out a block start signal of exactly one frame time during a sequence of block transmissions:

1) Write a 1 to the TXWAKE bit.

2) Write a data word (content not important — a don't care) to SCITXBUF (transmit data buffer) to send a block-start signal. (The first data word written is suppressed while the block-start signal is sent out and ignored after that.) When TXSHF (transmit shift register) is free again, SCITXBUF's contents are shifted to TXSHF, the TXWAKE value is shifted to WUT, and then TXWAKE is cleared.

   Because TXWAKE was set to a 1, the start, data, and parity bits are replaced by an idle period of 11 bits transmitted following the last stop bit of the previous frame.

3) Write a new address value to SCITXBUF.

A don't-care data word must first be written to SCITXBUF so that the TXWAKE bit value can be shifted to WUT. After the don't-care data word is shifted to TXSHF, SCITXBUF (and TXWAKE, if necessary) can be written to again, because TXSHF and WUT are both double-buffered.

The receiver operates, regardless of the SLEEP bit. However, the receiver neither sets RXRDY nor the error status bits, nor does it request a receive interrupt until an address frame is detected.

4.3.2 Address-Bit Multiprocessor Mode

In the address-bit protocol (ADDR/IDLE MODE bit = 1), frames have an extra bit, called an address bit, that immediately follows the last data bit. The address bit is set to 1 in the first frame of the block and to 0 in all other frames. The idle period timing is irrelevant (see Figure 4–5, ADDR/IDLE MODE bit is SCICCR.3).

The TXWAKE bit value is placed in the address bit. During transmission, when the SCITXBUF and TXWAKE are loaded into the TXSHF and WUT respectively, TXWAKE is reset to 0 and WUT becomes the value of the address bit of the current frame. Thus, to send an address:

1) Set the TXWAKE bit to 1 and write the appropriate address value to SCITXBUF.

2) When this address value is transferred to TXSHF and shifted out, its address bit is sent as a 1, which flags the other processors on the serial link to read the address.

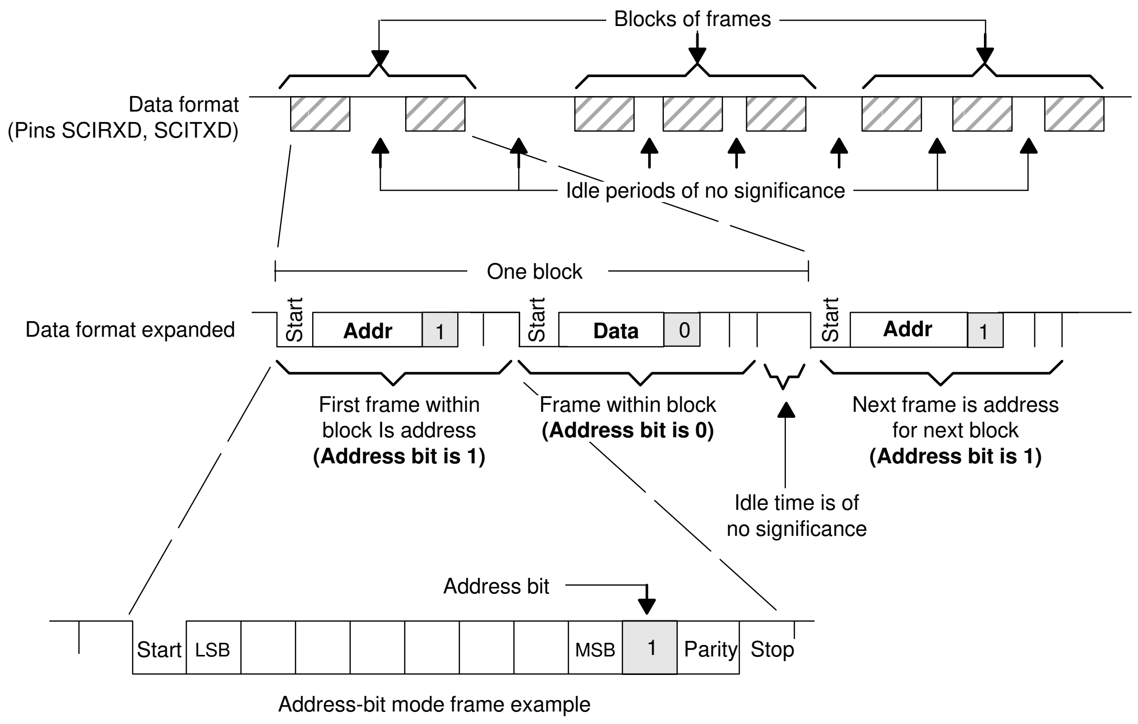
3) Since TXSHF and WUT are both double-buffered, SCITXBUF and TXWAKE can be written to immediately after TXSHF and WUT are loaded.

4) To transmit nonaddress frames in the block, leave the TXWAKE bit set to 0.

Note: Address-Bit Format for Transfers of 11 Bytes or Less

As a general rule, the address-bit format is typically used for data frames of 11 bytes or less. This format adds one bit value (1 for an address frame, 0 for a data frame) to all data bytes transmitted. The idle-line format is typically used for data frames of 12 bytes or more.

Figure 4–5. Address-Bit Multiprocessor Communication Format



4.4 SCI Communication Format

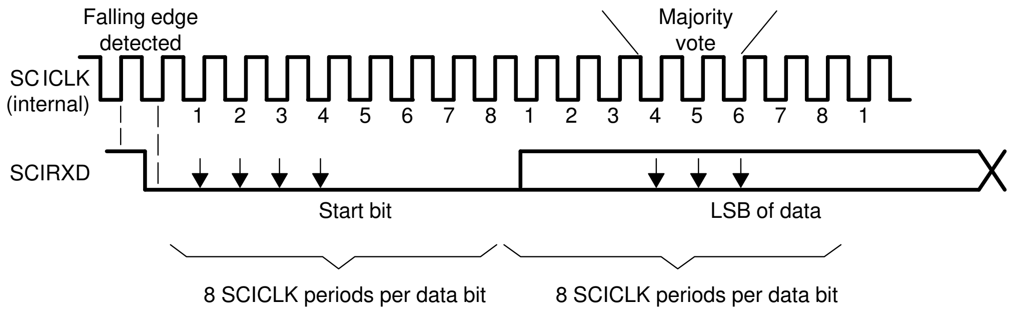
The SCI asynchronous communication format uses either single line (one way) or two line (two way) communications. In this mode, the frame consists of a start bit, one to eight data bits, an optional even/odd parity bit, and one or two stop bits (shown in Figure 4–6). There are **8 SCICLK periods** per data bit.

The receiver begins operation on receipt of a valid start bit. A valid start bit is identified by four consecutive internal SCICLK periods of zero bits as shown in Figure 4–6. If any bit is not zero, then the processor starts over and begins looking for another start bit.

For the bits following the start bit, the processor determines the bit value by making three samples in the middle of the bits. These samples occur on the fourth, fifth, and sixth SCICLK periods, and bit-value determination is on a majority (two out of three) basis. Figure 4–6 illustrates the asynchronous communication format for this with a start bit showing how edges are found and where a majority vote is taken.

Since the receiver synchronizes itself to frames, the external transmitting and receiving devices do not have to use a synchronized serial clock; the clock can be generated locally.

Figure 4–6. SCI – Asynchronous Communications Format

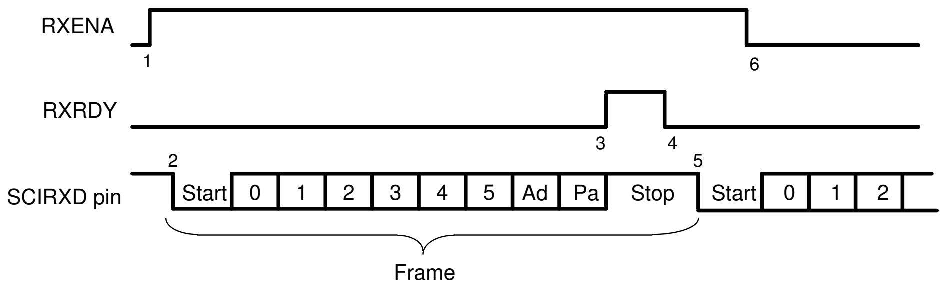


4.4.1 Receiver Signals in Communications Modes

Figure 4–7 illustrates an example of receiver signal timing that assumes the following conditions:

- Address-bit wake-up mode (address bit would not appear in idle-line mode)
- 6 bits per character

Figure 4–7. SCI RX Signals in Communication Modes



Notes:
1) Flag bit RXENA (SCICTL1.0) goes high to enable the receiver.
2) Data arrives on the SCIRXD pin, start bit detected.
3) Data is shifted from RXSHF to the receive buffer register (SCIRXBUF); an interrupt is requested. Flag bit RXRDY (SCIRXST.6) goes high to signal that a new character has been received.
4) The program reads SCIRXBUF, flag RXRDY is automatically cleared.
5) The next byte of data arrives on the SCIRXD pin; the start bit is detected, then cleared.
6) Bit RXENA is brought low to disable the receiver. Data continues to be assembled in RXSHF but is not transferred to the receive buffer register.

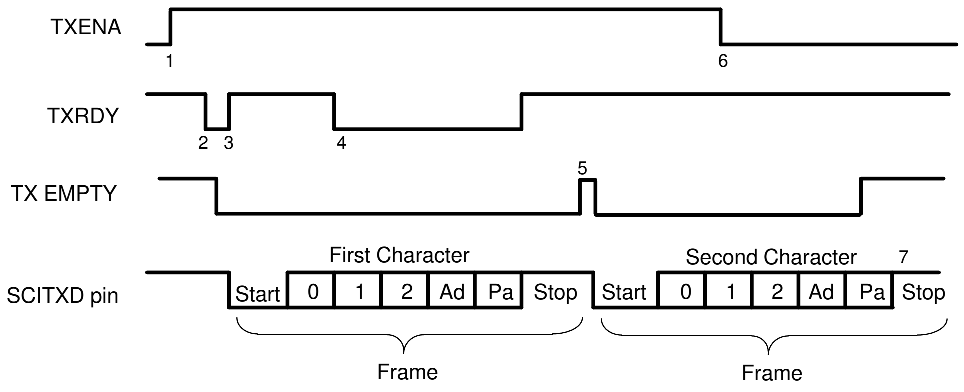
4.4.2 Transmitter Signals in Communications Modes

Figure 4–8 illustrates an example of transmitter signal timing that assumes the following conditions:

❑ Address-bit wake-up mode (address bit would not appear in idle-line mode)

❑ 3 bits per character

Figure 4–8. SCI TX Signals in Communications Modes



Notes:
1) Bit TXENA (SCICTL1.1) goes high, enabling the transmitter to send data.
2) SCITXBUF is written to; thus, (1) the transmitter is no longer empty, and (2) TXRDY goes low.
3) The SCI transfers data to the shift register (TXSHF). The transmitter is ready for a second character (TXRDY goes high), and it requests an interrupt (to enable an interrupt, bit TX INT ENA — SCICTL2.0 — must be set).
4) The program writes a second character to SCITXBUF after TXRDY goes high (item 3).
5) Transmission of the first character is complete. TX EMPTY goes high temporarily. Transfer of the second character to shift register TXSHF begins.
6) Bit TXENA goes low to disable the transmitter; the SCI finishes transmitting the current character.
7) Transmission of the second character is complete; transmitter is empty and ready for new character.

4.5 SCI Port Interrupts

The internally-generated serial clock is determined by the SYSCLK frequency and the bank-select registers. The SCI uses the 16-bit value of the baud-select registers to select one of 64K different serial clock rates.

The SCI's receiver and transmitter can be interrupt controlled. The SCICTL2 has one flag bit (TXRDY) that indicates active interrupt conditions and SCIRXST has two interrupt flag bits (RXRDY and BRKDT). The transmitter and receiver have separate interrupt-enable bits. When not enabled, the interrupts are not asserted; however, the condition flags remain active, reflecting transmission and receipt status.

The SCI provides independent interrupt requests and vectors for the receiver and transmitter.

- If the RX/BK INT ENA bit (SCICTL2.1) is set, the receiver interrupt is asserted when one of the following events occurs:
  - The SCI receives a complete frame and transfers the data in RXSHF to SCIRXBUF. This action sets the RXRDY flag (SCIRXST.6) and initiates an interrupt.
  - A break detect condition occurs (the SCIRXD is low for 10 bit periods following a stop bit). This action sets the BRKDT flag bit (SCIRXST.5) and initiates an interrupt.
- If the TX INT ENA bit (SCICTL2.0) is set, the transmitter interrupt is asserted whenever the data in SCITXBUF is transferred to TXSHF, indicating that the CPU can write to TXBUF; this action sets the TXRDY flag bit (SCICTL2.7) and initiates an interrupt.

SCI interrupts can be programmed onto different priority levels by the SCIRX PRIORITY (SCIPRI.5) and SCITX PRIORITY (SCIPRI.6) control bits. When both RX and TX interrupt requests are made on the same level, the receiver always has higher priority than the transmitter, reducing the possibility of receiver overrun.

4.5.1 SCI Baud Rate Calculation

The internally-generated serial clock is determined by the SYSCLK frequency and the baud-select registers. The SCI uses the 16-bit value of the baud-select registers to select one of the 64K different serial clock rates.

The SCI baud rate for the different communication modes is determined in the following ways:

❑ SCI Asynchronous Baud for BRR=1 to 65535

$$SCI\ Asynchronous\ Baud = \frac{SYSCLK}{(BRR + 1) \times 8}$$

$$BRR = \frac{SYSCLK}{SCI\ Asynchronous\ Baud \times 8} - 1$$

❑ SCI Asynchronous Baud for BRR = 0

$$SCI\ Asynchronous\ Baud = \frac{SYSCLK}{16}$$

Where BRR = The 16-bit value in the baud-select registers.

Refer to Table 4–3. The baud-select registers are further defined in subsection 4.6.3 on page 4-24.

Table 4–3. Asynchronous Baud Register Values for Common SCI Bit Rates

	SYSCLK Frequency											
	1 MHz			5 MHz			8 MHz			10 MHz		
Ideal Baud Selected	BRR	Actual Baud Selected	% Error	BRR	Actual Baud Selected	% Error	BRR	Actual Baud Selected	% Error	BRR	Actual Baud Selected	% Error
300	416	300	–0.08	2082	300	0.02	3332	300	0.01	4166	300	–0.01
600	207	601	0.16	1041	600	–0.03	1666	600	–0.02	2082	600	0.02
1200	103	1202	0.16	520	1200	–0.03	832	1200	0.04	1041	1200	–0.03
2400	51	2404	0.16	259	2404	0.16	416	2398	–0.08	520	2399	–0.03
4800	25	4808	0.16	129	4808	0.16	207	4808	0.16	259	4808	0.16
8192	14	8333	1.73	75	8224	0.39	121	8197	0.06	152	8170	–0.27
9600	12	9615	0.16	64	9615	0.16	103	9615	0.16	130	9542	–0.60
19200	6	17857	–6.99	32	18939	–1.36	51	19231	0.16	64	19231	0.16

4.6 SCI Control Registers

The functions of the SCI are software configurable. Sets of control bits, organized into dedicated bytes, are programmed to initialize the desired SCI communications format. This includes operating mode and protocol, baud value, character length, even/odd parity or no parity, number of stop bits, and interrupt priorities and enables. The SCI is controlled and accessed through registers listed in Figure 4–9 and described in the subsections that follow.

Figure 4–9. SCI Control Registers

		Bit number							
Address	**Register**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
7050h	SCICCR	STOP BITS	EVEN/ODD PARITY	PARITY ENABLE	SCI ENA	ADDR/ IDLE MODE	SCI CHAR2	SCI CHAR1	SCI CHAR0
7051h	SCICTL1	Reserved	RX ERR INT ENA	SW RESET	CLOCK ENA	TXWAKE	SLEEP	TXENA	RXENA
7052h	SCIHBAUD	BAUD15 (MSB)	BAUD14	BAUD13	BAUD12	BAUD11	BAUD10	BAUD9	BAUD8
7053h	SCILBAUD	BAUD7	BAUD6	BAUD5	BAUD4	BAUD3	BAUD2	BAUD1	BAUD0 (LSB)
7054h	SCICTL2	TXRDY	TX EMPTY	Reserved				RX/BK INT ENA	TX INT ENA
7055h	SCIRXST	RX ERROR	RXRDY	BRKDT	FE	OE	PE	RXWAKE	Reserved
7056h	SCIRXEMU	ERXDT7	ERXDT6	ERXDT5	ERXDT4	ERXDT3	ERXDT2	ERXDT1	ERXDT0
7057h	SCIRXBUF	RXDT7	RXDT6	RXDT5	RXDT4	RXDT3	RXDT2	RXDT1	RXDT0
7058h	—	Reserved							
7059h	SCITXBUF	TXDT7	TXDT6	TXDT5	TXDT4	TXDT3	TXDT2	TXDT1	TXDT0
705Ah	—	Reserved							
705Bh	—	Reserved							
705Ch	—	Reserved							
705Dh	—	Reserved							
705Eh	SCIPC2	SCITXD DATA IN	SCITXD DATA OUT	SCITXD FUNCTION	SCITXD DATA DIR	SCIRXD DATA IN	SCIRXD DATA OUT	SCIRXD FUNCTION	SCIRXD DATA DIR
705Fh	SCIPRI	Reserved	SCITX PRIORITY	SCIRX PRIORITY	SCI ESPEN	Reserved			

4.6.1 SCI Communication Control Register (SCICCR)

The SCI communication control register (SCICCR) defines the character format, protocol, and communications mode used by the SCI.

Figure 4–10. SCI Communication Control Register (SCICCR) — Address 7050h

7	6	5	4	3	2	1	0
STOP BITS	EVEN/ODD PARITY	PARITY ENABLE	SCI ENA	ADDR/IDLE MODE	SCI CHAR2	SCI CHAR1	SCI CHAR0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

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

Bit 7 **STOP BITS**. SCI number of stop bits. This bit specifies the number of stop bits transmitted. The receiver checks for only one stop bit.

0 = One stop bit
1 = Two stop bits

Bit 6 **PARITY**. SCI parity odd/even selection. If the PARITY ENABLE bit (SCICCR.5) is set, PARITY (bit 6) designates odd or even parity (odd or even number of bits with the value of 1 in both transmitted and received characters).

0 = Odd parity
1 = Even parity

Bit 5 **PARITY ENABLE**. SCI parity enable. This bit enables or disables the parity function. If the SCI is in the address-bit multiprocessor mode (set using bit 3 of this register), the address bit is included in the parity calculation (if parity is enabled). For characters of less than eight bits, the remaining unused bits should be masked out of the parity calculation.

0 = Parity disabled; no parity bit is generated during transmission or is expected during reception.
1 = Parity is enabled.

Bit 4 **SC I ENA**. SCI communication enable bit. This bit must be set to 1 for correct operation.

Bit 3 **ADDR/IDLE MODE**. SCI multiprocessor mode control bit. This bit selects one of the multiprocessor protocols:

0 = Idle-line mode protocol is selected.
1 = Address-bit mode protocol is selected.

Multiprocessor communication is different from the other communication modes because it uses SLEEP and TXWAKE functions (bits SCICTL1.2 and SCICTL1.3, respectively). (Both of these bits are further described in Section 4.3, *SCI Multiprocessor Communication*, on page 4-8). The idle-line mode is usually used for normal communications because the address-bit mode adds an extra bit to the frame. The idle-line mode does not add this extra bit and is compatible with RS-232-type communications.

Bits 2–0 **SCI CHAR2–0**. Character-length control bits. These bits select the SCI character length from one to eight bits. Characters of less than eight bits are right-justified in SCIRXBUF and SCIRXEMU, and are filled with leading zeros in SCIRXBUF. SCITXBUF is not filled with leading zeros. Table 4–4 lists the bit values and character lengths for SCI CHAR2–0 bits.

Table 4–4. SCI CHAR2–0 Bit Values and Character Lengths

SCI CHAR2–0 Bit Values			
SCI CHAR2	**SCI CHAR1**	**SCI CHAR0**	**Character Length (Bits)**
0	0	0	1
0	0	1	2
0	1	0	3
0	1	1	4
1	0	0	5
1	0	1	6
1	1	0	7
1	1	1	8

4.6.2 SCI Control Register 1 (SCICTL1)

The SCI control register 1 (SCICTL1) controls the receiver/transmitter enable, TXWAKE and SLEEP functions, internal clock enable, and the SCI software reset.

Figure 4–11. SCI Control Register 1 (SCICTL1) — Address 7051h

7	6	5	4	3	2	1	0
Reserved	RX ERR INT ENA	SW RESET	CLOCK ENA	TXWAKE	SLEEP	TXENA	RXENA
	RW–0	RW–0	RW–0	RS–0	RW–0	RW–0	RW–0

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

Bit 7 **Reserved**. Reads are indeterminate and writes have no effect.

Bit 6 **RX ERR INT ENA**. SCI receiver enable. Setting this bit enables an interrupt if the RX ERROR bit (SCIRXST.7) becomes set because of errors occurring.

0 = Receive error interrupt is disabled.
1 = Receive error interrupt is enabled.

Bit 5 **SW RESET**. SCI software reset (active low). Writing a 0 to this bit initializes the SCI state machines and operating flags (SCICTL2 and SCIRXST) to the reset condition.

The SW RESET bit does not affect any of the configuration bits and does not change the state of the CLOCK ENA bit (described on the next page).

All affected logic is held in the specified reset state until a 1 is written to SW RESET (the bit values following a reset are shown beneath each register diagram in this section). Thus, after a system reset, reenable the SCI by writing a 1 to this bit.

Clear this bit after a receiver break detect (BRKDT flag, bit SCIRXST.5).

SW RESET affects the operating flags of the SCI, but it neither affects the configuration bits nor restores the reset values. Table 4–5 lists the affected flags.

Once SW RESET is asserted, the flags are frozen until the bit is deasserted.

Note: Do Not Change Configuration when SW RESET bit = 1

The SCI configuration should not be set or altered unless the SW RESET bit is cleared. Set up all configuration registers before setting SW RESET; otherwise, unpredictable behavior can occur.

Table 4–5. SW RESET-Affected Flags

SCI Flag	Register.Bit	Value After SW RESET
TXRDY	SCICTL2.7	1
TX EMPTY	SCICTL2.6	1
RXWAKE	SCIRXST.1	0
PE	SCIRXST.2	0
OE	SCIRXST.3	0
FE	SCIRXST.4	0
BRKDT	SCIRXST.5	0
RXRDY	SCIRXST.6	0
RX ERROR	SCIRXST.7	0

Bit 4 **CLOCK ENA**. SCI internal clock enable. This bit determines the source of the module clock on the SCICLK pin (Figure 4–1, page 4-3):

0 = Disable internal clock
1 = Enable internal clock

Bit 3 **TXWAKE**. SCI transmitter wakeup method select. The TXWAKE bit controls selection of the data-transmit feature, depending on which transmit mode (idle line or address bit) is specified at the ADDR/IDLE MODE bit (SCICCR.3):

0 = Transmit feature is not selected.
1 = Transmit feature selected is dependent on the mode: idle-line or address-bit:
In idle-line mode: write a 1 to TXWAKE, then write data to register SCITXBUF to generate an idle period of 11 data bits.
In address-bit mode: write a 1 to TXWAKE, then write data to SCITX-BUF to set the address bit for that frame to 1.

TXWAKE is not cleared by the SW RESET bit (SCICTL1.5). A system reset along with the transfer of TXWAKE to the WUT flag (Figure 4–4 on page 4-10).

Bit 2 **SLEEP**. SCI sleep. In a multiprocessor configuration, this bit controls the receive sleep function. Clearing this bit brings the SCI out of the sleep mode. This configuration and process are further explained in Section 4.3, *SCI Multi-processor Communication,* on page 4-8.

0 = Sleep mode is disabled.
1 = Sleep mode is enabled.

The receiver still operates when the SLEEP bit is set; however, operation does not update the receive buffer ready bit (SCIRXST.6, RXRDY) or the error status bits (SCIRXST.5–2: BRKDT, FE, OE, and PE) unless the address byte is detected. This bit is not cleared when the address byte is detected.

Bit 1 **TXENA**. SCI transmitter enable. Data is transmitted through the SCITXD pin (Figure 4–1, page 4-3) only when TXENA is set. If reset, transmission is halted but only after all data previously written to SCITXBUF has been sent.

0 = Transmitter is disabled.
1 = Transmitter is enabled.

Bit 0 **RXENA**. SCI receiver enable. Data is received on the SCIRXD pin (Figure 4–1, page 4-3) and is sent to the receive shift register and then the receive buffers. This bit enables or disables the receiver (transfer to the buffers).

0 = Prevent received characters from transfer into SCIRXEMU and SCIRXBUF receive buffers.
1 = Send receive characters into SCIRXEMU and SCIRXBUF.

Clearing RXENA stops received characters from being transferred into the two receive buffers and also stops the generation of receiver interrupts. However, the receiver shift register can continue to assemble characters. Thus, if RXENA is set during the reception of a character, the complete character will be transferred into the receive buffer registers, SCIRXEMU and SCIRXBUF.

4.6.3 Baud-Select Registers (SCIHBAUD and SCILBAUD)

The values in the SCI baud-select registers (SCIHBAUD and SCILBAUD) specify the baud rate for the SCI.

Figure 4–12. SCI Baud-Select MSbyte Register (SCIHBAUD) — Address 7052h

15	14	13	12	11	10	9	8
BAUD15 (MSB)	BAUD14	BAUD13	BAUD12	BAUD11	BAUD10	BAUD9	BAUD8
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

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

Figure 4–13. SCI Baud-Select LSbyte Register (SCILBAUD) — Address 7053h

7	6	5	4	3	2	1	0
BAUD7	BAUD6	BAUD5	BAUD4	BAUD3	BAUD2	BAUD1	BAUD0 (LSB)
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

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

Bits 15–0 **BAUD15–BAUD0**. SCI 16-bit baud selection. SCIHBAUD (MSbyte) and SCILBAUD (LSbyte) concatenate to form a 16-bit baud value.

The internally-generated serial clock is determined by the SYSCLK and the two baud select registers. The SCI uses the 16-bit value of these registers to select one of 64K serial clock rates for the communication modes.

The SCI baud rate for the different communication modes is determined in the following ways:

❑ For BRR = 1 to 65535

$$SCI\ Asynchronous\ Baud = \frac{SYSCLK}{(BRR + 1) \times 8}$$

$$BRR = \frac{SYSCLK}{SCI\ Asynchronous\ Baud \times 8} - 1$$

❑ For BRR = 0

$$SCI\ Asynchronous\ Baud = \frac{SYSCLK}{16}$$

Where BRR = 16-bit value in the baud-select registers.

4.6.4 SCI Control Register 2 (SCICTL2)

SCI control register 2 (SCICTL2) enables the receive-ready, break-detect, and transmit-ready interrupts as well as transmitter-ready and -empty flags.

Figure 4–14. SCI Control Register 2 (SCICTL2) — Address 7054h

7	6	5 – 2	1	0
TXRDY	TX EMPTY	Reserved	RX/BK INT ENA	TX INT ENA
R–1	R–1		RW–0	RW–0

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

Bit 7 **TXRDY**. Transmitter buffer-register ready flag. When set, this bit indicates that the transmit buffer register, SCITXBUF, is ready to receive another character. Writing data to SCITXBUF automatically clears this bit. When set, this flag asserts a transmitter interrupt request if the interrupt-enable bit TX INT ENA (SCICTL2.0) is also set. TXRDY is set to 1 by enabling the SW RESET bit (SCICTL.2) or by a system reset.

0 = SCITXBUF is full.
1 = SCITXBUF is ready to receive the next character.

Bit 6 **TX EMPTY**. Transmitter empty flag. This flag's value indicates the contents of the transmitter's buffer register (SCITXBUF) and shift register (TXSHF). An active SW RESET (SCICTL1.2) or a system reset sets this bit. This bit does not cause an interrupt request.

0 = Transmitter buffer or shift register or both are loaded with data.
1 = Transmitter buffer and shift registers are both empty.

Bits 5–2 **Reserved**. Reads are indeterminate and writes have no effect.

Bit 1 **RX/BK INT ENA**. Receiver-buffer/break interrupt enable. This bit controls the interrupt request caused by either the RXRDY flag or the BRKDT flag (bits SCIRXST.6 and .5) being set. However, RX/BRK INT ENA does not prevent the setting of these flags.

0 = Disable RXRDY/BRKDT interrupt
1 = Enable RXRDY/BRKDT interrupt

Bit 0 **TX INT ENA**. SCITXBUF-interrupt enable. This bit controls issuing an interrupt request caused by setting the TXRDY flag bit (SCICTL2.7). However, it doesn't prevent the TXRDY flag from being set (being set indicates that SCITXBUF is ready to receive another character).

0 = Disable TXRDY interrupt
1 = Enable TXRDY interrupt

4.6.5 Receiver Status Register (SCIRXST)

The SCI receiver status register (SCIRXST) contains seven bits that are receiver status flags (two of which can generate interrupt requests). Each time a complete character is transferred to the receive buffers (SCIRXEMU and SCIRXBUF), the status flags are updated. Each time the buffers are read, the flags are cleared. Figure 4–16 on page 4-28 shows the relationships between several of the register's bits.

Figure 4–15. SCI Receiver Status Register (SCIRXST) — Address 7055h

7	6	5	4	3	2	1	0
RX ERROR	RXRDY	BRKDT	FE	OE	PE	RXWAKE	Reserved
R–0	R–0	R–0	R–0	R–0	R–0	R–0	

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

Bit 7 **RX ERROR**. SCI receiver-error flag. The RX ERROR flag indicates that one of the error flags in the receiver status register is set. RX ERROR is a logical OR of the break detect, framing error, overrun, and parity enable flags (bits 5–2: BRKDT, FE, OE, and PE).

0 = No error flags are set.
1 = Error flag(s) is set.

A 1 on this bit will cause an interrupt if the RX ERR INT ENA bit (SCICTL1.6) is set. This bit can be used for fast error-condition checking during the interrupt service routine. This error flag cannot be cleared directly; it is cleared by an active SW RESET or by a system reset.

Bit 6 **RXRDY**. SCI receiver-ready flag. When a new character is ready to be read into SCIRXBUF, the receiver sets this bit, and a receiver interrupt is generated if the RX/BK INT ENA bit (SCICTL2.1) is a 1. RXRDY is cleared by reading SCIRXBUF, by an active SW RESET, or by a system reset.

0 = No new character in SCIRXBUF.
1 = Character ready to be read from SCIRXBUF.

Bit 5 **BRKDT**. SCI break-detect flag. The SCI sets this bit when a break condition occurs. A break condition occurs when the SCI receive data line (SCIRXD, right side of Figure 4–1, page 4-3) remains continuously low for at least ten bits, beginning after a missing first stop bit. The occurrence of a break causes a receiver interrupt to be generated if the RX/BK INT ENA bit is a 1, but it does not cause the receiver buffer to be loaded. A BRKDT interrupt can occur, even if the receiver SLEEP bit is set to 1.

BRKDT is cleared by an active SW RESET or by a system reset. It is not cleared by receipt of a character after the break is detected. In order to receive more characters, the SCI must be reset by toggling the SW RESET bit or by a system reset.

0 = No break condition
1 = Break condition occurred

Bit 4 **FE**. SCI framing-error flag. The SCI sets this bit when an expected stop bit is not found. Only the first stop bit is checked. The missing stop bit indicates that synchronization with the start bit has been lost and that the character is incorrectly framed. It is reset by clearing the SW RESET bit or by a system reset.

0 = No framing error is detected.
1 = Framing error is detected.

Bit 3 **OE**. SCI overrun-error flag. The SCI sets this bit when a character is transferred into SCIRXEMU and SCIRXBUF before the previous character is fully read by the CPU. The previous character is overwritten and lost. The OE flag is reset by an active SW RESET or by a system reset.

0 = No overrun error is detected.
1 = Overrun error is detected.

Bit 2 **PE**. SCI parity-error flag. This flag bit is set when a character is received with a mismatch between the number of 1s and its parity bit. The address bit is included in the calculation. If parity generation and detection is not enabled, the PE flag is disabled and read as 0. The PE bit is reset by an active SW RESET or a system reset.

0 = No parity error **or** parity is disabled.
1 = Parity error is detected.

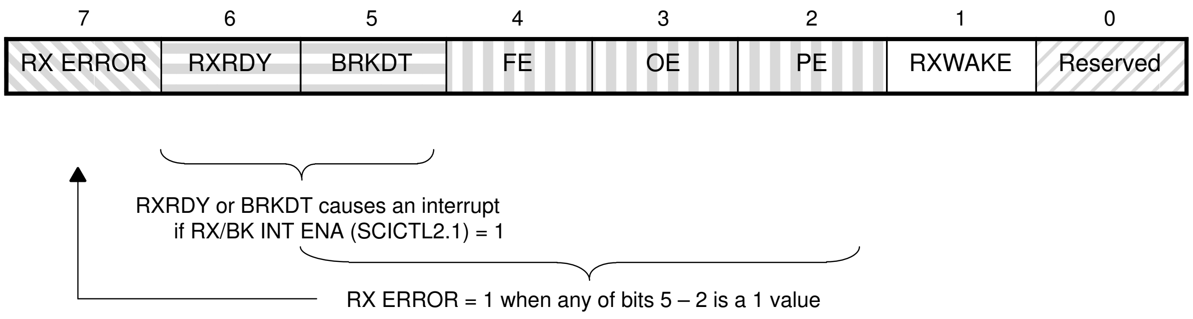
Bit 1 **RXWAKE**. Receiver wakeup-detect flag. A value of 1 in this bit indicates detection of a receiver wakeup condition. in the address bit multiprocessor mode (SCICCR.3 = 1), RXWAKE reflects the value of the address bit for the character contained in SCIRXBUF. In the idle-line multiprocessor mode, RXWAKE is set if the SCIRXD data line is detected as idle (this line is the input to RXSHF as shown in Figure 4–1, page 4-3). RXWAKE is a read-only flag, cleared by one of the following:

❑ The transfer of the first byte after the address byte to SCIRXBUF
❑ The reading of SCIRXBUF
❑ An active SW RESET
❑ A system reset

See Section 4.3, *SCI Multiprocessor Communication*, on page 4-8 for details on the SCI multiprocessor address-bit and idle-line communication modes.

Bit 0 **Reserved**. Reads are indeterminate and writes have no effect.

Figure 4–16. SCIRXST Bit Associations


7 6 5 4 3 2 1 0
RX ERROR
RXRDY
BRKDT
FE
OE
PE
RXWAKE
Reserved
RXRDY or BRKDT causes an interrupt if RX/BK INT ENA (SCICTL2.1) = 1
RX ERROR = 1 when any of bits 5 – 2 is a 1 value

4.6.6 Receiver Data Buffer Registers

Received data is transferred from RXSHF to SCIRXEMU and SCIRXBUF. When the transfer is complete, the RXRDY flag (bit SCIRXST.6) is set, indicating that the received data is ready to be read. Both registers contain the same data; they have separate addresses but are not physically separate buffers. The only difference is that reading SCIRXEMU does not clear the RXRDY flag; however, reading SCIRXBUF clears the flag.

Emulation Data Buffer (SCIRXEMU)

For normal SCI data receipt operations, read the data received from SCIRXBUF. SCIRXEMU is used principally by the emulator (EMU) because it can continually read the data received for screen updates without clearing the RXRDY flag. SCIRXEMU is cleared by system reset.

Figure 4–17. SCI Emulation Data Buffer Register (SCIRXEMU) — Address 7056h

7	6	5	4	3	2	1	0
ERXDT7	ERXDT6	ERXDT5	ERXDT4	ERXDT3	ERXDT2	ERXDT1	ERXDT0
R–x	R–x	R–x	R–x	R–x	R–x	R–x	R–x

Note: R = Read access, –n = value after reset (x = indeterminate)

Receiver Data Buffer (SCIRXBUF)

When the current data received is shifted from RXSHF to the receive buffer, flag bit RXRDY is set and the data is ready to be read. If the RX/BK INT ENA bit (SCICTL2.1) is set, this shift also causes an interrupt. When SCIRXBUF is read, the RXRDY flag is reset. SCIRXBUF is cleared by a system reset.

Figure 4–18. SCI Receiver Data Buffer Register (SCIRXBUF) — Address 7057h

7	6	5	4	3	2	1	0
RXDT7	RXDT6	RXDT5	RXDT4	RXDT3	RXDT2	RXDT1	RXDT0
R–x	R–x	R–x	R–x	R–x	R–x	R–x	R–x

Note: R = Read access, –n = value after reset (x = indeterminate)

4.6.7 Transmit Data Buffer Register (SCITXBUF)

Data bits to be transmitted are written to the transmit data buffer register (SCITXBUF). The transfer of data from this register to the TXSHF transmitter shift register sets the TXRDY flag (SCICTL2.7), indicating that SCITXBUF is ready to receive another set of data. If bit TX INT ENA (SCICTL2.0) is set, this data transfer also causes an interrupt. These bits must be right-justified because the left-most bits are ignored for characters less than eight bits long.

Figure 4–19. SCI Transmit Data Buffer Register (SCITXBUF) — Address 7059h

7	6	5	4	3	2	1	0
TXDT7	TXDT6	TXDT5	TXDT4	TXDT3	TXDT2	TXDT1	TXDT0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

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

4.6.8 Port Control Register 2 (SCIPC2)

The SCI port control register 2 (SCIPC2) controls the functions of pins SCITXD and SCIRXD.

Figure 4–20. SCI Port Control Register 2 (SCIPC2) — Address 705Eh

7	6	5	4	3	2	1	0
SCITXD DATA IN	SCITXD DATA OUT	SCITXD FUNCTION	SCITXD DATA DIR	SCIRXD DATA IN	SCIRXD DATA OUT	SCIRXD FUNCTION	SCIRXD DATA DIR
R–x	RW–0	RW–0	RW–0	R–x	RW–0	RW–0	RW–0

Note: R = Read access, W = Write access, –n = value after reset (x = indeterminate)

Bit 7 **SCITXD DATA IN**. Contains the current value of pin SCITXD.

0 = Pin SCITXD value read as 0.
1 = Pin SCITXD value read as 1.

Bit 6 **SCITXD DATA OUT**. Value to be output on pin SCITXD. This bit contains the data to be output on the SCITXD if it is a general-purpose digital-I/O output pin. (For this configuration, clear bit SCIPC2.5 to 0 and set bit SCIPC2.4 to 1.)

Bit 5 **SCITXD FUNCTION**. Defines the function of pin SCITXD.

0 = SCITXD is a general-purpose digital I/O pin.
1 = SCITXD is a the SCI transmit pin.

Bit 4 **SCITXD DATA DIR**. Defines the data direction of pin SCITXD. If SCITXD is configured as general-purpose I/O pin (bit 5 = 0), this bit specifies SCITXD's direction:

0 = SCITXD is a digital input pin.
1 = SCITXD is a digital output pin.

Bit 3 **SCIRXD DATA IN**. Contains current value of pin SCIRXD.

0 = SCIRXD value read as 0.
1 = SCIRXD value read as 1.

Bit 2 **SCIRXD DATA OUT**. Value to be output on pin SCIRXD. This bit contains the data to be output on pin SCIRXD if SCIRXD is a general-purpose digital output pin. For this configuration, you **must** clear bit SCIPC2.1 to 0 (pin is general-purpose) and set bit SCIPC2.0 to 1 (pin is digital output).

0 = A 0 value output on pin SCIRXD.
1 = A 1 value output on pin SCIRXD.

Bit 1	**SCIRXD FUNCTION**. Defines the function of pin SCIRXD. 0 = SCIRXD is a general-purpose digital I/O pin. 1 = SCIRXD is the SCI receive pin.
Bit 0	**SCIRXD DATA DIR**. Defines the data direction of pin SCIRXD. If SCIRXD is configured as general-purpose I/O pin (bit SCIPC2.1 = 0), this bit specifies pin SCIRXD's direction: 0 = SCIRXD is a digital input pin. 1 = SCIRXD is a digital output pin.

4.6.9 Priority Control Register (SCIPRI)

The SCI priority control register (SCIPRI) contains the receiver and transmitter interrupt priority select bits.

Figure 4–21. SCI Priority Control Register (SCIPRI) — Address 705Fh

7	6	5	4	3 – 0
Reserved	SCITX PRIORITY	SCIRX PRIORITY	SCI ESPEN	Reserved
	RW–0	RW–0	RW–0	

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

Bit 7 **Reserved**. Reads are indeterminate and writes have no effect.

Bit 6 **SCITX PRIORITY**. SCI transmitter interrupt priority select. This bit specifies priority level of the SCI transmitter interrupts.

0 = Interrupts are high-priority requests.
1 = Interrupts are low-priority requests.

Bit 5 **SCIRX PRIORITY**. SCI receiver interrupt priority select. This bit specifies the priority level to the SCI receiver interrupts.

0 = Interrupts are high-priority requests.
1 = Interrupts are low-priority requests.

Bit 4 **SCI ESPEN**. SCI emulator suspend enable. This bit has an effect only when debugging a program with the XDS emulator. This bit determines how the SCI operates when the program is suspended.

0 = When the suspended signal goes high, the SCI continues operation until the current transmit and receive functions are complete.
1 = When the suspend signal goes high, the SCI state machine is frozen.

Bits 3 – 0 **Reserved**. Reads are indeterminate and writes have no effect.

4.7 SCI Initialization Example

Example 4–1. Asynchronous Communications Routine

```
;*********************************************************************
; File Name:   TMS320x240 SCI Idle-line Mode Example Code
;
; Description: This program uses the SCI module to implement a simple
;              asynchronous communications routine.  The SCI is
;              initialized to operate in idle-line mode with 8 character
;              bits, 1 stop bit, and odd parity.  The SCI Baud Rate
;              Registers (BRR) are set to transmit and receive data at
;              19200 baud. The SCI generates an interrupt every time a
;              character is loaded into the receive buffer (SCIRXBUF).
;              The interrupt service routine(ISR) reads the receive
;              buffer and determines if the carriage return button, <CR>,
;              has been pressed.  If so, the character string ”Ready” is
;              transmitted. If not, no character string is transmitted.
;*********************************************************************
; SET statements for ’24x devices are device dependent.  The SET
; locations used in this example are typically true for devices with
; only one SCI module.  Consult the device data sheet to determine the
; exact memory map locations of the modules you will be accessing
; (control registers, RAM, ROM).
;
             .include ”c240reg.h”       ; contains a list of SET statements
                                        ; for all registers on TMS320x240.
; The following SET statements for the SCI are contained in c240reg.h
; and are shown explicitly for clarity.
;Serial Communications Interface (SCI) Registers
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
SCICCR       .set   07050h       ;SCI Comms Control Reg
SCICTL1      .set   07051h       ;SCI Control Reg 1
SCIHBAUD     .set   07052h       ;SCI Baud rate control
SCILBAUD     .set   07053h       ;SCI Baud rate control
SCICTL2      .set   07054h       ;SCI Control Reg 2
SCIRXST      .set   07055h       ;SCI Receive status reg
SCIRXEMU     .set   07056h       ;SCI EMU data buffer
SCIRXBUF     .set   07057h       ;SCI Receive data buffer
SCITXBUF     .set   07059h       ;SCI Transmit data buffer
SCIPC2       .set   0705Eh       ;SCI Port control reg2
SCIPRI       .set   0705Fh       ;SCI Priority control reg
;---------------------------------------------------------------------
; Constant definitions
;---------------------------------------------------------------------
LENGTH       .set   00005h       ; length of the data stream to be
                                 ; transmitted
;---------------------------------------------------------------------
; Variable definitions
;---------------------------------------------------------------------
             .bss   DATA_OUT,LENGTH ; Location of LENGTH byte character
                                  ; stream to be transmitted
             .bss   GPR0,1        ;General purpose register.
```

Example 4–1. Asynchronous Communications Routine (Continued)

```
;-----------------------------------------------------------------------
; Macro definitions
;-----------------------------------------------------------------------
KICK_DOG      .macro              ;Watchdog reset macro
              LDP    #00E0h
              SPLK   #055h, WDKEY
              SPLK   #0AAh, WDKEY
              LDP    #0h
              .endm
;-----------------------------------------------------------------------
;Bit codes for Test bit instruction (BIT)
;-----------------------------------------------------------------------
BIT15         .set   0000h        ;Bit Code for 15
BIT14         .set   0001h        ;Bit Code for 14
BIT13         .set   0002h        ;Bit Code for 13
BIT12         .set   0003h        ;Bit Code for 12
BIT11         .set   0004h        ;Bit Code for 11
BIT10         .set   0005h        ;Bit Code for 10
BIT9          .set   0006h        ;Bit Code for 9
BIT8          .set   0007h        ;Bit Code for 8
BIT7          .set   0008h        ;Bit Code for 7
BIT6          .set   0009h        ;Bit Code for 6
BIT5          .set   000Ah        ;Bit Code for 5
BIT4          .set   000Bh        ;Bit Code for 4
BIT3          .set   000Ch        ;Bit Code for 3
BIT2          .set   000Dh        ;Bit Code for 2
BIT1          .set   000Eh        ;Bit Code for 1
BIT0          .set   000Fh        ;Bit Code for 0
;-----------------------------------------------------------------------
; Initialized Transmit Data for Interrupt Service Routine
;-----------------------------------------------------------------------
              .data
TXDATA        .word 0052h         ;Hex equivalent of ASCII character ’R’
              .word 0065h         ;Hex equivalent of ASCII character ’e’
              .word 0061h         ;Hex equivalent of ASCII character ’a’
              .word 0064h         ;Hex equivalent of ASCII character ’d’
              .word 0079h         ;Hex equivalent of ASCII character ’y’
;-----------------------------------------------------------------------
; Vector address declarations
;-----------------------------------------------------------------------
              .sect  ”.vectors”
RSVECT        B      START        ; PM 0       Reset Vector    1
INT1          B      INT1_ISR     ; PM 2       Int level 1     4
INT2          B      PHANTOM      ; PM 4       Int level 2     5
INT3          B      PHANTOM      ; PM 6       Int level 3     6
INT4          B      PHANTOM      ; PM 8       Int level 4     7
INT5          B      PHANTOM      ; PM A       Int level 5     8
INT6          B      PHANTOM      ; PM C       Int level 6     9
RESERVED      B      PHANTOM      ; PM E       (Analysis Int)  10
SW_INT8       B      PHANTOM      ; PM 10      User S/W int    -
SW_INT9       B      PHANTOM      ; PM 12      User S/W int    -
```

Example 4–1. Asynchronous Communications Routine (Continued)

```
SW_INT10      B      PHANTOM      ; PM 14      User S/W int    -
SW_INT11      B      PHANTOM      ; PM 16      User S/W int    -
SW_INT12      B      PHANTOM      ; PM 18      User S/W int    -
SW_INT13      B      PHANTOM      ; PM 1A      User S/W int    -
SW_INT14      B      PHANTOM      ; PM 1C      User S/W int    -
SW_INT15      B      PHANTOM      ; PM 1E      User S/W int    -
SW_INT16      B      PHANTOM      ; PM 20      User S/W int    -
TRAP          B      PHANTOM      ; PM 22      Trap vector     -
NMI           B      PHANTOM      ; PM 24      Non maskable Int3
EMU_TRAP             PHANTOM      ; PM 26      Emulator Trap   2
SW_INT20      B      PHANTOM      ; PM 28      User S/W int    -
SW_INT21      B      PHANTOM      ; PM 2A      User S/W int    -
SW_INT22      B      PHANTOM      ; PM 2C      User S/W int    -
SW_INT23      B      PHANTOM      ; PM 2E      User S/W int    -
;=====================================================================
; M A I N   C O D E  - starts here
;=====================================================================
             .text
START:        SETC   INTM         ;Disable interrupts
              CLRC   SXM          ;Clear Sign Extension Mode
              CLRC   OVM          ;Reset Overflow Mode
              CLRC   CNF          ;Config Block B0 to Data mem.
              LDP    #00E0h
              SPLK   #006Fh, WDCR ;Disable Watchdog if VCCP=5V
              KICK_DOG            ;Reset Watchdog counter
              LDP    #00E0h
              SPLK   #00BBh,CKCR1 ;CLKIN(XTAL)=10MHz,CPUCLK=20MHz
              SPLK   #00C3h,CKCR0 ;CLKMD=PLL Enable,SYSCLK=CPUCLK/2,
              SPLK   #40C0h,SYSCR ;CLKOUT=CPUCLK
              LDP    #0000h
              SPLK   #4h,GPR0
              OUT    GPR0,WSGR    ;Set XMIF to run w/no wait states
;=====================================================================
; Initialize B2 RAM to zero’s.
;=====================================================================
              LAR    AR2,#B2_SADDR; AR2 -> B2 start address
              MAR    *,AR2        ; Set ARP=AR2
              ZAC                 ; Set ACC = 0
              RPT    #1fh         ; Set repeat cntr for 31+1 loops
              SACL   *+           ; Write zeros to B2 RAM
;=====================================================================
; Initialize DATAOUT with data to be transmitted.
;=====================================================================
              LAR    AR2,#B2_SADDR; Reset AR2 -> B2 start address
              LAR    AR1,#DATA_OUT; AR1 -> DATAOUT start address
              RPT    #04h         ; set repeat counter for 4+1 loops
              BLPD   #TXDATA,*+   ; loads B2 with TXDATA
```

Example 4–1. Asynchronous Communications Routine (Continued)

```
;======================================================================
; INITIALIZATION OF INTERRUPT DRIVEN SCI ROUTINE
;======================================================================
SCI_INIT:     LDP    #00E0h
              SPLK   #0037h, SCICCR ;1 stop bit,odd parity,8 char bits,
                                   ;async mode, idle-line protocol
              SPLK   #0013h, SCICTL1 ;Enable TX, RX, internal SCICLK,
                                    ;Disable RX ERR, SLEEP, TXWAKE
              SPLK   #0002h, SCICTL2 ;Enable RX INT,disable TX INT
              SPLK   #0000h, SCIHBAUD
              SPLK   #0040h, SCILBAUD ;Baud Rate=19200 b/s (10 MHz SYSCLK)
              SPLK   #0022h, SCIPC2  ;Enable TXD & RXD pins
              SPLK   #0033h, SCICTL1 ;Relinquish SCI from Reset.
              LAR    AR0, #SCITXBUF  ;Load AR0 with SCI_TX_BUF address
              LAR    AR1, #SCIRXBUF  ;Load AR1 with SCI_RX_BUF address
              LAR    AR2, #B2_SADDR  ;Load AR2 with TX data start address
              LDP    #0
              LACC   IFR            ;Load ACC with Interrupt flags
              SACL   IFR            ;Clear all pending interrupt flags
              SPLK   #0001h,IMR     ;Unmask interrupt level INT1
              CLRC   INTM           ;Enable interrupts
;======================================================================
; Main Program Routine
;======================================================================
MAIN:                              ;Main loop of code begins here
                                   ;insert actual code here
              NOP
              NOP
              NOP
;             ....                 ;insert actual code here
       B      MAIN                 ;The MAIN program loop has completed
                                   ;one pass, branch back to the
                                   ;beginning and continue.
;======================================================================
; I S R  -  INT1_ISR
;
; Description:  The INT1_ISR first determines if the SCI RXINT caused
;               the interrupt.  If so, the SCI Specific ISR reads the
;               character in the RX buffer.  If the character received
;               corresponds to a carriage return, <CR>, the character
;               string ”Ready” is transmitted. If the character
;               received does NOT correspond to a carriage return,
;               <CR>, then the ISR returns to the main program
;               without transmitting a character string.  If the
;               SCI RXINT did not cause an interrupt, then the value
;               ’0x0bad’ is stored in the accumulator and program
;               gets caught in the BAD_INT endless loop.
;======================================================================
```

Example 4–1. Asynchronous Communications Routine (Continued)

```
INT1_ISR:     LDP    #00E0h
              LACL   SYSIVR       ;Load peripheral INT vector address
              SUB    #0006h       ;Subtract RXINT offset from above
              BCND   SCI_ISR,EQ   ;verify RXINT initiated interrup
       B      BAD_INT             ;Else, bad interrupt occurred
SCI_ISR:      MAR    *,AR1        ;Set ARP=AR1
              LACC   *,AR2        ;Load ACC w/RX buffer character
              SUB    #000Dh       ;Check if <CR> has been pressed:
              BCND   XMIT_CHAR, EQ;YES? Transmit data.
       B      ISR_END             ;N0? Return from INT1_ISR.
XMIT_CHAR:    LACC   *+,AR0       ;Load char to be xmitted into ACC
              BCND   ISR_END,EQ   ;Check for Null Character
                                  ;YES? Return from INT1_ISR.
              SACL   *,AR2        ;NO? Load char into xmit buffer.
XMIT_RDY:     BIT    SCICTL2, BIT7;Test TXRDY bit
              BCND   XMIT_RDY, NTC;If TXRDY=0,then repeat loop
       B      XMIT_CHAR           ;else xmit next character
ISR_END:      LAR    AR2, #B2_SADDR ;Reload AR2 w/ TX data start address
              CLRC   INTM         ;Clear INT Mask flag
              RET                 ;Return from INT1_ISR
BAD_INT:      LACC   #0BADh       ;Load ACC with ”bad”
       B      BAD_INT             ;Repeat loop
;=======================================================================
; I S R  –  PHANTOM
;
; Description:      Dummy ISR, used to trap spurious interrupts.
;=======================================================================
PHANTOM:      B      PHANTOM
```

Chapter 5

Serial Peripheral Interface (SPI) Module

This chapter discusses the architecture, functions, and programming of the Serial Peripheral Interface (SPI) Module.

Note: 8-Bit Peripheral

This module is interfaced to the 16-bit peripheral bus as an 8-bit peripheral. Therefore, reads from bits 15–8 are undefined; writes to bits 15–8 have no effect.

Topic **Page**

5.1 SPI Overview

The SPI is a high-speed synchronous serial input/output (I/O) port that allows a serial bit stream of programmed length (one to eight bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI is normally used for communications between the DSP controller and external peripherals or another controller. Typical applications include external I/O or peripheral expansion via devices such as shift registers, display drivers, and analog-to-digital converters (ADCs).

Note: Register Bit Notation

For convenience, references to a bit in a register are abbreviated using the register name followed by a period and the number of the bit. For example, the notation for bit 7 of the SPI emulation buffer register (SPIEMU) is SPIEMU.7.

5.1.1 SPI Physical Description

The 4-pin SPI module, as shown in Figure 5–1, consists of:

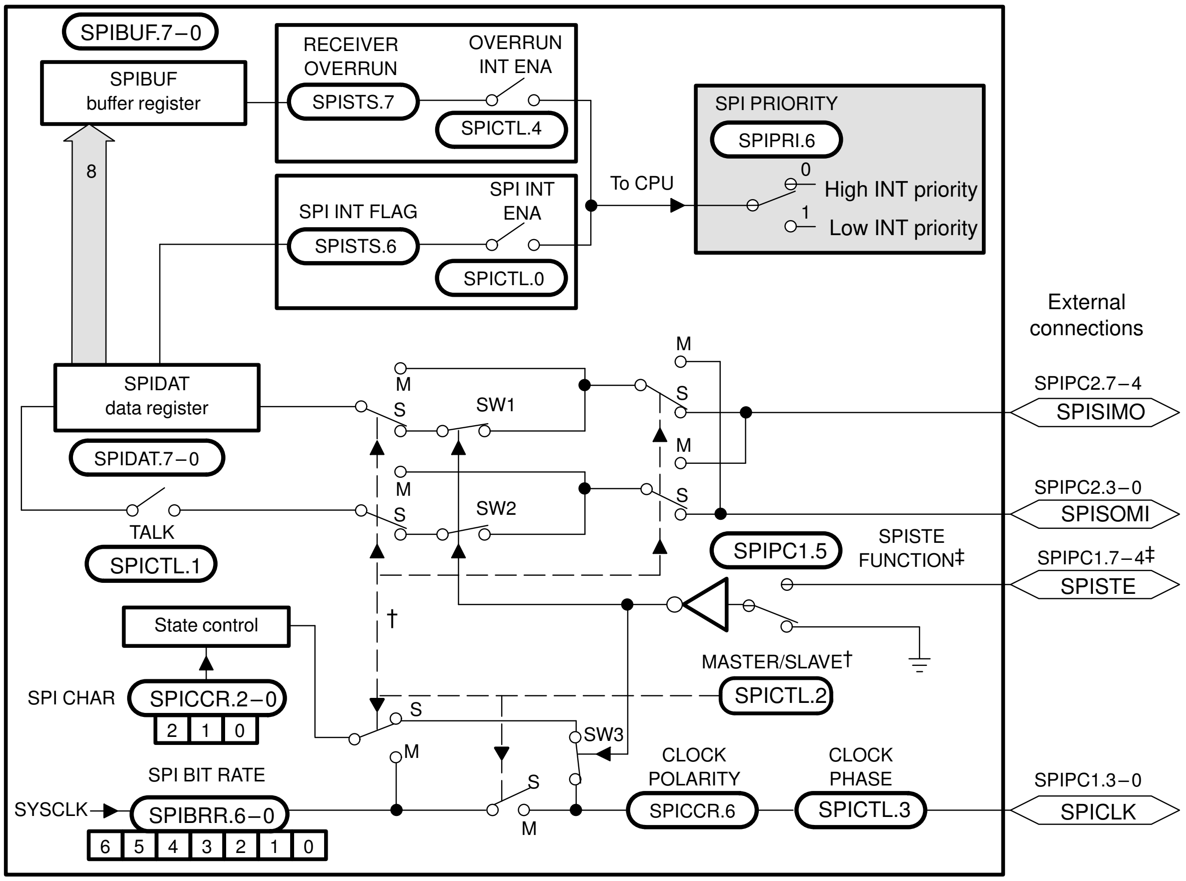
- ❑ Four I/O pins:
  - ■ SPISIMO (SPI slave in, master out) controlled by bits in SPIPC2
  - ■ SPISOMI (SPI slave out, master in) controlled by bits in SPIPC2
  - ■ SPICLK (SPI clock) controlled by bits in SPIPC1
  - ■ SPISTE (SPI strobe) controlled by bits in SPIPC1
- ❑ Master and slave mode operations
- ❑ SPI serial input buffer register (SPIBUF)

  This buffer register contains the data that is received from the network and that is ready for the CPU to read.
- ❑ SPI serial data register (SPIDAT)

  This data shift register serves as the transmit/receive shift register.
- ❑ SPICLK phase and polarity control
- ❑ State control logic
- ❑ Memory-mapped control and status registers

The basic function of the strobe (SPISTE) pin is to act as a transmit enable for the SPI module in slave mode. It stops the shift register so it cannot receive. It also 3-states the SOMI pin. When the SPI is in the master mode, the SPISTE pin always operates as a general-purpose digital I/O pin and can function as the SPI slave module select line. See subsection 5.3.8 on page 5-28 and subsection 5.3.9 on page 5-30 for additional information.

Figure 5–1. Four-Pin SPI Module Block Diagram



† The diagram is shown in slave mode.

‡ The SPISTE pin is shown as being disabled, meaning the data can be transmitted or received in this mode. Note that switches SW1, SW2, and SW3 are closed in this configuration.

5.1.2 SPI Control Registers

Ten registers inside the SPI module (listed in Table 5–1) control the SPI operations:

- ❑ SPICCR (SPI configuration control register). Contains control bits used for SPI configuration:
  - ■ SPI module software reset
  - ■ SPICLK polarity selection
  - ■ Three SPI character-length control bits
- ❑ SPICTL (SPI operation control register). Contains control bits for data transmission:
  - ■ Two SPI interrupt enable bits
  - ■ SPICLK phase selection
  - ■ Operational mode (master/slave)
  - ■ Data transmission enable
- ❑ SPISTS (SPI status register). Contains two receiver buffer status bits:
  - ■ RECEIVER OVERRUN
  - ■ SPI INT FLAG
- ❑ SPIBRR (SPI baud rate register). Contains seven bits that determine the bit transfer rate.
- ❑ SPIEMU (SPI emulation buffer register). Contains the received data. Supports correct emulation. The SPIBUF should be used for normal operation.
- ❑ SPIBUF (SPI receive buffer — the serial input buffer register). Contains the received data.
- ❑ SPIDAT (SPI data register). Contains data transmitted by the SPI, acting as the transmit/receive shift register. Data written to SPIDAT is shifted out on subsequent SPICLK cycles. For every bit shifted out of the SPI, a bit is shifted into the other end of the shift register.
- ❑ SPIPC1 (SPI port control register 1). Contains control bits to select the functions of the SPISTE pin and the SPICLK pin.
- ❑ SPIPC2 (SPI port control register 2). Contains control bits to select the functions of the SPISIMO and SPISOMI pins.
- ❑ SPIPRI (SPI priority register). Contains two bits that specify interrupt priority and determine SPI operation on the XDS emulator during program suspensions.

Table 5–1. Addresses of SPI Control Registers

			Described in	
Address	**Register**	**Name**	**Subsection**	**Page**
7040h	SPICCR	SPI configuration control register	5.3.1	5-18
7041h	SPICTL	SPI operation control register	5.3.2	5-20
7042h	SPISTS	SPI status register	5.3.3	5-23
7043h		Reserved		
7044h	SPIBRR	SPI baud rate register	5.3.4	5-24
7045h		Reserved		
7046h	SPIEMU	SPI emulation buffer register	5.3.5	5-25
7047h	SPIBUF	SPI serial input buffer register	5.3.6	5-26
7048h		Reserved		
7049h	SPIDAT	SPI serial data register	5.3.7	5-27
704Ah		Reserved		
704Bh		Reserved		
704Ch		Reserved		
704Dh	SPIPC1	SPI port control register 1	5.3.8	5-28
704Eh	SPIPC2	SPI port control register 2	5.3.9	5-30
704Fh	SPIPRI	SPI priority control register	5.3.10	5-32

5.2 SPI Operation

This section describes the operation of the SPI. Included are explanations of the operation modes, interrupts, data format, clock sources, and initialization. Typical timing diagrams for data transfers are given.

Note: Register Bit Naming Protocol

Reference to a bit in a register is abbreviated using the register acronym followed by a period and the bit number. For example, the MASTER/SLAVE bit of the SPI operation control register (SPICTL) is SPICTL.2.

5.2.1 Introduction to Operation

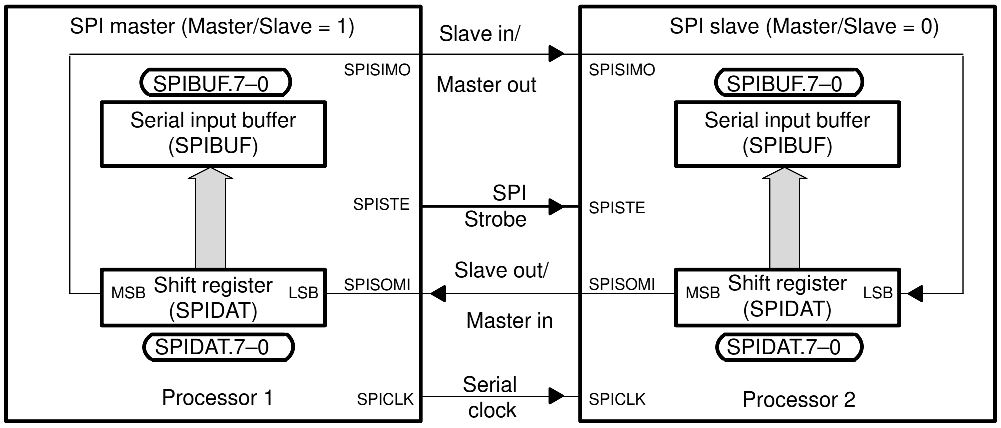
Figure 5–2 show typical connections of the SPI for communications between two controllers: a master and a slave.

The master initiates data transfer by sending the SPICLK signal. For both the slave and the master, data is shifted out of the shift registers on one edge of the SPICLK and latched into the shift register on the opposite SPICLK clock edge. If the CLOCK PHASE (bit SPICTL.3) bit is active, data is transmitted and received a half-cycle before the SPICLK transition (see subsection 5.2.5, *Baud Rate and Clocking Schemes*, on page 5-11). As a result, both controllers send and receive data simultaneously. The application software determines whether the data is meaningful or dummy data. There are three possible methods for data transmission:

- ❑ Master sends data, and slave sends dummy data.
- ❑ Master sends data, and slave sends data.
- ❑ Master sends dummy data, and slave sends data.

The master can initiate data transfer at any time because it controls the SPICLK signal. The software, however, determines how the master detects when the slave is ready to broadcast data.

Figure 5–2. SPI Master/Slave Connection (4-Pin Option)



5.2.2 SPI Module Slave and Master Operation Modes

The SPI can operate in master or slave mode. The MASTER/SLAVE bit (SPICTL.2) selects the operating mode and the source of the SPICLK signal.

Master Mode

In the master mode (MASTER/SLAVE = 1), the SPI provides the serial clock on the SPICLK pin for the entire serial communications network. Data is output on the SPISIMO pin and latched from the SPISOMI pin.

The SPIBRR determines both the transmit and receive bit transfer rate for the network. The SPIBRR can select 125 different data transfer rates.

Data written to SPIDAT initiates data transmission on the SPISIMO pin, MSB (most significant bit) first. Simultaneously, received data is shifted through the SPISOMI pin into the LSB (least significant bit) of SPIDAT. When the selected number of bits has been transmitted, the data is transferred, MSB first, to the SPIBUF (buffered receiver) for the CPU to read. Data is stored right-justified in SPIBUF.

When the specified number of data bits has been shifted through SPIDAT, the following events occur:

- ❑ SPI INT FLAG bit (SPISTS.6) is set to 1
- ❑ SPIDAT contents transfer to SPIBUF
- ❑ If the SPI INT ENA bit (SPICTL.0) is set to 1, an interrupt is asserted

In the master mode, the SPISTE pin will always operate as a general-purpose I/O pin, regardless of the value of the SPISTE FUNCTION bit (SPIPC1.5). In a typical application, the SPISTE pin could serve as a chip enable pin for slave SPI devices. (Drive this slave select pin low before transmitting master data to the slave device, and drive this pin high again after transmitting the master data.)

Slave Mode

In the slave mode (MASTER/SLAVE = 0), data shifts out on the SPISOMI pin and in on the SPISIMO pin. The SPICLK pin is used as the input for the serial shift clock, which is supplied from the external network master. The transfer rate is defined by this clock. The SPICLK input frequency should be no greater than the SYSCLK (system clock of the device) frequency divided by 8.

Data written to SPIDAT is transmitted to the network when the SPICLK signal is received from the network master. To receive data, the SPI waits for the network master to send the SPICLK signal and then shifts the data on the SPISIMO pin into SPIDAT. If data is to be transmitted by the slave simultaneously, it must be written to the SPIDAT before the beginning of the SPICLK signal.

When the TALK bit (SPICTL.1) is cleared, data transmission is disabled, and the output line (SPISOMI) is put into the high-impedance state. This allows many slave devices to be tied together on the network, but only one slave at a time is allowed to talk.

In the slave mode, the SPISTE pin operates as a general-purpose I/O pin if the value of the SPISTE FUNCTION bit (SPIPC1.5) is cleared (0). The SPISTE pin operates as the slave select pin if SPIPC1.5 is set to 1. When the SPISTE pin is operating as the slave select pin, an active low signal on the SPISTE pin allows the slave SPI to transfer data to the serial data line; an inactive high signal causes the slave SPI's serial shift register to stop and its serial output pin to be put into the high-impedance state. This allows many slave devices to be tied together on the network, although only one slave device will be selected at a time.

5.2.3 SPI Interrupts

Four control bits are used to initialize the SPI's interrupts:

- ❏ SPI INT ENA bit (SPICTL.0)
- ❏ SPI INT FLAG bit (SPISTS.6)
- ❏ OVERRUN INT ENA bit (SPICTL.4)
- ❏ RECEIVER OVERRUN flag bit (SPISTS.7)

SPI INT ENA Bit (SPICTL.0)

When the SPI interrupt enable bit is set and an interrupt condition occurs, the corresponding interrupt is asserted.

0	=	Disable SPI interrupts
1	=	Enable SPI interrupts

SPI INT FLAG Bit (SPISTS.6)

This status flag indicates that a character has been placed in the SPI receiver buffer and is ready to be read.

When a complete character has been shifted into or out of SPIDAT, the SPI INT FLAG bit (SPISTS.6) is set, and an interrupt is generated if enabled by the SPI INT ENA bit. The interrupt flag remains set until it is cleared by one of the following four events:

- ❑ The CPU reads the SPIBUF (reading the SPIEMU does not clear the SPI INT FLAG)
- ❑ The CPU enters the OSC Power Down or PLL Power Down mode with an IDLE instruction
- ❑ Software sets the SPI SW RESET bit (SPICCR.7 on page 5-18)
- ❑ A system reset occurs

To avoid the generation of another interrupt, an interrupt request must be explicitly cleared by one of the four methods listed above. An interrupt request can be temporarily disabled by clearing the SPI INT ENA bit. Unless the SPI INT FLAG bit is cleared, however, the interrupt request will be reasserted when the SPI INT ENA bit is again set to 1.

When the SPI INT FLAG bit is set, a character has been placed into the SPIBUF and is ready to be read. If the CPU does not read the character by the time the next complete character has been received, the new character is written into SPIBUF, and the RECEIVER OVERRUN flag bit (SPISTS.7) is set.

OVERRUN INT ENA Bit (SPICTL.4)

Setting the overrun interrupt enable bit allows the assertion of an interrupt whenever the RECEIVER OVERRUN flag bit (SPISTS.7) is set by hardware. Interrupts generated by SPISTS.7 and by the SPI INT FLAG (SPISTS.6) bit share the same interrupt vector.

0	=	Disable RECEIVER OVERRUN flag bit interrupts
1	=	Enable RECEIVER OVERRUN flag bit interrupts

RECEIVER OVERRUN Flag Bit (SPISTS.7)

The RECEIVER OVERRUN flag bit is set whenever a new character has been received and loaded into the SPIBUF before the previously received character has been read from the SPIBUF. The RECEIVER OVERRUN flag bit must be cleared by software.

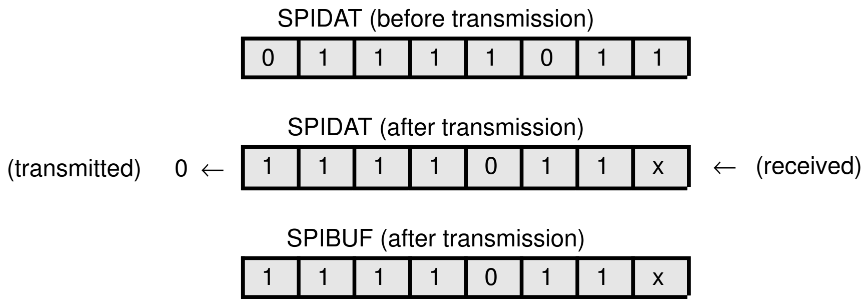
5.2.4 Data Format

Three bits (SPICCR.2-0) specify the number of bits (one to eight) in the data character. This information directs the state control logic to count the number of bits received or transmitted to determine when a complete character has been processed. The following apply to characters with fewer than eight bits:

- ❑ Data must be left justified when written to SPIDAT.
- ❑ Data read back from SPIBUF is right justified.
- ❑ SPIBUF contains the most recently received character, right justified, plus any bits that remain from previous transmission(s) that have been shifted to the left (shown in Example 5–1).

Example 5–1. Transmission of Bit from SPIBUF

Conditions:
1. Transmission character length = 1 bit (specified in bits SPICCR.2-0)
2. The current value of SPIDAT = 07Bh


SPIDAT (before transmission)
0 1 1 1 1 0 1 1
SPIDAT (after transmission)
(transmitted) 0 ←
1 1 1 1 0 1 1 x
← (received)
SPIBUF (after transmission)
1 1 1 1 0 1 1 x

Note: x = 1 if SPISOMI data is high; x = 0 if SPISOMI data is low; master mode is assumed.

5.2.5 Baud Rate and Clocking Schemes

The SPI module supports 125 different baud rates and four different clock schemes. Depending on whether the SPI clock is in slave or master mode, the SPICLK pin can receive an external SPI clock signal or provide the SPI clock signal, respectively.

- ❑ In the slave mode, the SPI clock is received on the SPICLK pin from the external source and can be no greater than the SYSCLK frequency divided by 8.
- ❑ In the master mode, the SPI clock is generated by the SPI and is output on the SPICLK pin.

Baud Rate Determination

Equation 5–1 shows how to determine the SPI baud rates.

Equation 5–1. SPI Baud-Rate Calculations

- ■ For SPIBRR = 3 to 127:

$$SPI\ Baud\ Rate = \frac{SYSCLK}{(SPIBRR + 1)}$$

- ■ For SPIBRR = 0, 1, or 2:

$$SPI\ Baud\ Rate = \frac{SYSCLK}{4}$$

where:

SYSCLK = System clock frequency of the device
SPIBRR = Contents of the SPIBRR in the master SPI device

To determine what value to load into SPIBRR, you must know the device system clock (SYSCLK) frequency (which is device/user-specific) and the baud rate at which you will be operating.

Example 5–2 shows how to determine the maximum baud rate at which a ’C24x can communicate. Assume that SYSCLK = 10 MHz.

Example 5–2. Maximum Baud-Rate Calculation

$$SPI\ Baud\ Rate = \frac{SYSCLK}{(SPIBRR\ +\ 1)}$$

$$= \frac{10\ \times 10^6}{(SPIBBR\ +\ 1)}$$

$$= \frac{10\ \times 10^6}{(3\ +\ 1)}$$

$$= \frac{10\ \times 10^6}{4}$$

$$= 2.5\ \times\ 10^6 = 2.5\ \text{Mbps}$$

The maximum master SPI baud rate would be 2.5 Mbps. However, the SPI slave baud data has a maximum speed of SYSCLK/8.

SPI Clocking Schemes

The CLOCK POLARITY (SPICCR.6 on page 5-18) and CLOCK PHASE (SPICTL.3) bits control four different clocking schemes on the SPICLK pin. The CLOCK POLARITY bit selects the active edge of the clock, either rising or falling. The CLOCK PHASE bit selects a half-cycle delay of the clock. The four different clocking schemes are as follows:

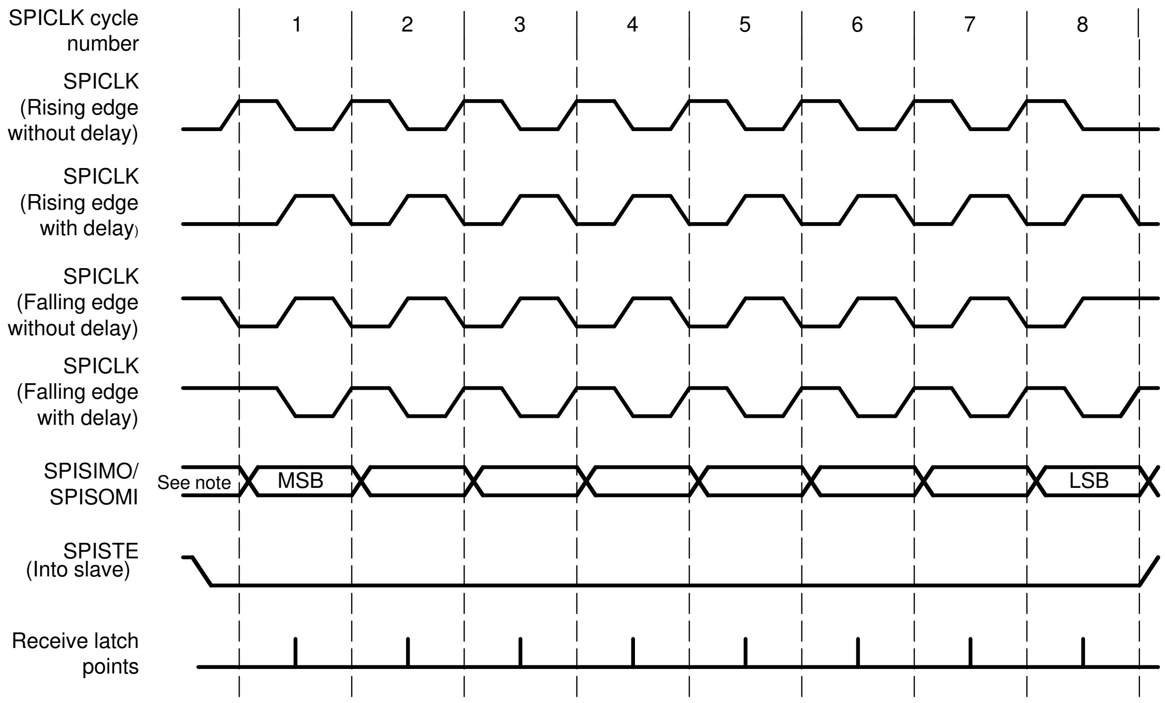
- **Falling Edge Without Delay**. The SPI transmits data on the falling edge of the SPICLK and receive data on the rising edge of the SPICLK.
- **Falling Edge With Delay**. The SPI transmits data one half-cycle ahead of the falling edge of the SPICLK signal and receive data on the falling edge of the SPICLK signal.
- **Rising Edge Without Delay**. The SPI transmits data on the rising edge of the SPICLK signal and receive data on the falling edge of the SPICLK signal.
- **Rising Edge With Delay**. The SPI transmits data one half-cycle ahead of the rising edge of the SPICLK signal and receives data on the rising edge of the SPICLK signal.

The selection procedure for the SPI clocking scheme is shown in Table 5–2. Examples of these four clocking schemes relative to transmitted and received data are shown in Figure 5–3.

Table 5–2. SPI Clocking Scheme Selection Guide

SPICLK Scheme	CLOCK POLARITY (SPICCR.6)	CLOCK PHASE (SPICTL.3)
Rising edge without delay	0	0
Rising edge with delay	0	1
Falling edge without delay	1	0
Falling edge with delay	1	1

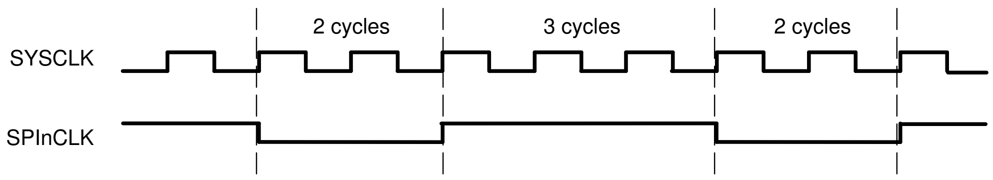
Figure 5–3. SPICLK Signal Options


SPICLK cycle number
1
2
3
4
5
6
7
8
SPICLK (Rising edge without delay)
SPICLK (Rising edge with delay)
SPICLK (Falling edge without delay)
SPICLK (Falling edge with delay)
SPISIMO/ SPISOMI
See note
MSB
LSB
SPISTE (Into slave)
Receive latch points

Note: Previous data bit

For the SPI, the SPICLK symmetry is retained only when the result of (SPIBRR + 1) is an even value. When (SPIBRR + 1) is an odd value and SPIBRR is greater than 3, the SPICLK becomes asymmetrical. The low pulse of the SPICLK is one SYSCLK longer than the high pulse when the CLOCK POLARITY bit is clear (0) as shown in Figure 5–4. When the CLOCK POLARITY bit set to 1, the high pulse of the SPICLK is one SYSCLK longer than the low pulse as shown in Figure 5–4.

Figure 5–4. SPI: SPICLK-SYSCLK Characteristic when (BRR + 1) is Odd, BRR > 3, and CLOCK POLARITY = 1



5.2.6 Initialization Upon Reset

A system reset forces the SPI peripheral module into the following default configuration:

❑ The unit is configured as a slave module (MASTER/SLAVE = 0).
❑ The transmit capability is disabled (TALK = 0).
❑ Data is latched at the input on the falling edge of the SPICLK signal.
❑ Character length is assumed to be one bit.
❑ The SPI interrupts are disabled.
❑ Data in SPIDAT is reset to 00h.
❑ Pin functions are selected as general-purpose inputs.

To change this SPI configuration:

1) Set the SPI SW RESET bit (SPICCR.7 on page 5-18) to 1.

2) Initialize the SPI configuration, format, baud rate, and pin functions as desired.

3) Clear the SPI SW RESET bit.

Note: Proper SPI Initialization Using the SPI SW RESET Bit

To prevent unwanted and unforeseen events from occurring during or as a result of initialization changes, set the SPI SW RESET bit (SPICCR.7 on page 5-18) before making initialization changes and then clear this bit after the initialization is complete.

4) Write to SPIDAT (this initiates the communication process in the master).

5) Read SPIBUF after the data transmission has completed (SPISTS.6 = 1) to determine what data was received.

5.2.7 Data Transfer Example

The two timing diagrams, Figure 5–5 and Figure 5–6, illustrate an SPI data transfer between two devices using a character length of five bits with the SPICLK being symmetrical.

The timing diagram with SPICLK unsymmetrical (Figure 5–4 on page 5-14) shares similar characterizations with Figure 5–5 and Figure 5–6 with the exception of the data transfer being one SYSCLK cycle longer per bit during the low pulse (CLOCK POLARITY = 0) or during the high pulse (CLOCK POLARITY = 1) of the SPICLK.

Figure 5–5. Signals Connecting to Master Processor

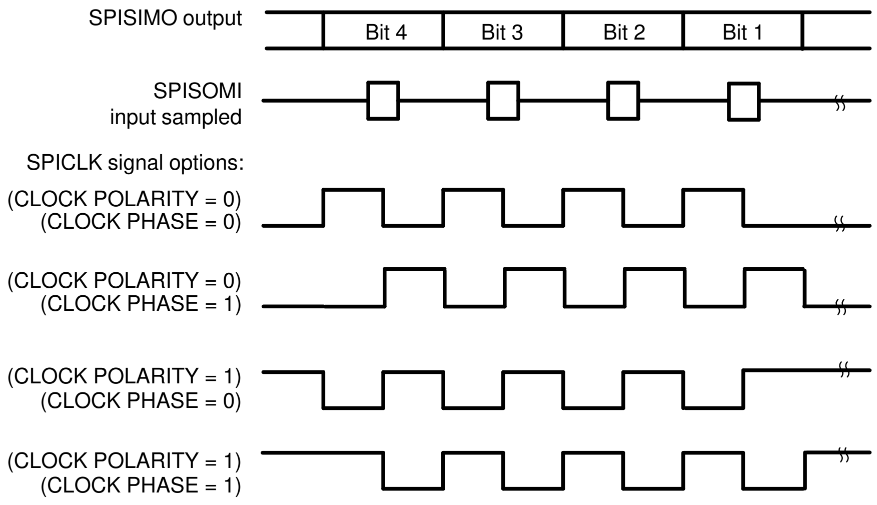
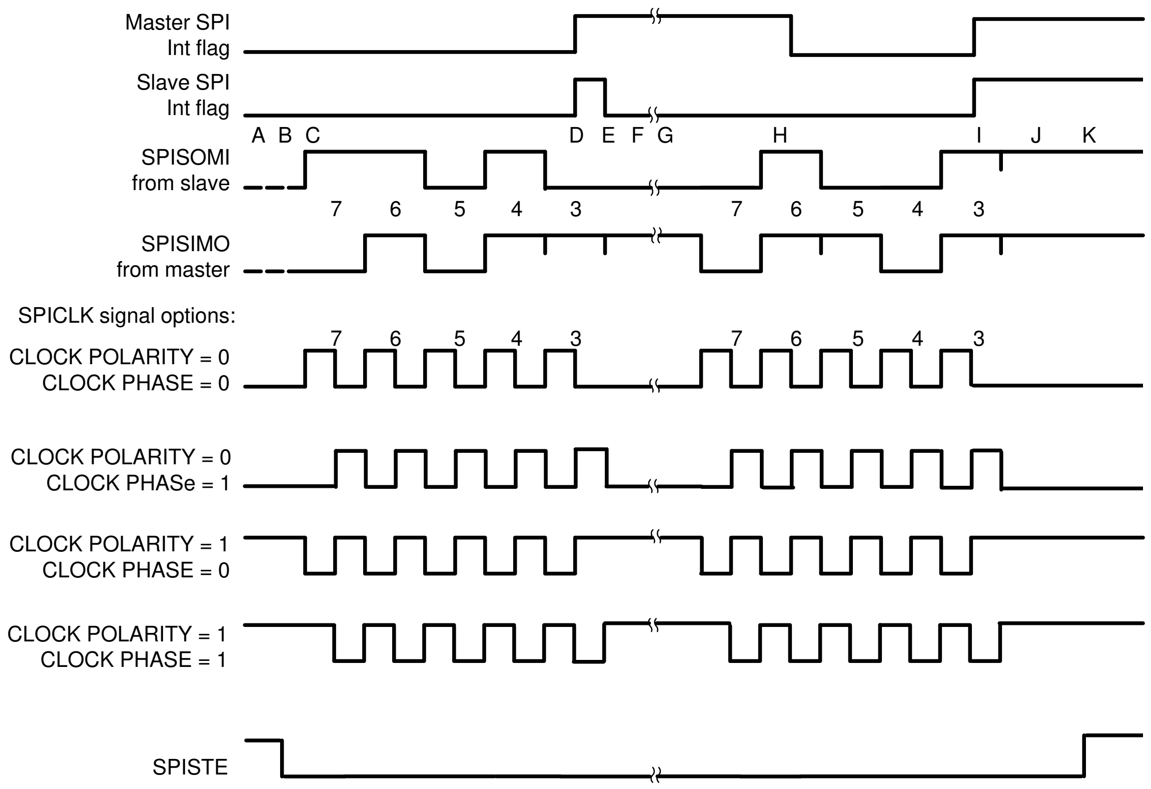

SPISIMO output
Bit 4
Bit 3
Bit 2
Bit 1
SPISOMI input sampled
SPICLK signal options:
(CLOCK POLARITY = 0) (CLOCK PHASE = 0)
(CLOCK POLARITY = 0) (CLOCK PHASE = 1)
(CLOCK POLARITY = 1) (CLOCK PHASE = 0)
(CLOCK POLARITY = 1) (CLOCK PHASE = 1)

Figure 5–6. Five Bits per Character



A. Slave writes 0D0h to SPIDAT and waits for the master to shift out the data.
B. Master sets the slave SPISTE signal low (active).
C. Master writes 058h to SPIDAT, which starts the transmission procedure.
D. First byte is finished and sets the interrupt flags.
E. Slave reads 0Bh from its SPIBUF (right justified).
F Slave writes 04Ch to SPIDAT and waits for the master to shift out the data.
G. Master writes 06Ch to SPIDAT, which starts the transmission procedure.
H. Master reads 01Ah from the SPIBUF (right justified).
I. Second byte is finished and sets the interrupt flags.
J. Master reads 89h and the slave reads 8Dh from their respective SPIBUF. After the user’s software masks-off the unused bits, the master receives 09h and the slave receives 0Dh.
K. Master clears the slave SPISTE signal high (inactive).

5.3 SPI Control Registers

The SPI is controlled and accessed through registers in the control register file. These registers are shown in Figure 5–7 and described in the following subsections.

Figure 5–7. SPI Control Registers

		Bit number							
Address	**Register**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
7040h	SPICCR	SPI SW RESET	CLOCK POLARITY	Reserved			SPI CHAR2	SPI CHAR1	SPI CHAR0
7041h	SPICTL	Reserved			OVERRUN INT ENA	CLOCK PHASE	MASTER/ SLAVE	TALK	SPI INT ENA
7042h	SPISTS	RECEIVER OVERRUN	SPI INT FLAG	Reserved					
7043h	—	Reserved							
7044h	SPIBRR	Reserved	SPI BIT RATE 6	SPI BIT RATE 5	SPI BIT RATE 4	SPI BIT RATE 3	SPI BIT RATE 2	SPI BIT RATE 1	SPI BIT RATE 0
7045h	—	Reserved							
7046h	SPIEMU	ERCVD7	ERCVD6	ERCVD5	ERCVD4	ERCVD3	ERCVD2	ERCVD1	ERCVD0
7047h	SPIBUF	RCVD7	RCVD6	RCVD5	RCVD4	RCVD3	RCVD2	RCVD1	RCVD0
7048h	—	Reserved							
7049h	SPIDAT	SDAT7	SDAT6	SDAT5	SDAT4	SDAT3	SDAT2	SDAT1	SDAT0
704Ah	—	Reserved							
704Bh	—	Reserved							
704Ch	—	Reserved							
704Dh	SPIPC1	SPISTE DATA IN	SPISTE DATA OUT	SPISTE FUNCTION	SPISTE DATA DIR	SPICLK DATA IN	SPICLK DATA OUT	SPICLK FUNCTION	SPICLK DATA DIR
704Eh	SPIPC2	SPISIMO DATA IN	SPISIMO DATA OUT	SPISIMO FUNCTION	SPISIMO DATA DIR	SPISOMI DATA IN	SPISOMI DATA OUT	SPISOMI FUNCTION	SPISOMI DATA DIR
704Fh	SPIPRI	Reserved	SPI PRIORITY	SPI ESPEN	Reserved				

5.3.1 SPI Configuration Control Register (SPICCR)

The SPICCR controls the setup of the SPI for operation.

Figure 5–8. SPI Configuration Control Register (SPICCR) — Address 7040h

7	6	5 – 3	2	1	0
SPI SW RESET	CLOCK POLARITY	Reserved	SPI CHAR2	SPI CHAR1	SPI CHAR0
RW–0	RW–0		RW–0	RW–0	RW–0

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

Bit 7 **SPI SW RESET**. SPI Software Reset. When changing configuration, you should set this bit before the changes and clear this bit before resuming operation. (See subsection 5.2.6, *Initialization Upon Reset,* on page 5-14.)

1 = Initializes the SPI operating flags to the reset condition.

Specifically, the RECEIVER OVERRUN flag bit (SPISTS.7) and the SPI INT FLAG (SPISTS.6) bit are cleared. The SPI configuration remains unchanged. If the module is operating as a master, the SPICLK signal output returns to its inactive level.

0 = SPI is ready to transmit or receive the next character.

When the SPI SW RESET bit is a 1, a character written to the transmitter will not be shifted out when this bit clears. A new character must be written to the serial data register.

Bit 6 **CLOCK POLARITY**. Shift Clock Polarity. This bit controls the polarity of the SPICLK signal. CLOCK POLARITY and CLOCK PHASE (SPICTL.3) control four clocking schemes on the SPICLK pin. See subsubsection 5.2.5, *Baud Rate and Clocking Schemes* on page 5-11, and Figure 5–10 on page 5-22.

0 = Inactive level is low

The data input and output edges depend on the value of the CLOCK PHASE (SPICTL.3) bit as follows:

- CLOCK PHASE = 0: Data is output on the rising edge of the SPICLK signal; input data is latched on the falling edge of the SPICLK signal.
- CLOCK PHASE = 1: Data is output one half-cycle before the first rising edge of the SPICLK signal and on subsequent falling edges of the SPICLK signal; input data is latched on the rising edge of the SPICLK signal.

1 = Inactive level is high

The data input and output edges depend on the value of the CLOCK PHASE (SPICTL.3) bit as follows:

- CLOCK PHASE = 0: Data is output on the falling edge of the SPICLK signal; input data is latched on the rising edge of the SPICLK signal.
- CLOCK PHASE = 1: Data is output one half-cycle before the first falling edge of the SPICLK signal and on subsequent rising edges of the SPICLK signal; input data is latched on the falling edge of the SPICLK signal.

Bits 5–3 **Reserved**. Reads are indeterminate and writes have no effect.

Bits 2–0 **SPI CHAR2–SPI CHAR0**. Character Length Control Bits 2–0. These three bits determine the number of bits to be shifted in or out as a single character during one shift sequence. Table 5–3 lists the character length selected by the bit values.

Table 5–3. Character Length Control Bit Values

SPI CHAR2	SPI CHAR1	SPI CHAR0	Character Length
0	0	0	1
0	0	1	2
0	1	0	3
0	1	1	4
1	0	0	5
1	0	1	6
1	1	0	7
1	1	1	8

5.3.2 SPI Operation Control Register (SPICTL)

The SPICTL operation control register controls data transmission, the SPIs ability to generate interrupts, the SPICLK phase, and the operational mode (slave or master).

Figure 5–9. SPI Operation Control Register (SPICTL) — Address 7041h

7 – 5	4	3	2	1	0
Reserved	OVERRUN INT ENA	CLOCK PHASE	MASTER/ SLAVE	TALK	SPI INT ENA
	RW–0	RW–0	RW–0	RW–0	RW–0

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

Bits 7–5 **Reserved**. Reads are indeterminate and writes have no effect.

Bit 4 **OVERRUN INT ENA**. Overrun Interrupt Enable. Setting this bit (OVERRUN INTERRUPT ENABLE) causes an interrupt to be generated when the RECEIVER OVERRUN flag bit (SPISTS.7 on page 5-23) is set by hardware. Interrupts generated by the RECEIVER OVERRUN flag bit and the SPI INT FLAG bit (SPISTS.6) share the same interrupt vector.

0 = Disable RECEIVER OVERRUN flag bit (SPISTS.7) interrupts
1 = Enable RECEIVER OVERRUN flag bit (SPISTS.7) interrupts

Bit 3 **CLOCK PHASE**. SPI Clock Phase Select. This bit controls the phase of the SPICLK signal.

0 = Normal SPI clocking scheme, depending on the CLOCK POLARITY bit (SPICCR.6 on page 5-18)
1 = SPICLK signal delayed by one half-cycle, polarity determined by the CLOCK POLARITY bit

CLOCK PHASE and CLOCK POLARITY (SPICCR.6 on page 5-18) bits make four different clocking schemes possible. See subsection 5.2.5, *Baud Rate and Clocking Schemes* on page 5-11, subsection 5.3.1, *SPI Configuration Control Register (SPICCR)* on page 5-18, and Figure 5–10 on page 5-22. When operating with CLOCK PHASE high, the SPI (master or slave) makes the first bit of data available after SPIDAT is written and before the first edge of the SPICLK signal, regardless of which SPI mode is being used.

Bit 2 **MASTER/SLAVE**. SPI Network Mode Control. This bit determines whether the SPI is a network master or slave. During reset initialization, the SPI is automatically configured as a network slave.

0 = SPI configured as a slave
1 = SPI configured as a master

Bit 1 **TALK**. Master/Slave Transmit Enable. The TALK bit can disable data transmission (master or slave) by placing the serial data output in the high-impedance state. If this bit is disabled during a transmission, the transmit shift register continues to operate until the previous character is shifted out. When the TALK bit is disabled, the SPI is still able to receive characters and update the status flags. TALK is cleared (disabled) by a system reset.

0 = Disables transmission:

Slave mode operation: If not previously configured as a general-purpose I/O pin, the SPISOMI pin will be put in the high-impedance state.

Master mode operation: If not previously configured as a general-purpose I/O pin, the SPISIMO pin will be put in the high-impedance state.

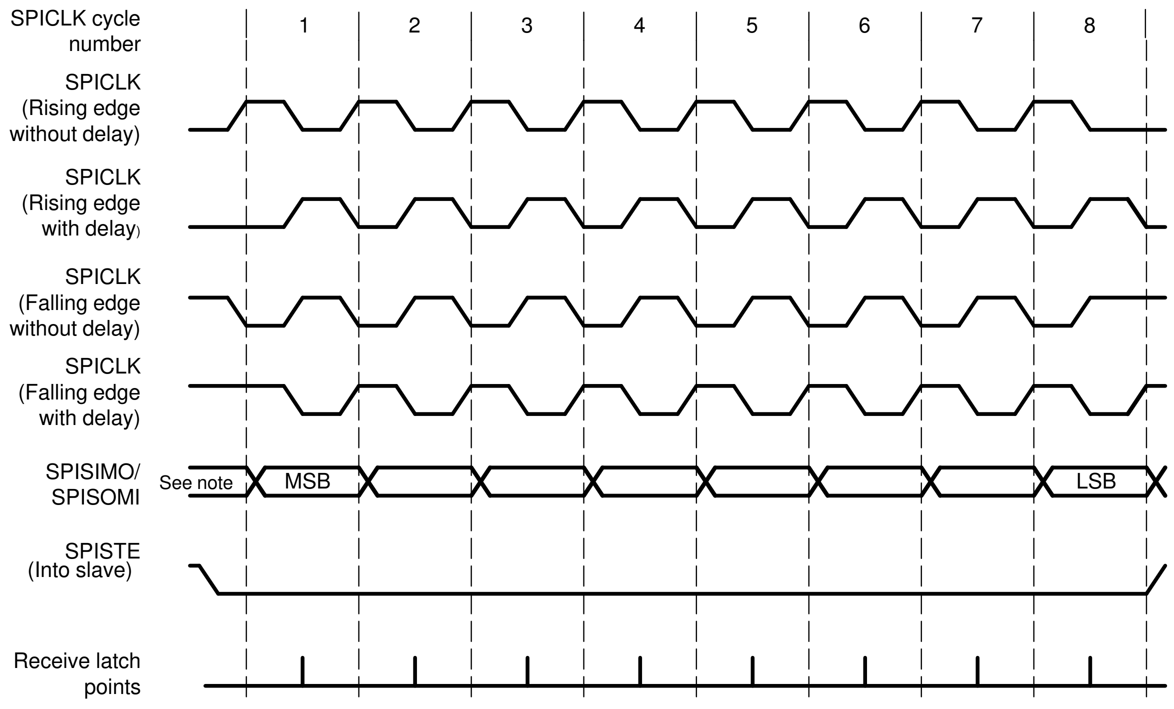
1 = Enables transmission

For the 4-pin option, ensure to enable the receiver's SPISTB input pin.

Bit 0 **SPI INT ENA**. SPI Interrupt Enable. This bit controls the SPI's ability to generate an interrupt. The SPI INT FLAG bit (SPISTS.6) is unaffected by this bit.

0 = Disables interrupt
1 = Enables interrupt

Figure 5–10. SPICLK Signal Options


SPICLK cycle number
1
2
3
4
5
6
7
8
SPICLK (Rising edge without delay)
SPICLK (Rising edge with delay)
SPICLK (Falling edge without delay)
SPICLK (Falling edge with delay)
SPISIMO/ SPISOMI
See note
MSB
LSB
SPISTE (Into slave)
Receive latch points

Note: Previous data bit

5.3.3 SPI Status Register (SPISTS)

The SPISTS contains the receive buffer status bits.

Figure 5–11.SPI Status Register (SPISTS) — Address 7042h

7	6	5 – 0
RECEIVER OVERRUN	SPI INT FLAG	Reserved
RC–0	R–0	

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

Bit 7 **RECEIVER OVERRUN**. SPI Receiver Overrun Flag. This bit is a read/clear only flag. The SPI hardware sets this bit when a receive or transmit operation completes before the previous character has been read from the buffer. The bit indicates that the last received character has been overwritten and therefore lost. The SPI requests one interrupt sequence each time this bit is set if the OVERRUN INT ENA bit (SPICTL.4 on page 5-20) is set high. The bit is cleared in one of three ways:

- ❑ Writing a 0 to this bit
- ❑ Writing a 1 to SPI SW RESET (SPICCR.7 on page 5-18)
- ❑ Resetting the system

If the OVERRUN INT ENA bit (SPICTL.4) is set, the SPI requests only one interrupt each time an overrun condition occurs. In other words, if the RECEIVER OVERRUN flag bit is left set (not cleared) by the interrupt service routine, another overrun interrupt will not be immediately reentered when the interrupt service routine is exited. An interrupt is requested each time an overrun condition occurs if the OVERRUN INT ENA bit is enabled, regardless of the previous condition of the RECEIVER OVERRUN flag bit.

However, the RECEIVER OVERRUN flag bit should be cleared during the interrupt service routine because the RECEIVER OVERRUN flag bit and SPI INT FLAG bit share the same interrupt vector. This will alleviate any possible doubt as to the source of the interrupt when the next byte is received.

Bit 6 **SPI INT FLAG**. SPI Interrupt Flag. SPI INT FLAG is a read-only flag. The SPI hardware sets this bit to indicate that it has completed sending or receiving the last bit and is ready to be serviced. The received character is placed in the receiver buffer at the same time this bit is set. This flag causes an interrupt to be requested if the SPI INT ENA bit (SPICTL.0 on page 5-21) is set. This bit is cleared in one of three ways:

- ❑ Reading SPIBUF
- ❑ Writing a 1 to SPI SW RESET (SPICCR.7 on page 5-18)
- ❑ Resetting the system

Bits 5–0 **Reserved**. Reads are indeterminate and writes have no effect.

5.3.4 SPI Baud Rate Register (SPIBRR)

The SPIBRR contains the bits used for baud-rate calculation.

Figure 5–12. SPI Baud Rate Register (SPIBRR) — Address 7044h

7	6	5	4	3	2	1	0
Reserved	SPI BIT RATE 6	SPI BIT RATE 5	SPI BIT RATE 4	SPI BIT RATE 3	SPI BIT RATE 2	SPI BIT RATE 1	SPI BIT RATE 0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

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

Bit 7 **Reserved**. Reads are indeterminate and writes have no effect.

Bits 6–0 **SPI BIT RATE 6–SPI BIT RATE 0**. SPI Bit Rate (Baud) Control. These bits determine the bit transfer rate if the SPI is the network master. There are 125 data transfer rates (each a function of the system clock) that can be selected. One data bit is shifted per SPICLK cycle.

If the SPI is a network slave, the module receives a clock on the SPICLK pin from the network master; therefore, these bits have no effect on the SPICLK signal. The frequency of the input clock from the master should not exceed the slave SPI's SPICLK signal divided by 8.

In master mode, the SPI clock is generated by the SPI and is output on the SPICLK pin. The SPI baud rates are determined by the formula in Equation 5–2.

Equation 5–2. SPI Baud Rate Calculation

- ❑ SPI Baud Rate for SPIBRR = 3 to 127:

$$SPI\ Baud\ Rate = \frac{SYSCLK}{(SPIBRR + 1)}$$

- ❑ SPI Baud Rate for SPIBRR = 0, 1, or 2:

$$SPI\ Baud\ Rate = \frac{SYSCLK}{4}$$

where:
SYSCLK = System clock frequency of the device
SPIBRR = Contents of the SPIBRR in the master SPI device

5.3.5 SPI Emulation Buffer Register (SPIEMU)

The SPIEMU contains the received data. Reading the SPIEMU does not clear the SPI INT FLAG bit (SPISTS.6 on page 5-23).

Figure 5–13. SPI Emulation Buffer Register (SPIEMU) — Address 7046h

7	6	5	4	3	2	1	0
ERCVD7	ERCVD6	ERCVD5	ERCVD4	ERCVD3	ERCVD2	ERCVD1	ERCVD0
R–x	R–x	R–x	R–x	R–x	R–x	R–x	R–x

Note: R = Read access, –n = value after reset (x=indeterminate)

Bits 7–0 **ERCVD7–ERCVD0**. Emulation Buffer Received Data. The SPIEMU functions almost identically to the SPIBUF, except that reading the SPIEMU does not clear the SPI INT FLAG bit (SPISTS.6 on page 5-23). Once the SPIDAT has received the complete character, the character is transferred to the SPIEMU and SPIBUF, where it can be read. At the same time, SPI INT FLAG is set.

This mirror register was created to support emulation. Reading the SPIBUF clears the SPI INT FLAG bit (SPISTS.6). In the normal operation of the emulator, the control registers are read to continually update the contents of these registers on the display screen. The SPIEMU was created so that the emulator can read this register and properly update the contents on the display screen. Reading SPIEMU does not clear the SPI INT FLAG, but reading SPIBUF clears this flag. In other words, SPIEMU enables the emulator to emulate the true operation of the SPI more accurately.

It is recommended that you read SPIBUF in the normal emulator run mode.

5.3.6 SPI Serial Input Buffer Register (SPIBUF)

The SPIBUF contains the received data. Reading the SPIBUF clears the SPI INT FLAG bit (SPISTS.6 on page 5-23).

Figure 5–14. SPI Serial Input Buffer Register (SPIBUF) — Address 7047h

7	6	5	4	3	2	1	0
RCVD7	RCVD6	RCVD5	RCVD4	RCVD3	RCVD2	RCVD1	RCVD0
R–x	R–x	R–x	R–x	R–x	R–x	R–x	R–x

Note: R = Read access, –n = value after reset (x=indeterminate)

Bits 7–0 **RCVD7–RCVD0**. Received Data. Once SPIDAT has received the complete character, the character is transferred to SPIBUF, where it can be read. At the same time, the SPI INT FLAG bit (SPISTS.6 on page 5-23) is set. Since data is shifted into the SPI most significant bit first, it is stored right justified in this register.

Note: Reading the SPIBUF

Reading SPIBUF clears the SPI INT FLAG bit (SPISTS.6 on page 5-23).

5.3.7 SPI Serial Data Register (SPIDAT)

The SPIDAT is the transmit/receive shift register. Data written to the SPIDAT is shifted out (MSB) on subsequent SPICLK cycles. For every bit shifted out (MSB) of the SPI, a bit is shifted into the LSB end of the shift register.

Figure 5–15. SPI Serial Data Register (SPIDAT) — Address 7049h

7	6	5	4	3	2	1	0
SDAT7	SDAT6	SDAT5	SDAT4	SDAT3	SDAT2	SDAT1	SDAT0
RW–x	RW–x	RW–x	RW–x	RW–x	RW–x	RW–x	RW–x

Note: R = Read access, W = Write access, –n = value after reset (x=indeterminate)

Bits 7–0 **SDAT7–SDAT70**. Serial Data. Writing to the SPIDAT performs two functions:

- ❑ It provides data to be output on the serial output pin if the TALK bit (SPICTL.1 on page 5-21) is set.
- ❑ When the SPI is operating as a master, a data transfer is initiated. When initiating a transfer, see the CLOCK POLARITY bit (SPICCR.6 on page 5-18) and the CLOCK PHASE bit (SPICTL.3 on page 5-22) for the requirements.

Writing dummy data to SPIDAT initiates a receiver sequence. Since the data is not hardware-justified for characters shorter than eight bits, transmit data must be written in left-justified form, and received data read in right-justified form.

5.3.8 SPI Port Control Register 1 (SPIPC1)

The SPIPC1 controls the SPISTE pin and the SPICLK pin functions.

Figure 5–16. SPI Port Control Register 1 (SPIPC1) — Address 704Dh

7	6	5	4	3	2	1	0
SPISTE DATA IN	SPISTE DATA OUT	SPISTE FUNCTION	SPISTE DATA DIR	SPICLK DATA IN	SPICLK DATA OUT	SPICLK FUNCTION	SPICLK DATA DIR
R–x	RW–0	RW–0	RW–0	R–x	RW–0	RW–0	RW–0

Note: R = Read access, W = Write access, –n = value after reset (x=indeterminate)

Bit 7 **SPISTE DATA IN**. SPI Slave Transmit Enable Data In Flag. This bit contains the current value on the SPISTE pin, regardless of the mode. A write to this bit has no effect.

Bit 6 **SPISTE DATA OUT**. SPI Slave Transmit Enable Data Out Flag. This bit contains the data to be output on the SPISTE pin (depending on the mode of operation) if the following conditions are met:

❑ **Master** mode (SPICTL.2 = 1 — this bit is defined on page 5-21)

- SPISTE DATA DIR (SPIPC1.4) bit = 1
- SPISTE FUNCTION (SPIPC1.5) bit = x (indeterminate)

❑ **Slave** mode (SPICTL.2 = 0 — this bit is defined on page 5-21)

Typically, the SPISTE pin in the slave mode acts as an SPI module select (SPIPC1.5 = 1); when this occurs, there is no attempt to write a value out on the SPISTE pin. However, the SPISTE pin can be used as a general-purpose digital I/O pin in the slave mode if the following conditions are met:

- SPISTE DATA DIR (SPIPC1.4) bit = x (1 = output; 0 = input)
- SPISTE FUNCTION (SPIPC1.5) bit = 0

Bit 5 **SPISTE FUNCTION**. SPI Slave Transmit Enable Pin Function Select. This bit selects the function of the SPISTE pin (depending on the mode of operation) if the following conditions are met:

❑ **Maste**r mode (SPICTL.2 = 1 — this bit is defined on page 5-21)

- The SPISTE FUNCTION bit has no effect on the SPISTE pin in the master mode (in other words, the bit is a don't care).
- The SPISTE pin always functions as a general-purpose I/O pin.

❑ **Slave** mode (SPICTL.2 = 0 — this bit is defined on page 5-21)

0 = SPISTE pin functions as a general-purpose digital I/O pin
1 = SPISTE pin functions as the SPI module select pin

Bit 4 **SPISTE DATA DIR**. SPI Slave Transmit Enable Pin Data Direction. This bit determines the data direction on the SPISTE pin if the SPISTE FUNCTION bit = 0 or if the SPI is operating in the master mode (SPICTL.1 = 1 — this bit is defined on page 5-21).

0 = SPISTE is configured as an input pin.
1 = SPISTE is configured as an output pin.

Bit 3 **SPICLK DATA IN**. SPICLK Pin Port Data In Flag. This bit contains the current value on the SPICLK pin, regardless of the mode. A write to this bit has no effect.

Bit 2 **SPICLK DATA OUT**. SPICLK Pin Port Data Out. This bit contains the data to be output on the SPICLK pin if the following conditions are met:

❑ SPICLK pin has been defined as a general-purpose digital I/O pin (SPICLK FUNCTION = 0).

❑ SPICLK pin data direction has been defined as output (SPICLK DATA DIR = 1).

Bit 1 **SPICLK FUNCTION**. SPICLK Pin Function Select. This bit defines the function of the SPICLK pin.

0 = SPICLK is a general-purpose digital I/O pin.
1 = SPICLK pin contains the SPI clock.

Bit 0 **SPICLK DATA DIR**. SPICLK Data Direction. This bit determines the data direction on the SPICLK pin if SPICLK has been defined as a general-purpose digital I/O pin (SPICLK FUNCTION = 0).

0 = SPICLK pin is an input pin.
1 = SPICLK pin is an output pin.

5.3.9 SPI Port Control Register 2 (SPIPC2)

The SPIPC2 controls the SPISOMI and SPISIMO pin functions.

Figure 5–17. SPI Port Control Register 2 (SPIPC2) — Address 704Eh

7	6	5	4	3	2	1	0
SPISIMO DATA IN	SPISIMO DATA OUT	SPISiMO FUNCTION	SPISiMO DATA DIR	SPISOMI DATA IN	SPISOMI DATA OUT	SPISOMI FUNCTION	SPISOMI DATA DIR
R–x	RW–0	RW–0	RW–0	R–x	RW–0	RW–0	RW–0

Note: R = Read access, W = Write access, –n = value after reset (x=indeterminate)

Bit 7 **SPISIMO DATA IN**. SPISIMO Pin Data In. This bit contains the current value on the SPISIMO pin, regardless of the mode. A write to this bit has no effect.

Bit 6 **SPISIMO DATA OUT**. SPISIMO Pin Data Out. This bit contains the data to be output on the SPISIMO pin if both the following conditions are met:

- ❑ The SPISIMO pin has been defined as a general-purpose digital I/O pin (SPISIMO FUNCTION = 0).
- ❑ The SPISIMO pin data direction has been defined as output (SPISIMO DATA DIR = 1).

Bit 5 **SPISIMO FUNCTION**. SPISIMO Pin Function Select. This bit defines the function of the SPISIMO pin.

0 = The SPISIMO pin is a general-purpose digital I/O pin.
1 = The SPISIMO pin contains the SPI data.

Bit 4 **SPISIMO DATA DIR**. SPISIMO Data Direction. This bit determines the data direction on the SPISIMO pin if this pin has been defined as a general-purpose I/O pin (SPISIMO FUNCTION = 0).

0 = The SPISIMO pin is an input pin.
1 = The SPISIMO pin is an output pin.

Bit 3 **SPISOMI DATA IN**. SPISOMI Pin Data In. This bit contains the current value on the SPISOMI pin, regardless of the mode. A write to this bit has no effect.

Bit 2 **SPISOMI DATA OUT**. SPISOMI Pin Data Out. This bit contains the data to be output on the SPISOMI pin if both the following conditions are met:

- ❑ The SPISOMI pin has been defined as a general-purpose digital I/O pin (SPISOMI FUNCTION = 0).
- ❑ The SPISOMI pin data direction has been defined as output (SPISOMI DATA DIR = 1).

Bit 1	**SPISOMI FUNCTION**. SPISOMI Pin Function Select. This bit defines the function of the SPISOMI pin. When SPISOMI is an input signal and SPISOMI FUNCTION and SPISOMI DATA DIR are disabled, the SPICLK signal still clocks the internal circuitry.

0 =	The SPISOMI pin is a general-purpose digital I/O pin.
1 =	The SPISOMI pin contains the SPI data.

Bit 0	**SPISOMI DATA DIR**. SPISOMI Data Direction. This bit determines the data direction on the SPISOMI pin if this pin has been defined as a general-purpose digital I/O pin (SPISOMI FUNCTION = 0).

0 =	The SPISOMI pin is an input pin.
1 =	The SPISOMI pin is an output pin.

5.3.10 SPI Priority Control Register (SPIPRI)

The SPIPRI selects the interrupt priority level of the SPI interrupt and controls the SPI operation on the XDS emulator during program suspensions.

Figure 5–18. SPI Priority Control Register (SPIPRI) — Address 704Fh

7	6	5	4 – 0
Reserved	SPI PRIORITY	SPI ESPEN	Reserved
	RW–0	RW–0	

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

Bit 7 **Reserved**. Reads are indeterminate and writes have no effect.

Bit 6 **SPI PRIORITY**. Interrupt Priority Select. This bit specifies the priority level of the SPI interrupt.

0 = Interrupts are high priority requests.
1 = Interrupts are low priority requests.

Bit 5 **SPI ESPEN**. Emulator Suspend Enable. This bit has no effect except when you are using the XDS emulator to debug a program; in that case, this bit determines SPI operation when the program is suspended by an action such as a hardware or software breakpoint.

0 = When the emulator is suspended, the SPI continues to work until the current transmit/receive sequence is complete.
1 = When the emulator is suspended, the state of the SPI is frozen so that it can be examined at the point that the emulator was suspended.

Bits 4–0 **Reserved**. Reads are indeterminate and writes have no effect.

5.4 SPI Operation-Mode Initialization Examples

Example 5–3. TMS320x240 SPI Slave Mode Code

```
;***************************************************************************
; File Name   : TMS320x240 SPI Slave Mode Example Code
;
; TMS320x240 SPI example code #2:  4 Pin SPI option
;                                  - SLAVE MODE
;                                  - Interrupts are enabled
;                                  - # bytes of data transmitted - 7h
;                                  - # bytes of data received    - 8h
;
****************************************************************************
; SET statements for ’24x devices are device dependent.  The SET locations
; used in this example are typically true for devices with only one SPI
; module.  Consult the device data sheet to determine the exact memory map
; locations of the modules you will be accessing (control registers, RAM,
; ROM).
;
       .include ”c240reg.h”       ; contains a list of SET statements
                                  ; for all registers on TMS320x240.
; The following SET statements for the SPI are contained in c240reg.h
; and are shown explicitly for clarity.
;Serial Peripheral Interface (SPI) Registers
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
SPICCR       .set   07040h        ; SPI Configuration Control Register
SPICTL       .set   07041h        ; SPI Operation Control Register
SPISTS       .set   07042h        ; SPI Status Register
SPIBRR       .set   07044h        ; SPI Baud Rate Register
SPIEMU       .set   07046h        ; SPI Emulation Buffer Register
SPIBUF       .set   07047h        ; SPI Serial Input Buffer Register
SPIDAT       .set   07049h        ; SPI Serial Data Register
SPIPC1       .set   0704Dh        ; SPI Port Control Register #1
SPIPC2       .set   0704Eh        ; SPI Port Control Register #2
SPIPRI       .set   0704Fh        ; SPI Priority Register
;------------------------------------------------------------
; Constant definitions
;------------------------------------------------------------
LENGTH       .set   08h           ; length of the data stream to be
                                  ; transmitted/received by SEND_ALL
DECODE       .set   01h           ; Decode value to used to enable slave
                                  ; transmissions.
;------------------------------------------------------------
; Variable definitions
;------------------------------------------------------------
;------------------------------------------------------------
; The transmit and receive buffer locations are defined below.
; The actual .bss location will need to be defined in the
; linker control file.
;
```

Example 5–3. TMS320x240 SPI Slave Mode Code (Continued)

```
       .bss DATAOUT,LENGTH        ; Location of LENGTH byte character stream
                                  ; transmitted by the SEND_ALL routine.
       .bss DATAIN,LENGTH         ; Location of LENGTH byte character stream
                                  ; received by the SEND_ALL routine.
;------------------------------------------------------------
;------------------------------------------------------------
; Macro definitions
;------------------------------------------------------------
KICK_DOG      .macro              ;Watchdog reset macro
              LDP    #00E0h
              SPLK   #05555h, WD_KEY
              SPLK   #0AAAAh, WD_KEY
              LDP    #0h
              .endm
;============================================================
; Initialized data for SEND_ALL subroutine
;============================================================
       .data
TXDATA        .word  0FDh,0FBh,0F7h,0EFh,0DFh,0BFh,07Fh
;============================================================
; Reset & interrupt vectors
;============================================================
       .sect        ”vectors”
RSVECT        B      START        ; PM 0       Reset Vector        1
INT1          B      INT1_ISR     ; PM 2       Int level 1         4
INT2          B      PHANTOM      ; PM 4       Int level 2         5
INT3          B      PHANTOM      ; PM 6       Int level 3         6
INT4          B      PHANTOM      ; PM 8       Int level 4         7
INT5          B      PHANTOM      ; PM A       Int level 5         8
INT6          B      PHANTOM      ; PM C       Int level 6         9
RESERVED      B      PHANTOM      ; PM E       (Analysis Int)      10
SW_INT8       B      PHANTOM      ; PM 10      User S/W int        -
SW_INT9       B      PHANTOM      ; PM 12      User S/W int        -
SW_INT10      B      PHANTOM      ; PM 14      User S/W int        -
SW_INT11      B      PHANTOM      ; PM 16      User S/W int        -
SW_INT12      B      PHANTOM      ; PM 18      User S/W int        -
SW_INT13      B      PHANTOM      ; PM 1A      User S/W int        -
SW_INT14      B      PHANTOM      ; PM 1C      User S/W int        -
SW_INT15      B      PHANTOM      ; PM 1E      User S/W int        -
SW_INT16      B      PHANTOM      ; PM 20      User S/W int        -
TRAP          B      PHANTOM      ; PM 22      Trap vector         -
NMI           B      PHANTOM      ; PM 24      Non maskable Int    3
EMU_TRAP      B      PHANTOM      ; PM 26      Emulator Trap       2
SW_INT20      B      PHANTOM      ; PM 28      User S/W int        -
SW_INT21      B      PHANTOM      ; PM 2A      User S/W int        -
SW_INT22      B      PHANTOM      ; PM 2C      User S/W int        -
SW_INT23      B      PHANTOM      ; PM 2E      User S/W int        -
```

Example 5–3. TMS320x240 SPI Slave Mode Code (Continued)

```
; Begin the Reset initialization here ...
             .text
START:
       CLRC   SXM                  ; Clear Sign Extension Mode
       CLRC   OVM                  ; Reset Overflow Mode
* Set Data Page pointer to  page 1 of the peripheral frame
       LDP    #DP_PF1              ; Page DP_PF1 includes WET through EINT frames
* initialize WDT registers
       SPLK   #06Fh, WDTCR         ; clear WDFLAG, Disable WDT, set WDT for 1 second
                                   ; overflow (max)
       SPLK   #07h, RTICR          ; clear RTI Flag, set RTI for 1 second overflow
                                   ; (max)
* configure PLL for 4MHz xtal, 10MHz SYSCLK and 20MHz CPUCLK
       SPLK   #00E4h,CKCR1         ; CLKIN(XTAL)=4MHz,CPUCLK=20MHz
       SPLK   #0043h,CKCR0         ; CLKMD=PLL disable,SYSCLK=CPUCLK/2,
       SPLK   #00C3h,CKCR0         ; CLKMD=PLL Enable,SYSCLK=CPUCLK/2,
* Clear reset flag bits in SYSSR (PORRST, PLLRST, ILLRST, SWRST, WDRST)
       LACL   SYSSR                ; ACCL <= SYSSR
       AND    #00FFh               ; Clear upper 8 bits of SYSSR
       SACL   SYSSR                ; Load new value into SYSSR
* Initialize IOP20/CLKOUT pin for use as DSP clock out
       SPLK   #40C8h,SYSCR         ; No reset, CLKOUT=CPUCLK, VCCA on
* initialize B2 RAM to zero's.
       LAR    AR1,#B2_SADDR        ; AR1 <= B2 start address
       MAR    *,AR1                ; use B2 start address for next indirect
       ZAC                         ; ACC <= 0
       RPT    #1fh                 ; set repeat counter for 1fh+1=20h or 32 loops
       SACL   *+                   ; write zeros to B2 RAM
* initialize DATAOUT with data to be transmitted.
       LAR    AR1,#DATAOUT         ; AR1 <= DATAOUT start address
       RPT    #06h                 ; set repeat counter for 6h+1=7h or 7 loops
       BLPD   #TXDATA,*+           ; loads 60h - 67h with TXDATA
       CALL INIT_SPI
* Initialize DSP for interrupts
       LAR    AR6,#IMR             ;
       LAR    AR7,#IFR             ;
       MAR    *,AR6
       LACL   #01h                 ;
       SACL   *,AR7                ; Enable interrupts 1 only
       LACL   *                    ; Clear IFR by reading and
       SACL   *,AR2                ; writing contents back into itself
       CLRC   INTM                 ; Enable DSP interrupts
; Main routine goes here.
```

Example 5–3. TMS320x240 SPI Slave Mode Code (Continued)

```
MAIN                       ; Main loop of code begins here
;    ....                  ; insert actual code here
     NOP
     NOP
     NOP
;      ....                ; insert actual code here
       B      MAIN         ; The MAIN program loop has completed one
                           ; pass, branch back to the beginning and
                           ; continue.
*******************************************************************************
*               Subroutines                                                   *
*******************************************************************************
;============================================================================
; Routine Name: INIT_SPI         Routine Type: SR
;
; Description: This SR initializes the SPI for data stream transfer
;              to a master SPI.  The ’240 SPI is configured for
;              8-bit transfers as a slave.
;============================================================================
INIT_SPI:
* initialize SPI in slave mode
       SPLK   #008Fh,SPICCR; Reset SPI by writing 1 to SWRST
       SPLK   #0008h,SPICTL; Disable ints & TALK, normal clock, slave mode
       SPLK   #000Eh,SPIBRR; Set baud rate to ’fastest’
; NOTE: The baud rate should be as fast as possible for communications
;       between two or more SPI’s.  Issues in the baud rate selection to
;       remember are the master vs. slave maximum speed differences, and
;       to a lessor degree, the clock speed of each device.  For DSP
;       controllers and PRISM devices this determined by SYSCLK.
;
;       A value of ’0Eh’ in the SPIBRR will insure the fastest available
;       baud rate for the master and slave device (assuming two DSP controller
;       devices with the same SYSCLK are doing the communication).  This is
;       case when the master SPI uses a polling routine to determine when
;       to transmit the next byte.
       SPLK   #0022h,SPIPC1; Enable the SPISTE and SPICLK pin functions.
                           ; SPISTE will functiion as a transmit enable
                           ; input for the slave SPI module.
       SPLK   #0022h,SPIPC2; Set SIMO & SOMI functions to serial I/O
       SPLK   #0000h,SPIPRI; Set SPI interrupt to high priority.
                           ; For emulation purposes, allow the SPI
                           ; to continue after an XDS suspension.
                           ; HAS NO EFFECT ON THE ACTUAL DEVICE.
       SPLK   #0000h,SPISTS; Clear the SPI interrupt status bits
       SPLK   #0007h,SPICCR; Release SWRST, clock polarity 0, 8 bits
       SPLK   #0009h,SPICTL; Disable TALK & RCV int, CLK ph 1, slave mode
```

Example 5–3. TMS320x240 SPI Slave Mode Code (Continued)

```
* Initialize Auxilliary Registers for SPI receive ISR
       LAR    AR1,#LENGTH-1; load length of data stream into AR1
                           ; and use for transmit/receive loop counter.
       LAR    AR2,#DATAOUT ; load location of transmit data stream into
                           ; AR2.
       LAR    AR3,#DATAIN  ; load location of receive data stream into
                           ; AR3.
       RET                 ; Return to MAIN routine.
********************************************************************************
*                ISR's                                                         *
********************************************************************************
;===============================================================================
; I S R  -  INT1 interrupt service routine
;
; Description: This is an implementation of Method 3: ISR for Single Event
;              per Interrupt Level (see interrupt section in System Functions
;              chapter in Vol 1 of the user's guide).
;
;              This ISR performs initial receive of the DECODE byte from the
;              master SPI.  This byte is checked to determine if the master
;              is requesting data from this SPI, and if so, sets the TALK bit
;              to enable transmission and writes a byte into SPIDAT
;              from DATAOUT.  The number of bytes received is controlled
;              by the constant LENGTH, which is determined prior to
;              assembly.
;              The TALK bit is cleared after LENGTH # of bytes have been
;              received, and the Auxiliary register pointers are reloaded
;              in preparation of the next transfer.
;===============================================================================
INT1_ISR                   ; Interrupt 1 Interrupt Service Routine

       MAR    *,AR3        ; use location of DATAIN for next indirect
       LACL   SPIBUF       ; ACC <= SPI Buffer Register
       SACL   *+,AR2       ; store value in B2 @ DATAIN
                           ; use DATAOUT address for next indirect
       XOR    #DECODE      ; compare received byte to determine if slave SPI
                           ; is selected.
       BCND   SKIP,NEQ
       SPLK   #000Bh,SPICTL; Enable TALK & RCV int, CLK ph 1, slave mode
SKIP
       LACL   *+,AR1       ; ACC <= byte to xmit
                           ; Increment AR2 by one to point to next byte
                           ; in data stream.
                           ; use # bytes left to TX for next indirect address
       SACL   SPIDAT       ; store Xmit byte to SPIDAT and wait for master clock.
       BANZ   SKIP2,AR2    ; Branch to SKIP2 if AR1 is not zero,
                           ; decrement AR1 by one,
                           ; use DATAOUT address for next address
```

Example 5–3. TMS320x240 SPI Slave Mode Code (Continued)

```
* Re-Initialize Auxilliary Registers for SPI receive ISR
       LAR    AR1,#LENGTH-1; load length of data stream into AR1
                           ; and use for transmit/receive loop counter.
       LAR    AR2,#DATAOUT ; load location of transmit data stream into
                           ; AR2.
       LAR    AR3,#DATAIN  ; load location of receive data stream into
                           ; AR3.
* Disable talk after LENGTH # of transfers.
       SPLK   #0009h,SPICTL; Disable TALK & RCV int, CLK ph 1, slave mode
SKIP2
       CLRC   INTM         ; Enable DSP interrupts
       RET                 ; Return from interrupt
;===============================================================================
; I S R  -  PHANTOM
;
; Description:      ISR used to trap spurious interrupts.
;
;===============================================================================
PHANTOM
END    B      END          ;
       .end
```

Example 5–4. TMS320x240 SPI Master Mode Code

```
;****************************************************************************
; File Name   : TMS320x240 SPI Master Mode Example Code
;
; TMS320x240 SPI example code #1:  4 Pin SPI option
;                                  - MASTER MODE
;                                  - Interrupts are polled
;                                  - # bytes of data transmitted - 8h
;                                  - # bytes of data received    - 8h
;
*****************************************************************************
; SET statements for '24x devices are device dependent.  The SET locations
; used in this example are typically true for devices with only one SPI
; module.  Consult the device data sheet to determine the exact memory map
; locations of the modules you will be accessing (control registers, RAM,
; ROM).
;
       .include "c240reg.h"       ; contains a list of SET statements
                                  ; for all registers on TMS320x240.
; The following SET statements for the SPI are contained in c240reg.h
; and are shown explicitly for clarity.
;Serial Peripheral Interface (SPI) Registers
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
SPICCR       .set   07040h       ; SPI Configuration Control Register
SPICTL       .set   07041h       ; SPI Operation Control Register
SPISTS       .set   07042h       ; SPI Status Register
SPIBRR       .set   07044h       ; SPI Baud Rate Register
SPIEMU       .set   07046h       ; SPI Emulation Buffer Register
SPIBUF       .set   07047h       ; SPI Serial Input Buffer Register
SPIDAT       .set   07049h       ; SPI Serial Data Register
SPIPC1       .set   0704Dh       ; SPI Port Control Register #1
SPIPC2       .set   0704Eh       ; SPI Port Control Register #2
SPIPRI       .set   0704Fh       ; SPI Priority Register
;------------------------------------------------------------
; Constant definitions
;------------------------------------------------------------
LENGTH       .set 08h            ; length of the data stream to be
                                 ; transmitted/received by SEND_ALL
;------------------------------------------------------------
; Variable definitions
;------------------------------------------------------------
;------------------------------------------------------------
; The transmit and receive buffer locations are defined below.
; The actual .bss location will need to be defined in the
; linker control file.
;
             .bss DATAOUT,LENGTH ; Location of LENGTH byte character stream
                                 ; transmitted by the SEND_ALL routine.
             .bss DATAIN,LENGTH  ; Location of LENGTH byte character stream
                                 ; received by the SEND_ALL routine.
;------------------------------------------------------------
```

Example 5–4. TMS320x240 SPI Master Mode Code (Continued)

```
SPI_DONE      .set 070h           ; Defined B2 RAM location ’70h’ as SPI transmit
                                  ; status location.  1=complete, 0=not complete
;-------------------------------------------------------------
; Macro definitions
;-------------------------------------------------------------
KICK_DOG      .macro              ;Watchdog reset macro
              LDP    #00E0h
              SPLK   #05555h, WD_KEY
              SPLK   #0AAAAh, WD_KEY
              LDP    #0h
              .endm
;=============================================================
; Initialized data for SEND_ALL subroutine
;=============================================================
     .data
TXDATA    .word     01h,02h,04h,08h,10h,20h,40h,80h
;=============================================================
; Reset & interrupt vectors
;=============================================================
       .sect  ”vectors”
RSVECT        B      START        ; PM 0       Reset Vector        1
INT1          B      PHANTOM      ; PM 2       Int level 1         4
INT2          B      PHANTOM      ; PM 4       Int level 2         5
INT3          B      PHANTOM      ; PM 6       Int level 3         6
INT4          B      PHANTOM      ; PM 8       Int level 4         7
INT5          B      PHANTOM      ; PM A       Int level 5         8
INT6          B      PHANTOM      ; PM C       Int level 6         9
RESERVED      B      PHANTOM      ; PM E       (Analysis Int)      10
SW_INT8       B      PHANTOM      ; PM 10      User S/W int        -
SW_INT9       B      PHANTOM      ; PM 12      User S/W int        -
SW_INT10      B      PHANTOM      ; PM 14      User S/W int        -
SW_INT11      B      PHANTOM      ; PM 16      User S/W int        -
SW_INT12      B      PHANTOM      ; PM 18      User S/W int        -
SW_INT13      B      PHANTOM      ; PM 1A      User S/W int        -
SW_INT14      B      PHANTOM      ; PM 1C      User S/W int        -
SW_INT15      B      PHANTOM      ; PM 1E      User S/W int        -
SW_INT16      B      PHANTOM      ; PM 20      User S/W int        -
TRAP          B      PHANTOM      ; PM 22      Trap vector         -
NMI           B      PHANTOM      ; PM 24      Non maskable Int    3
EMU_TRAP      B      PHANTOM      ; PM 26      Emulator Trap       2
SW_INT20      B      PHANTOM      ; PM 28      User S/W int        -
SW_INT21      B      PHANTOM      ; PM 2A      User S/W int        -
SW_INT22      B      PHANTOM      ; PM 2C      User S/W int        -
SW_INT23      B      PHANTOM      ; PM 2E      User S/W int        -
; Begin the Reset initialization here ...
              .text
START:
       CLRC   SXM                 ; Clear Sign Extension Mode
       CLRC   OVM                 ; Reset Overflow Mode
* Set Data Page pointer to  page 1 of the peripheral frame
       LDP    #DP_PF1       ; Page DP_PF1 includes WET through EINT frames
```

Example 5–4. TMS320x240 SPI Master Mode Code (Continued)

```
* initialize WDT registers
       SPLK   #06Fh, WDTCR ; clear WDFLAG, Disable WDT, set WDT for 1 second
                           ; overflow (max)
       SPLK   #07h, RTICR  ; clear RTI Flag, set RTI for 1 second overflow (max)
* configure PLL for 4MHz xtal, 10MHz SYSCLK and 20MHz CPUCLK
       SPLK   #00E4h,CKCR1 ; CLKIN(XTAL)=4MHz,CPUCLK=20MHz
       SPLK   #0043h,CKCR0 ; CLKMD=PLL disable,SYSCLK=CPUCLK/2,
       SPLK   #00C3h,CKCR0 ; CLKMD=PLL Enable,SYSCLK=CPUCLK/2,
* Clear reset flag bits in SYSSR (PORRST, PLLRST, ILLRST, SWRST, WDRST)
       LACL   SYSSR        ; ACCL <= SYSSR
       AND    #00FFh       ; Clear upper 8 bits of SYSSR
       SACL   SYSSR        ; Load new value into SYSSR
* Initialize IOP20/CLKOUT pin for use as DSP clock out
       SPLK   #40C8h,SYSCR ; No reset, CLKOUT=CPUCLK, VCCA on
* initialize B2 RAM to zero’s.
       LAR    AR1,#B2_SADDR; AR1 <= B2 start address
       MAR    *,AR1        ; use B2 start address for next indirect
       ZAC                 ; ACC <= 0
       RPT    #1fh         ; set repeat counter for 1fh+1=20h or 32 loops
       SACL   *+           ; write zeros to B2 RAM
* initialize DATAOUT with data to be transmitted.
       LAR    AR1,#DATAOUT ; AR1 <= DATAOUT start address
       RPT    #07h         ; set repeat counter for 7h+1=8h or 8 loops
       BLPD   #TXDATA,*+   ; loads 60h - 68h with 01, 02, 04, ... , 40, 80h
       CALL INIT_SPI
; Main routine goes here.  Whenever the data stream previously loaded
; into the DATAOUT location is desired to be transmitted,
; the SEND_ALL subroutine is called.
MAIN                       ; Main loop of code begins here ...
;      ....                ; insert actual code here
       CALL SEND_ALL       ; Call the SEND_ALL subroutine and when it is
                           ; finished, continue with the MAIN loop.
;      ....                ; insert actual code here
       B    MAIN           ; The MAIN program loop has completed one
                           ; pass, branch back to the beginning and
                           ; continue.
*******************************************************************************
*                Subroutines                                                  *
*******************************************************************************
;=============================================================================
; Routine Name: INIT_SPI          Routine Type: SR
;
; Description: This SR initializes the SPI for data stream transfer
;              to a slave SPI.  The ’240 SPI is configured for
;              8-bit transfers as a master.
;;=============================================================================
INIT_SPI:
* initialize SPI in master mode
       SPLK   #008Fh,SPICCR       ; Reset SPI by writing 1 to SWRST
       SPLK   #000Ch,SPICTL       ; Disable ints & TALK, normal clock, master mode
       SPLK   #000Eh,SPIBRR       ; Set baud rate to ’fastest’
```

Example 5–4. TMS320x240 SPI Master Mode Code (Continued)

```
; NOTE: The baud rate should be as fast as possible for communications
;       between two or more SPI’s.  Issues in the baud rate selection to
;       remember are the master vs. slave maximum speed differences, and
;       to a lessor degree, the clock speed of each device.  For DSP
;       controllers and PRISM devices this determined by SYSCLK .
;
;       A value of ’0Eh’ in the SPIBRR will insure the fastest available
;       baud rate for the master and slave device (assuming two DSP controller
;       devices with the same SYSCLK are doing the communication).  This
;       is true for the case when the master SPI uses a polling routine to
;       determine when to tranmsit the next byte.
      SPLK   #0052h,SPIPC1       ; Enable the SPICLK pin function.
                                 ; SPISTE will always be general I/O when
                                 ; SPI is in master mode, regardless of
                                 ; function bit state.  Set SPISTE as output
                                 ; high – disable receiver SPI output.
      SPLK   #0022h,SPIPC2       ; Set SIMO & SOMI functions to serial I/O
      SPLK   #0000h,SPIPRI       ; Set SPI interrupt to high priority.
                                 ; For emulation purposes, allow the SPI
                                 ; to continue after an XDS suspension.
                                 ; HAS NO EFFECT ON THE ACTUAL DEVICE.
      SPLK   #0000h,SPISTS       ; Clear the SPI interrupt status bits
      SPLK   #0007h,SPICCR       ; Release SWRST, clock polarity 0, 8 bits
      SPLK   #000Eh,SPICTL       ; Enable TALK, CLK ph 1, master mode
      RET                        ; Return to MAIN routine.
;============================================================================
; Routine Name: SEND_ALL         Routine Type: SR
;
; Description: This SR performs the data stream transfer.  Data to be
;              transmitted is located at DATAOUT.  Received Data is stored
;              at DATAIN.  This routine polls the SPI INT FLAG bit,
;              SPISTS.6, to determine when each byte transfer has
;              completed.  The number of bytes transfered is controlled
;              by the constant LENGTH, which is determined prior to
;              assembly.
;============================================================================
SEND_ALL:
      LAR    AR1,#LENGTH–1       ; load length of data stream into AR1
                                 ; and use for transmit/receive loop counter.
      LAR    AR2,#DATAOUT        ; load location of transmit data stream into
                                 ; AR2.
      LAR    AR3,#DATAIN         ; load location of receive data stream into
                                 ; AR3.
      MAR    *,AR2               ; use DATAOUT for next indirect address
; Perform a read–modify–write on SPIPC1 to set SPISTE pin active low
; and enable the slave SPI.
      LACL   SPIPC1              ; load contents of SPIPC1 into ACC.
      AND    #0BFh               ; clear SPIPC1.6 to make SPISTE pin active
                                 ; low.
      SACL   SPIPC1              ; store ACC out to SPIPC1.
```

Example 5–4. TMS320x240 SPI Master Mode Code (Continued)

```
LOOP
; Begin Xmit by writing byte to SPIDAT.
       LACL   *+,AR3              ; ACC <= byte to xmit
                                  ; Increment AR2 by one to point to next byte
                                  ; in data stream.
                                  ; use DATAIN for next indirect address
       SACL   SPIDAT              ; Xmit byte
POLL   LACL   SPISTS              ; Poll the INT Flag Bit to determine when
                                  ; to begin next xmit
       AND    #040h               ; Clear all bits except SPI INT FLAG bit
       XOR    #040h               ; ACC=0 if bit is set
       BCND   POLL,NEQ            ; continue polling if ACC != 0.
       LACL   SPIBUF              ; load the received byte into ACC.
       SACL   *+,AR1              ; save the received byte to DATAIN
                                  ; increment AR3 by one point to next
                                  ; DATAIN location.
                                  ; use AR1, # bytes left to tranfer,
                                  ; for next indirect address.
       BANZ   LOOP,AR2            ; Branch to LOOP if AR1 is not zero,
                                  ; decrement AR1 by one,
                                  ; use DATAOUT address for next address
; Optional code section:  This code loads a value into B2 RAM to
;                         inform the MAIN routine that the data
;                         stream transfer is complete.
       LAR    AR4,#SPI_DONE       ; load address of SPI status location
                                  ; into AR4
       MAR    *,AR4               ; use SPI status location for next indirect
       SPLK   #01h,*              ; write ’01h’ into status location to indicate
                                  ; data stream transfer is complete.
; Perform a read–modify–write on SPIPC1 to set SPISTE pin active high
; and disable the slave SPI.
       LACL   SPIPC1              ; load contents of SPIPC1 into ACC.
       OR     #040h               ; set SPIPC1.6 to make SPISTE pin active
                                  ; high.
       SACL   SPIPC1              ; store ACC out to SPIPC1.
       RET                        ; Return to MAIN routine.
*******************************************************************************
*                ISR’s                                                        *
*******************************************************************************
;==============================================================================
; I S R  –  PHANTOM
;
; Description:      ISR used to trap spurious interrupts.
;==============================================================================
PHANTOM
END    B      END          ;
       .end
```

Chapter 6

Watchdog (WD) and Real-Time Interrupt (RTI) Module

The Watchdog (WD) and Real-Time Interrupt (RTI) module monitors software and hardware operation, provides interrupts at programmable intervals, and implements system reset functions upon CPU disruption. If the software goes into an improper loop, or if the CPU becomes temporarily disrupted, the WD timer overflows to assert a system reset.

Most conditions that temporarily disrupt chip operation and inhibit proper CPU function can be cleared and reset by the *watchdog* function. By its consistent performance, the watchdog increases the reliability of the CPU, thus ensuring system integrity.

The WD/RTI module is part of the PRISM Module Library. It can be combined with other modular building blocks from the PRISM Module Library to generate a diversified family of highly integrated devices.

Note: 8-Bit Peripheral

This module is interfaced to the 16-bit peripheral bus as an 8-bit peripheral. Therefore, reads from bits 15–8 are undefined; writes to bits 15–8 have no effect.

Topic **Page**

6.1 Watchdog (WD) and Real-Time Interrupt (RTI) Overview

6.1.1 Features

The WD/RTI module design includes the following features:

❑ WD timer

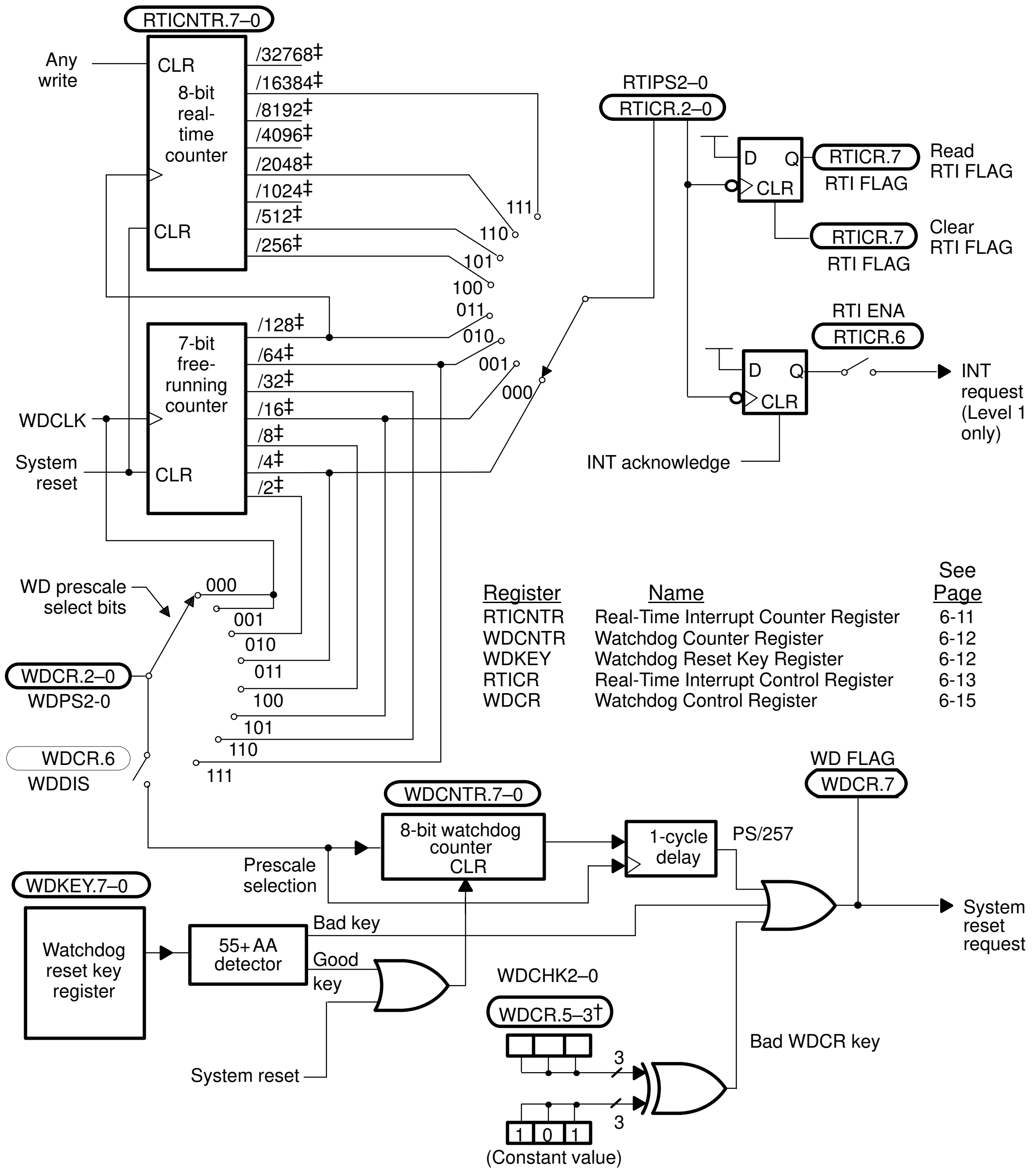
- 8-bit WD counter that generates a system reset upon overflow.
- 7-bit free-running counter that feeds the WD counter via the WD counter prescale.
- A WD reset key (WDKEY) register that clears the WD counter when the correct combination of values are written, and generates a reset if an incorrect value is written to the register.
- A WD flag (WD FLAG) bit that indicates whether the WD timer initiated a system reset.
- WD check bits that initiate a system reset if the WD timer is corrupted.
- Automatic activation of the WD timer, once system reset is released.
- A WD prescale with six selections from the 7-bit free-running counter and two (identical) WDCLK signal inputs.

❑ RTI timer

- RTI prescale that selects from 4 taps of the 8-bit real-time counter and 4 taps of the 7-bit free-running counter.
- Interrupt or polled operation (a software bit enables/disables RTI interrupts).
- An RTI flag (RTI FLAG) bit that indicates whether the RTI counter (RTICNTR) overflows.

Figure 6–1 is a block diagram of the WD/RTI module.

Figure 6–1. Watchdog (WD) and Real-Time Interrupt (RTI) Module Block Diagram



Register	Name	See Page
RTICNTR	Real-Time Interrupt Counter Register	6-11
WDCNTR	Watchdog Counter Register	6-12
WDKEY	Watchdog Reset Key Register	6-12
RTICR	Real-Time Interrupt Control Register	6-13
WDCR	Watchdog Control Register	6-15

† Writing to bits WDCR.5–3 with anything but the correct pattern (101) generates a system reset.
‡ These prescale values are with respect to the WDCLK signal.

6.1.2 Control Registers

Five registers control the WD/RTI operations:

- ❑ **RTI Counter Register (RTICNTR)** contains the value of the RTI counter.
- ❑ **WD Counter Register (WDCNTR)** contains the value of the WD counter.
- ❑ **WD Reset Key Register (WDKEY)** clears the WDCNTR when a 55h value followed by an AAh value is written to WDKEY.
- ❑ **RTI Control Register (RTICR)** contains the following control bits used for RTI configuration:
  - ■ RTI flag bit
  - ■ RTI enable bit
  - ■ RTI prescale select bits (three)
- ❑ **WD Control Register (WDCR)** contains the following control bits used for watchdog configuration:
  - ■ WD flag bit
  - ■ WD check bits (three)
  - ■ WD prescale select bits (three)

Table 6–1 lists the addresses of the WD/RTI registers.

Table 6–1. Addresses of WD/RTI Module Registers

			Described in	
Address	**Register**	**Name**	**Subsection**	**Page**
7020h		Reserved		
7021h	RTICNTR	RTI counter register	6.3.1	6-11
7022h		Reserved		
7023h	WDCNTR	WD counter register	6.3.2	6-12
7024h		Reserved		
7025h	WDKEY	WD reset key register	6.3.3	6-12
7026h		Reserved		
7027h	RTICR	RTI control register	6.3.4	6-13
7028h		Reserved		
7029h	WDCR	WD control register	6.3.5	6-15
702Ah		Reserved		
702Bh		Reserved†		
702Ch		Reserved		
702Dh		Reserved†		
702Eh		Reserved		
702Fh		Reserved		

† Reserved for PLL Clock Module control registers, see Chapter 10, *PLL Clock Module*.

6.2 Operation of Watchdog (WD) and Real-Time Interrupt (RTI) Timers

6.2.1 WD Timer

The WD timer is an 8-bit resettable incrementing counter that is clocked by the output of the prescaler. The timer protects against system software failures and CPU disruption by providing a system reset when WDKEY is not serviced before a watchdog overflow. This reset returns the system to a known starting point. Software then clears WDCNTR by writing a correct data pattern to the WD key logic.

A separate internal clocking signal (WDCLK) is generated by the clock module and is active in all operational modes except the oscillator power-down mode. WDCLK enables the WD timer to function, regardless of the state of any register bit(s) on the chip, except during the oscillator power-down mode, which disables the WDCLK signal. The typical WDCLK frequency is 16 384 Hz. The current state of WDCNTR can be read at any time during its operation.

The typical WDCLK frequency of 16 384 Hz is derived from a power of two input clock frequency (for example, 4.194 MHz, 8.389 MHz). See Chapter 10, *PLL Clock Module*, for more information about the WDCLK signal.

WD Prescale Select

The 8-bit WDCNTR can be clocked directly by the WDCLK signal or through one of six taps from the free-running counter. The 7-bit free-running counter continuously increments at a rate provided by WDCLK (typically 16 384 Hz or 32 768 Hz, both of which are device specific). The WD functions are enabled as long as WDCLK is provided to the module. Any one of the first six taps or the direct input from WDCLK can be selected by the WD prescale select (bits WDPS2–0) as the input to the time base for the WDCNTR. This prescale provides selectable watchdog overflow rates of from 15.63 ms to 1 second for a WDCLK rate of 16 384 Hz. While the chip is in normal operation mode, the free-running counter cannot be stopped or reset, except by a system reset. Clearing either RTICNTR or WDCNTR does not clear the free-running counter.

Servicing the WD Timer

The WDCNTR is reset when the proper sequence is written to the WDKEY before the WDCNTR overflows. The WDCNTR is enabled to be reset when a value of 55h is written to the WDKEY. When the next AAh value is written to the WDKEY, then the WDCNTR actually is reset. Any value written to the WDKEY other than 55h or AAh causes a system reset. Any sequence of 55h and AAh values can be written to the WDKEY without causing a system reset; only a write of 55h followed by a write of AAh to the WDKEY resets the WDCNTR.

Table 6–2 shows a typical sequence written to WDKEY after power-up.

Table 6–2. Typical WDKEY Register Power-up Sequence

Sequential Step	Value Written to WDKEY	Result
1	AAh	No action.
2	AAh	No action.
3	55h	WDCNTR is enabled to be reset by the next AAh.
4	55h	WDCNTR is enabled to be reset by the next AAh.
5	55h	WDCNTR is enabled to be reset by the next AAh.
6	AAh	WDCNTR is reset.
7	AAh	No action.
8	55h	WDCNTR is enabled to be reset by the next AAh.
9	AAh	WDCNTR is reset.
10	55h	WDCNTR is enabled to be reset by the next AAh.
11	23h	System reset due to an improper key value written to WDKEY.

Step 3 in Table 6–2 is the first action to enable the WDCNTR to be reset. The WDCNTR is not actually reset until step 6. Step 8 reenables the WDCNTR to be reset, and step 9 resets the WDCNTR. Step 10 again re-enables the WDCNTR to be reset. Writing the wrong key value to the WDKEY in step 11 causes a system reset.

A WDCNTR overflow or an incorrect key value written to the WDKEY also sets the WD flag (WDFLAG). After a reset, the program reads this flag to determine the source of the reset. After reset, WDFLAG should be cleared by the software to allow the source of subsequent WD resets to be determined. WD resets are not prevented when the flag is set.

WD Reset

When the WDCNTR overflows, the WD timer asserts a system reset. Reset occurs one WDCNTR clock cycle (either WDCLK or WDCLK divided by a prescale value) later. The reset cannot be disabled in normal operation as long as WDCLK is present. The WD timer is, however, disabled in the oscillator power-down mode when WDCLK is not active.

WD Disable

For development purposes, the WD timer can be disabled by applying 5V to the V_{CCP} pin during the device reset sequence and setting the WDDIS bit in the WD control register (WDCR.6). However, if the hardware and software conditions are not met, the WD timer will not be disabled.

WD Check Bit Logic

The WD check bits (WDCR.5–3, described in detail in subsection 6.3.5 on page 6-15) are continuously compared to a constant value (101_2). If the WD check bits do not match this value, a system reset is generated. This functions as a logic check, in case the software improperly writes to the WDCR, or if an external stimulus (such as voltage spikes, EMI, or other disruptive sources) corrupt the contents of the WDCR. Writing to bits WDCR.5–3 with anything but the correct pattern (101_2) generates a system reset.

Note: WDCR Required Values

Any values written to WDCR must include the value 101_2 written to bits 5–3 (WDCHK2 – WDCHK0).

WD Setup

The WD timer operates independently of the CPU and is always enabled. It does not need any CPU initialization to function. When a system reset occurs, the WD timer defaults to the fastest WD timer rate available (15.63 ms for a 16 384 Hz WDCLK signal). As soon as reset is released internally, the CPU starts executing code, and the WD timer begins incrementing. This means that, to avoid a premature reset, WD/RTI setup should occur early in the power-up sequence.

RTI Timer

The RTI timer is an 8-bit counter that can be programmed to generate periodic interrupts at a software-selectable frequency. Eight taps in all—four from the RTI counter (RTICNTR, further described in subsection 6.3.1 on page 6-11) and four from the free-running counter—can be selected through a 1-of-8 multiplexer to generate the frequency of the periodic interrupt requests. The interrupts are enabled and disabled through software. The RTICNTR can be read or cleared at any time. The 7-bit free-running counter, however, cannot be cleared except by system reset.

The actual memory-mapped location of the RTI interrupt vector is device specific. The CPU always accesses the information in a vector in the same manner, regardless of the device, although the physical locations of the vectors may differ. Consult the specific device data sheet for the actual physical location of the RTI interrupt vector.

RTI Prescale Select

The RTICNTR (RTI counter, described in subsection 6.3.1 on page 6-11) uses the /128 (divide-by-128) tap from the free-running counter as an input to generate four output taps. These four taps, plus four additional taps taken from the free-running counter can be selected through a 1-of-8 multiplexer that is controlled through the three prescale select bits of the RTI control register (RTICR.2–0, described in subsection 6.3.4 on page 6-13). This prescale provides selectable interrupt rates of from 1 to 4096 interrupts per second (16 384 Hz WDCLK signal). The RTICNTR is clocked by the WDCLK signal. The RTICNTR can be reset to 00h at any time during normal operation. RTI timer functions are enabled in all modes except the halt mode.

Clearing the RTICNTR with software does not clear the free-running counter, which provides the RTICNTR prescale. Therefore, after the RTICNTR is cleared, timing measurements from this counter yield up to a 1-bit uncertainty.

RTI Enable

The RTI can be enabled or disabled through the RTI enable bit (RTI ENA), which is bit 6 of the RTI control register (RTICR.6). When the RTI is enabled, it allows an interrupt to be generated when the selected overflow occurs. The interrupt logic generates only one interrupt for each transition on the selected tap.

Because the free-running counter cannot be cleared by software, the software logic assumes that the time period specified is measured between consecutive overflows of RTICNTR. Therefore, the timing of the first interrupt cannot be predicted. Waiting until RTI FLAG (RTICR.7) acknowledges the first overflow before enabling the interrupt provides accurate timing.

RTI Flag

The RTI flag bit (RTIFLAG) indicates that an overflow has occurred. This is bit 7 of the RTI control register (RTICR.7). The bit can be cleared by software servicing the RTI timer. Clearing the bit is not required for interrupt generation, and setting the bit with software does not generate an interrupt.

6.3 Watchdog (WD) and Real-Time Interrupt (RTI) Control Registers

The WD/RTI module control registers are shown in Figure 6–2 and discussed in detail in the following subsections.

Figure 6–2. WD/RTI Module Control Registers

		Bit number							
Address	**Register**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
7020h	—	Reserved							
7021h	RTICNTR	D7	D6	D5	D4	D3	D2	D1	D0
7022h	—	Reserved							
7023h	WDCNTR	D7	D6	D5	D4	D3	D2	D1	D0
7024h	—	Reserved							
7025h	WDKEY	D7	D6	D5	D4	S3	S2	D1	D0
7026h	—	Reserved							
7027h	RTICR	RTI FLAG	RTI ENA	Reserved			RTIPS2	RTIPS1	RTIPS0
7028h	—	Reserved							
7029h	WDCR	WD FLAG	Reserved	WDCHK2	WDCHK1	WDCHK0	WDPS2	WDPS1	WDPS0
702Ah	—	Reserved							
702Bh	—	Reserved†							
702Ch	—	Reserved							
702Dh	—	Reserved†							
702Eh	—	Reserved							
702Fh	—	Reserved							

† Reserved for PLL Clock Module control registers, see Chapter 10, *PLL Clock Module*.

6.3.1 Real-Time Interrupt Counter Register (RTICNTR)

The 8-bit real-time interrupt counter register (RTICNTR) contains the value of the real-time counter. It continuously increments using the /128 (128-Hz) overflow from the free-running counter. Although this is an 8-bit counter, only 7 bits are actually needed for the RTI function. The eighth bit (bit 7) can be read and used as an RTI extension bit. The RTICNTR does not stop while the device is in the normal run mode, idle 1 mode, idle 2 mode, or PLL power-down mode.

Figure 6–3. Real–Time Interrupt Counter Register (RTICNTR) — Address 7021h

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0
RC–0	RC–0	RC–0	RC–0	RC–0	RC–0	RC–0	RC–0

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

Bits 7–0 **D7–D0**. Data Values. These read-only data bits contain the 8-bit RTI counter value. Writing any value to this register clears it to 0.

Note: Counter Not Cleared When RTICNTR is Cleared

The free-running counter that prescales RTICNTR is not cleared when RTICNTR is cleared. This gives a cumulative maximum uncertainty of one RTICNTR bit when RTICNTR is used for time measurement.

6.3.2 WD Counter Register (WDCNTR)

The 8-bit WD counter register (WDCNTR) contains the current value of the WD counter. This register continuously increments at a rate selected through the WD control register. When this register overflows, an additional single-cycle (either WDCLK or WDCLK divided by a prescale value) delay is incurred before system reset is asserted. This allows the RTI timer and the WD timer to be programmed to the same period, while still giving the program time to write the proper WD key sequence. Writing the proper sequence to the WD reset key register clears WDCNTR and prevents a system reset; however, this does not clear the free-running counter.

Figure 6–4. WD Counter Register (WDCNTR) — Address 7023h

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0
R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0

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

Bits 7–0 **D7–D0**. Data Values. These read-only data bits contain the 8-bit WD counter value. Writing to this register has no effect.

6.3.3 WD Reset Key Register (WDKEY)

The WD reset key register (WDKEY) clears WDCNTR when a 55h value followed by an AAh value is written to WDKEY. Any combination of AAh and 55h is allowed, but only a 55h followed by an AAh resets the counter. Any other value causes a system reset.

Figure 6–5. WD Reset Key Register (WDKEY) — Address 7025h

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

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

Bits 7–0 **D7–D0**. Data Values. These write-only data bits contain the 8-bit WD reset key value. When read, WDKEY does **not** return the last key value but rather returns the contents of WDCR.

6.3.4 RTI Control Register (RTICR)

Figure 6–6. RTI Control Register (RTICR) — Address 7027h

7	6	5 – 3	2	1	0
RTI FLAG	RTI ENA	Reserved	RTIPS2	RTIPS1	RTIPS0
RW–0	RW–0		RW–0	RW–0	RW–0

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

Bit 7 **RTI FLAG**. Real-Time Interrupt Flag Bit. This status bit shows if an overflow occurred. The bit can be cleared by software servicing the RTI (writing a 0 clears the bit). Clearing this bit is not required for interrupt generation. Setting this bit does not cause an interrupt to be generated.

0 = Overflow has not occurred.
1 = Overflow has occurred.

Bit 6 **RTI ENA**. Real-Time Interrupt Enable Bit. This bit allows an interrupt to be generated when the selected overflow occurs. Clearing this bit clears any pending interrupt request not yet acknowledged. The interrupt is issued based on the value of the free-running counter (and RTICNTR for frequencies of 64 Hz and slower). Only a single interrupt is requested on the leading edge of an overflow.

0 = Clears any pending interrupt request not acknowledged and disables future RTI interrupts.
1 = Enables interrupts to be generated when the RTI flag detects an overflow.

Note:

The value of the free-running counter is software-independent. The software should assume only that the time period specified is measured between consecutive interrupts. Software cannot predict when the first interrupt will occur, except as measured from system reset.

Bits 5–3 **Reserved**. Writing to these bits has no effect. These bits always read as 0.

Bits 2–0 **RTIPS2–RTIPS0**. Real-Time Interrupt Prescale Select Bits. These bits select the divider tap used to generate an interrupt. Table 6–3 shows timing data with the assumption that the WDCLK is running at a nominal frequency of 16 384 Hz or 15 625 Hz.

Table 6–3. Real-Time Interrupt Selections

RTI Prescale Select Bits				16.384 kHz WDCLK†		15.625 kHz WDCLK‡	
RTIPS2	RTIPS1	RTIPS0	WDCLK Divider	Frequency (Hz)	Overflow Time	Frequency (Hz)	Overflow Time
0	0	0	4	4096	244.14 µs	3906.25	256 µs
0	0	1	16	1024	976.56 µs	976.56	1.024 ms
0	1	0	64	256	3.91 ms	244.14	4.096 ms
0	1	1	128	128	7.81 ms	122.07	8.192 ms
1	0	0	256	64	15.63 ms	61.04	16.384 ms
1	0	1	512	32	31.25 ms	30.52	32.768 ms
1	1	0	2048	8	125.00 ms	7.63	131.072 ms
1	1	1	16 384	1	1.0 second	0.95	1.049 second

† Can be generated by a 4.194 MHz crystal.
‡ Can be generated by a 4.00 MHz crystal.

6.3.5 WD Timer Control Register (WDCR)

Figure 6–7. WD Timer Control Register (WDCR) — Address 7029h

7	6	5	4	3	2	1	0
WD FLAG	WDDIS	WDCHK2	WDCHK1	WDCHK0	WDPS2	WDPS1	WDPS0
RW–x		RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

Note: R = Read access, W = Write access, –n = value after reset (x = indeterminate)

Note: WDCR Required Values

Any values written to WDCR must include the value 101_2 written to bits 5–3 (bits WDCHK2 – WDCHK0).

Bit 7 **WD FLAG**. Watchdog Flag Bit. This bit indicates whether a system reset was asserted by the WD timer. The bit is set to 1 by a WD-generated reset. It is unaffected by any other type of system reset.

0 = Indicates that the WD timer has not asserted a reset since the bit was last cleared.
1 = Indicates that the WD timer has asserted a reset since the bit was last cleared.

Bit 6 **WDDIS**. Watchdog Disable. This bit is valid (available to the user) only when the HP0 bit in the SYSCR (system control register) is set. This only occurs if pin V_{CCP} is at 5V during the device reset sequence. See System Functions chapter in *TMS320C24x DSP Controllers Reference Set, Volume 1: CPU, System, and Instruction Set*.

0 = Watchdog is enabled.
1 = Watchdog is disabled.

Bit 5 **WDCHK2**. Watchdog Check Bit 2. This bit must be written as a 1 when you write to WDCR, or else a system reset is asserted. This bit is always read as 0.

0 = System reset is asserted.
1 = Normal operation continues if all check bits are written correctly.

Bit 4 **WDCHK1**. Watchdog Check Bit 1. This bit must be written as a 0 when you write to WDCR, or else a system reset is asserted. This bit is always read as 0.

0 = Normal operation continues if all check bits are written correctly.
1 = System reset is asserted.

Bit 3 **WDCHK0**. Watchdog Check Bit 0. This bit must be written as a 1 when you write to WDCR, or else a system reset is asserted. This bit is always read as 0.

0 = System reset is asserted.
1 = Normal operation continues if all check bits are written correctly.

Bits 2–0 **WDPS2–WDPS0**. Watchdog Prescale Select Bits. These bits select the counter overflow tap that generates a reset. Each selection sets up the maximum time elapsed before servicing the WD key logic. All timing assumes that the WDCLK is running at 16 384 Hz. Because the WD timer counts 257 clocks before overflowing, the times given are the minimum for overflow (reset). The maximum timeout can be up to 1/256 longer than the times listed in Table 6–4 because of the added uncertainty resulting from not clearing the prescaler.

Table 6–4. WD Overflow (Timeout) Selections

WD Prescale Select Bits				16.384 kHz WDCLK‡		15.625 kHz WDCLK§	
WDPS2	WDPS1	WDPS0	WDCLK Divider	Frequency (Hz)	Minimum Overflow	Frequency (Hz)	Minimum Overflow
0	0	X†	1	64	15.63 ms	61.04	16.38 ms
0	1	0	2	32	31.25 ms	30.52	32.77 ms
0	1	1	4	16	62.50 ms	15.26	65.54 ms
1	0	0	8	8	125.00 ms	7.63	131.07 ms
1	0	1	16	4	250.00 ms	3.81	262.14 ms
1	1	0	32	2	500.00 ms	1.91	524.29 ms
1	1	1	64	1	1.0 second	0.95	1.05 second

† X = Don't care
‡ Can be generated by a 4.194 MHz crystal.
§ Can be generated by a 4.00 MHz crystal.

6.4 Watchdog (WD) and Real-Time Interrupt (RTI) Routines

Example 6–1 shows a WD/RTI enable routine and Example 6–2 shows a WD/RTI disable routine for the DSP controller.

Example 6–1. Watchdog (WD) and Real-Time Interrupt (RTI) Enable Routine

```
;***********************************************************************
; File Name:   Watchdog Enable Example Code
;
; Description: The sample code below demonstrates the software required
;              to enable the watchdog with an overflow time of 1.05
;              seconds. The watchdog counter must be reset before the
;              counter overflows in order to avoid a watchdog reset.
;              The KICK_DOG macro is used to reset the watchdog counter.
;              This macro should be placed throughout the main body of
;              code to ensure that the counter is indeed reset.
;
;              Note:  The code listed below is not a complete program.
;                     It only demonstrates the steps required to enable
;                     the watchdog.
;-----------------------------------------------------------------------
; Macro definitions
;-----------------------------------------------------------------------
KICK_DOG     .macro             ;watchdog reset macro
             LDP    #00E0h
             SPLK   #05555h, WDKEY
             SPLK   #0AAAAh, WDKEY
             LDP    #0h
             .endm
;=====================================================================
; M A I N   C O D E  - starts here
;=====================================================================
             .text
START:       LDP    #00E0h
             SPLK   #002Fh, WDCR ;Enable watchdog w/max. overflow
             KICK_DOG           ;Reset watchdog counter
;=====================================================================
; Main Program Routine
;=====================================================================
MAIN:                           ;Main loop of code begins here
;            ...                ;Insert actual code here
             KICK_DOG           ;Reset watchdog counter
;            ....               ;Insert actual code here
             KICK_DOG           ;Reset watchdog counter
       B     MAIN               ;The MAIN program loop has completed
                                ;one pass, branch back to the
                                ;beginning and continue.
```

Example 6–2. Watchdog (WD) and Real-Time Interrupt (RTI) Disable Routine

```
;*********************************************************************
; File Name:   Watchdog Disable Example Code
;
; Description: The sample code below demonstrates the software required
;              to disable the watchdog when 5 volts is applied to the
;              Vccp pin on the device.  After the watchdog has been
;              disabled, the watchdog counter must be reset in order to
;              clear the pending interrupt.  The KICK_DOG macro is used
;              to reset the watchdog counter.
;              Note:  The code listed below is not a complete program.
;                     It only demonstrates the steps required to disable
;                     the watchdog.
;---------------------------------------------------------------------
; Macro definitions
;---------------------------------------------------------------------
KICK_DOG      .macro              ;Watchdog reset macro
              LDP    #00E0h
              SPLK   #055h, WDKEY
              SPLK   #0AAh, WDKEY
              LDP    #0h
              .endm
;===================================================================
; M A I N   C O D E  - starts here
;===================================================================
              .text
START:        LDP    #00E0h
              SPLK   #006Fh, WDCR ;Disable watchdog if VCCP=5V
              KICK_DOG            ;Reset Watchdog counter
;===================================================================
; Main Program Routine
;===================================================================
MAIN:                             ;Main loop of code begins here ...
;                                 ;insert actual code here
              NOP
              NOP
              NOP
;             ....                ;insert actual code here
       B      MAIN                ;The MAIN program loop has completed
                                  ;one pass, branch back to the
                                  ;beginning and continue.
```

Chapter 7

Flash Memory Module

This chapter describes how the flash EEPROM module is used and how to erase and program the flash array. Within this discussion, the flash EEPROM may be referred to as simply flash or flash module. The term flash array refers to the actual memory array within the flash module.

The flash EEPROM module provides an attractive alternative to masked ROM program memory. Like ROM, flash EEPROM is also a nonvolatile memory type, however it has the advantage of “in-target” reprogrammability both on the production floor or in the field.

Topic **Page**

7.1 Flash EEPROM Overview

The following are the available features and options for a single block module of flash. In an actual ’C24x device specific implementation, more than one block may be utilized to give a larger overall flash memory capacity. For specific ’C24x device details, see Chapter11, *TMS320C240 DSP Controller*.

- ❑ Organization: 4K / 8K / 16K / 32K
- ❑ Word ($\times$16 bits) implementation
- ❑ Segmented with up to eight sections for selective erasure
- ❑ Low power mode
- ❑ Access rate supports 50 ns CPU machine cycles with no wait states
- ❑ Data retention: 10 years at 55^{o}C T_j
- ❑ Write/Erase endurance pro-rated for smaller array sizes for data space applications
- ❑ Write/Erase performed by DSP core

The flash EEPROM module can contain up to 32K$\times$16 bits of electrically erasable, electrically programmable read-only memory. It may be used to replace masked ROM or single-access RAM (SARAM) and may be mapped to either program or data space but not both simultaneously. The block size determines the resolution of the start address boundary. For example, an 8K$\times$16 bit block starts on an 8K-word boundary. The flash is implemented and accessed in word-wide ($\times$16) blocks.

The flash module is erased and programmed by the DSP core itself. This allows the application code to manage the use of the flash memory without the requirement of external programming equipment. The initial programming of the flash may be done using the XDS510 scan-based emulator by scanning in the erase/program algorithm and data into on-chip RAM. The segmentation of the flash allows part of the memory block to be erased while maintaining the rest of the programmed data. The module is segmented into eight individually erasable and reprogrammable segments. However, multiple segments may also be erased when required.

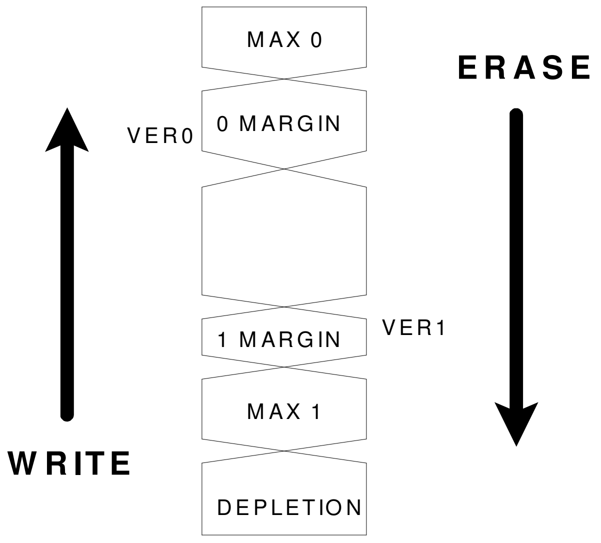
The flash module interfaces to the regular DSP memory interface. This allows the design to take advantage of performance gains provided by the Harvard architecture of the ’C24x DSP core.

The flash module includes both the flash array and the necessary control registers to erase and program the array.

7.2 Fundamental Concepts

Erasing the flash bits is accomplished by adjusting the charge on all the array (or segment) bits to a level so that they are read as ones. Writing the bits is accomplished by adjusting the charge on individual bits to a level so that they are read as zeros. Figure 7–1 shows this mechanism. Although the write operation adds charge, the written bit is read as zero. The erase operation removes charge from the bits thus the orientation of the arrows.

Figure 7–1. Flash Bit Programming



Erasing is a block operation. It simultaneously moves all the bits within the selected segments of an array. Erasure is complete when all bits erase to a point where the application can consistently read them as ones (1 MARGIN in Figure 7–1). Due to minor process variations across the array, bits do not erase at the same rate. Occasionally, a fast erasing bit may erase beyond the valid threshold into depletion at the same time other bits reach the valid read margin. These bits may be returned to the 1 margin range using the flash–write operation.

The flash bits are addressed on word boundaries with respect to the write operation to allow various word patterns to be loaded into the flash. Although all 16 bits are addressed on a flash word boundary, only 8 bits may be written at a time due to current limitations in the flash pump. The algorithm limits the write to 8 bits by masking the word to be written into the array.

The programming algorithms use leveling techniques to provide a balanced erasure and writing of the array bits. This balance provides a measured margin for the coded bit patterns programmed within a minimum time. It also matches the device production test programming levels to the application programming levels. The leveling mechanism provides special array read modes (VER1 and VER0) to verify that bits are erase/written to a level beyond what is necessary to meet the specified operational voltage range of the device. The extent beyond the operation level assures data retention for the life of the application and is tested in production test flow of each device.

The leveling technique steps the erase or write operations in increments to align closer to the margins applied. Due to process variances, some bits program faster than others. All bits must be programmed to the specified margins. These smaller incremental steps are used instead of a single larger step to align closer to the applied margins on a bit-by-bit basis. This better alignment to the margin has the primary effect of taking less time to erase and write and the secondary effect of reducing the time for the opposing write and erase operations. This incremental method aligns well to DSP devices with embedded flash memory, where an embedded DSP core is capable of managing the adaptive algorithm; it would be more costly in a discrete flash module due to the added complexity of the programming state machine.

The array, in a coded state, includes an assortment of ones and zeros of the coded patterns. The clearing step levels the array to all zeros before erasing. This is necessary so that the erase operation processes the bits evenly. If the array is not cleared prior to erasure, 1 bits will be erased further than 0 bits. This makes it more difficult to erase programmed bits to an adequate margin without pushing already erased bits into depletion. Although it may be possible to erase the array without the clear step, it is likely to take much longer due to the flash–write operation necessary to remove the overerased bits from depletion. The program step will also take longer to reprogram the deeper erased bits. This conflicts with the minimization of the programming time. It is also possible to reduce the erasure margin but this conflicts the reliability criteria.

The boost step maintains the 0 margin in conjunction with segment and partial programming (when part of an array is coded) operations. When other segments of the array are being erased and recoded, it slightly stresses bits in protected segments. The program margins on protected segments may be reduced after numerous clear, erase, and code operations on other, unprotected segments. In this case, the programming algorithm should include the boost operation to assure that the margins in the already coded areas remain robust. The boost operation should be executed any time that part of the array is written.

7.2.1 Erasing

The basic flow of the erase algorithm involves the following key steps:

1) Erasing the array to the VER1 margin
2) Inverse-erase verification to identify bits clearly in depletion
3) Flash–write pulses to recover depletion bits
4) Additional flash–write pulse to adjust bits near depletion

Erasing the flash array (Step 1) employs a leveling technique as discussed in Section 7.2, *Fundamental Concepts*. The special margin read mode, used in erasure, is called VER1. VER1 verifies that the erased array reads 1s at a much lower voltage than the V_{CC} specification of the device (approximately 3.5V). The effective minimum for the erase pulse is 5ms due to the flash pump operation. In previous revisions of the flash module, a longer pulse produced quicker erasures by limiting the overhead associated with the generation of the erase pulse to lower pulse counts. However, characterization of the current revision of the array indicates the minimum erase pulse of 5ms pulse provides the best balance for erasure.

The inverse-erase verification test (Step 2) identifies bits that are erased into depletion. This test identifies depletion bits on bit line boundaries instead of individual bits. This carries the advantage of checking the full array for depletion bits by checking only 32 words of the first row. The inverse-erase test is executed after the array is successfully erased and passes VER1 to minimize overhead associated with the margin test. If the inverse-erase test indicates a depletion bit then the algorithm recovers the bit using an additional flash–write pulse (Step 4) to adjust bits near depletion. If the inverse-erase test passes, flash–write (Step 3) is unnecessary and skipped.

It is difficult (and sometimes impossible) to write an 0 to a bit that has been erased into depletion. In many flash devices, depletion indicates device failure. However, this array employs a flash–write (Step 3) to recover bits that have erased into depletion. The flash–write feature recovers depleted bits without adversely affecting other erased bits. Flash–write, like erase, operates on all unprotected segments and therefore recovers all depleted bits. The flash–write operation also employs shorter incremental steps to recover instead of single long steps. Excessive numbers of flash–write pulses stresses the word line of the flash row. For this reason, the number of flash–writes is limited to align to the word line stress exercised in device test.

Segment erasure is required when reprogramming only parts of the array. This is done by masking protected segments while erasing the segments to be reprogrammed. The number of erase pulses for a given segment is about the same as the number of pulses required for the entire array.

As discussed previously, the inverse-erase test operates on bit lines. These bit lines cross segment boundaries so the inverse–erase test does not identify the segment where the depletion bit resides. This is an issue because the inverse-erase test is a affected by temperature and V_{DD} level and therefore, it is possible that a bit that passed inverse-erase in the full array erase might fail later in a different environment. For this reason the algorithm adds one flash write operation (Step 2) between erasing to VER1 margin and inverse erase to push bits that are close to depletion far enough away to avoid the environment changes issue. Another way to address this issue is to flash–write the complete array when inverse-erase indicates a depleted bit in segment erase operations. As long as the above discussed flash–write limit is in place, either method (or both) may be used.

7.2.2 Writing

Bits are written in each of the clear, code, and boost steps. This write operation is the same for each of these steps. The difference between these step involves the source of the data to be programmed which does not affect the algorithm used in the writing.

In order to minimize application costs, the embedded flash modules include charge pumps to provide the higher voltages required for write operations so that the device may operate with a 5V supply. In order to minimize the cost of the device, these charge pumps are designed to provide enough current to program up to 8 bits at a time. It is possible to design pumps capable of programming more bits at a time but these pumps are significantly larger and cost more.

The bit writing method, described in this document, uses a margin read to provide robust writing of the bits. That is to say, it reads the bits at a voltage greater than what is specified for the application to assure adequate margin for the life of the application. The array design provides a special read mode (VER0) that changes the voltage at the array cell to approximately 6.5V. After a word has been written, it is read back in VER0 mode to assure that the programming level still reads a zero beyond the V_{DD} maximum. Any bits that do not meet this margin are written again. However, bits that do meet the VER0 read are masked on the second write to assure balanced programming.

Section 7.2, *Fundamental Concepts*, discusses the use of incremental smaller steps for writing the bits instead of one larger step to better align to the margin. The write pulse used here is set to 100μs based upon characterization over all existing revisions of the flash module. Future revisions program faster. In the future, a shorter write pulse provides cost reduction in application manufacture due to reduced programming time.

7.3 Flash Registers

The flash module includes four registers used to control erasing, programming, and testing of the flash array. The DSP core accesses these registers over the same buses as it accesses the flash array. The access to the flash registers (known as register access mode) is enabled by activating an OUT command. The OUT FF0Fh instruction makes the flash registers accessible for reads and/or writes. After executing OUT FF0Fh, the registers are accessed in the memory space decoded for the flash module and the flash array cannot be accessed. The four registers are repeated every four address locations within the flash modules decoded range. After completing all the necessary reads and/or writes to the control registers, an IN FF0Fh instruction is executed to place the flash array back in array access mode. After executing IN FF0Fh, the flash array is accessed in the decoded space and the flash registers are not available. The four registers are described in Table 7–1.

Table 7–1. Addresses of Flash EEPROM Module Registers

				Described in	
Address	**Register**	**Name**	**Description**	**Subsection**	**Page**
0	SEG_CTR	Flash Segment Control Register	The 8 MSBs enable specific segments for erasure or programing. Setting a bit to 1 enables the block. The 8 LSBs control the program, erase, and verify operations of the module.	7.3.1	7-7
1	TST	Flash Test Register	Reserved for test and is not accessible during normal operational modes.	7.3.2	7-10
2	WADRS	Write Address Register	Holds the address for a write operation.	7.3.3	7-10
3	WDATA	Write Data Register	Holds the data for a write operation.	7.3.4	7-10

7.3.1 Flash Segment Control Register (SEG_CTR)

The SEG_CTR is a 16-bit register used to initiate and monitor the programming and erasing of the flash array. This register includes the signals necessary to initiate the active operations (WRITE, ERASE, and EXE), those used for verification (VER0 and VER1), and those used for protection (KEY0, KEY1, and SEG7–SEG0). The PWR_RST signal clears all bits of SEG_CTR to 0.

The WRITE signal initiates the programming operation. However, the modification of the flash EEPROM array data does not actually actively start until the

EXE is activated and the KILL_EXE signals is deactivated. The WRITE, KEY1, and appropriate SEG0–7 signals must be set high while all other signals set low for the programming operation to begin. Section 7.4, *Programming the Flash Array*, on page 7-11 describes the programming sequence.

The ERASE operation is done in a similar manner except that the ERASE signal is high and the WRITE signal is low. The FLASH–WRITE operate requires both the WRITE and ERASE signals be active high. Section 7.5, *Erasing the Flash Array*, on page 7-14 describes the erase and flash–write sequences.

Once the EXE bit is active, all register bits, other than the control bits are latched and protected. The user must deactivate the EXE bits to modify the SEG7–0 bits. This protects the array from an inadvertent change of the protected segments. However, it also means the unprotected segments cannot be masked in the same register load with the deactivation of EXE.

The SEG_CTR is shown in Figure 7–2 and descriptions of the bits follow the figure.

Figure 7–2. Segment Control Register (SEG_CTR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2–1	0
SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0	Res	KEY1	KEY0	VER0	VER1	WRITE/ ERASE	EXE
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0		RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

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

Bits 15–8 **SEG7–SEG0**. Flash segment enable bits. Each segment enable bit is used to protect or enable write and erase operations for each of the segments in the flash array.

SEG7–SEG0 Bits									
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**Flash Array Segment Which is Enabled**†	
0	0	0	0	0	0	0	1	0000 – 07FFh	(segment 0)
0	0	0	0	0	0	1	0	0800 – 0FFFh	(segment 1)
0	0	0	0	0	1	0	0	1000 – 17FFh	(segment 2)
0	0	0	0	1	0	0	0	1800 – 1FFFh	(segment 3)
0	0	0	1	0	0	0	0	2000 – 27FFh	(segment 4)
0	0	1	0	0	0	0	0	2800 – 2FFFh	(segment 5)
0	1	0	0	0	0	0	0	3000 – 37FFh	(segment 6)
1	0	0	0	0	0	0	0	3800 – 3FFFh	(segment 7)

† Any number of segments (from 1 to 8 or any combination) can be enabled at any one time.

Bit 7	**Reserved**.
Bits 6–5	**KEY1, KEY0**. Execute key bits. These bits must be written as 1 0 in the same DSP core access as the EXE bit is set for the EXE operation to start. These bits are used as additional protection against inadvertent programming or erasure of the flash. These bits are read as zeros.
Bits 4–3	**VER0, VER1**. Verify bits. The DSP core verifies proper erasure and programming by monitoring these bits. The 1s verify test is used to verify that the array is sufficiently erased. This operation mode lowers the voltage on the cell gate during the read. If the bits still read as 1, the erasure provides sufficient margin for reliable read and program operations. The 0s verify test is used to verify that a given bit has been sufficiently programmed. This operation mode raises the voltage at the cell gate. If the bits still read as 0, the programming provides sufficient margin for reliable read operations. Possible modes for bits 4 and 3: 0 0 = Normal read 0 1 = Read using low-sense voltage to verify margin of 1s 1 0 = Read using high-sense voltage to verify margin of 0s 1 1 = Inverse-erase mode; tests for bits erased to depletion
Bits 2–1	**WRITE/ERASE**. Possible modes for bits 2 and 1: 0 0 = Undefined 0 1 = ERASE 1 0 = WRITE 1 1 = FLASH–WRITE
Bit 0	**EXE**. Execute bit. This bit starts and stops programming and erasing to the flash array. The EXE bit and the KEY0/1 bits must be written in the same write access. When the programming time has expired, the EXE bit should be cleared in the same write as the KEY0/1 bits. The data and address latches are locked whenever the EXE bit is set and all attempts to read from or write to the array will be ignored (read data will be indeterminate).

7.3.2 Flash Test Register (TST)

The flash test register (TST) is a 5-bit register used for testing the flash array. The details of the test operations do not pertain to the customer responsibilities for DSP design and test and are therefore not discussed. This register is not accessible to the DSP core unless the DSP core is in test mode.

7.3.3 Write Address Register (WADRS)

The write address register (WADRS) is a 16-bit register that contains the latched write address for a programming operation. This register is loaded with the value on the address bus when writing a data value to the flash module when in array access mode. It may also be loaded by writing to this register when in register access mode.

7.3.4 Write Data Register (WDATA)

The write address register (WDATA) is a 16-bit register that contains the latched write data for a programming operation. This register may be loaded by writing a data value to the flash module when in array access mode or by writing to this register when in register access mode. The WDATA must be loaded with the value FFFFh before the erase operation starts. If this register is not loaded with this value, the erase operation will not begin.

7.4 Programming the Flash Array

The flash array is programmed by code running on the DSP core. This code may originate from off-chip memory, on-chip ROM, or be loaded into on-chip RAM. The 'C24x core includes on-chip RAM to hold the programming code. The XDS510 emulator may be used to scan the programming program into the DSP core RAM.

The procedure for programming the FLASH module is controlled by SEG_CTR. The EXE bit initiates a programming operation when set to 1 and disables a programming operation when cleared to 0.

After programming, the programmer should wait $td_{(BUSY–WRITE)}$ before attempting a read or another write to the FLASH. No more than 8 bits may be programmed to 0 at one time. This is a current limitation of the voltage supply used to program the array. The data provided to the array is 16-bits wide during programming. The programming algorithm must assure no more than 8 bits are low in this word. In the case of a word that contains more than 8 low bits, the programming should be staged so that some of the low bits are masked in the first pass of the programming. On the second pass of the programming, the already programmed bits should be masked so the remaining bits are programmed. Reads, between a write to the FLASH and a programming operation, will not corrupt the data or address to be used for the programming operation.

Note:

In the case of a verification error, the algorithm requires that the successfully programmed bits be masked before attempting to reprogram the error bits. This assures even programming of the word so that, should the word be erased, it will be evenly erased.

The sequence used to program the flash array is shown in Table 7–2. The sequence assumes that the array elements, to be programmed, are currently erased. If this is not the case, the array data should be checked against the bits to be programmed to mask already programmed bits. This assures even programming of the word so that, should the word be erased, it will be evenly erased. Words are programmed one byte at a time. This is to simplify the checking for more than 8 low bits in a word. The programming operation is accelerated by specifically checking the words and using the two pass programming only where necessary.

Table 7–2. Steps for Programming the Flash Array

Step	Action	Description
1	Power up the V_{CCP} terminal.	The V_{CCP} terminal does not need to be powered to read the array. You may choose to disconnect this terminal in normal operation to protect the array from inadvertent programming or erasure. However, you may choose to leave this signal connected in which case this step is not necessary.
2	Load address and data registers.	Use either of two methods: (a) If the flash module is in array access mode (following an IN FF0Fh instruction), write the data to be programmed to its address; or, (b) If the flash module is in register access mode (following an OUT FF0Fh instruction), load the individual registers. A maximum of 8 bits may be programmed on a given write operation. Therefore, the algorithm should mask one byte while programming the other.
3	Enable specific segment.	Set the corresponding bit in SEG_CTR for the segment that the programmed word resides.
4	Activate the WRITE mode.	Set the WRITE bit in SEG_CTR. NOTE: Steps 3 and 4 may be done in the same CPU write operation to SEG_CTR. For example, loading SEG_CTR with 0104h would initiate the write charge pumps and enable segment 0.
5	Wait for write pump charge time.	The CPU executes a delay loop for the $td_{(WRITE–PUMP)}$ time period.
6	Activate the EXEBIN mode.	Load the EXE, KEY1, and KEY0 bits with 1, 1, 0 respectively. All three bits must be loaded in the same write operation. NOTE: The segment enable bits and WRITE bit must also be maintained. For example, loading SEG_CTR with 0145h would maintain the WRITE and segment bits and also initiate the write execution.
7	Delay for write operation time.	The CPU executes a delay loop to delay the $td_{(WRITE)}$ time period.
8	Deactivate WRITE and EXEBIN modes.	Clear the WRITE and EXE bits in SEG_CTR. For example, loading SEG_CTR with 0100h will deactivate these two signals.
9	Delay for array stabilization time.	The CPU executes a delay loop to delay for the $td_{(BUSY–WRITE)}$ time period.
10	Activate verify 0 mode.	Set the VER0 bit in SEG_CTR. For example, loading SEG_CTR with 0110h will initiate the verify 0 operation.
11	Delay for verify 0 voltage stabilization.	The CPU executes a delay loop to delay for the $td_{(BUSY–VERIFY)}$ time period.
12	Read location to verify write operation.	The CPU reads the programmed location. The flash module must be in array access mode (following an IN FF0Fh instruction).

Note: See Table 7–4 on page 7-17 for typical values of timing parameters.

Table 7–2. Steps for Programming the Flash Array (Continued)

Step	Action	Description
13	Deactivate the verify 0 mode.	Clear the VER0 bit in SEG_CTR. For example, loading SEG_CTR with 0100h will deactivate the verify 0 operation.
14	Verify programmed data.	**Option 1:** If the verification fails. If the data read during the verify 0 operation is not the expected programmed value, then the programming algorithm should mask off the bits that did program correctly and start again at step 4 above. The programming algorithm may have a program failure counter and trap to an error routine on too many programming failures.
		Option 2: If the verification passes, program the next byte. If the data read during the verify 0 operation is correct, the programming algorithm checks if both bytes of the word are programmed. If both bytes are not programmed, then the second byte is selected and the algorithm returns to Step 2.
		Option 3: If the verification passes, program the next word. If the data (read during the verify 0 operation) is correct, then the programming algorithm checks to see if there are more words to program. If there are, than the next value is chosen and the algorithm returns to Step 2.
		Option 4: If the verification passes, last word. If the data (read during the verify 0 operation) is correct, then the programming algorithm checks to see if there are more words to program. If not than the segment enable bit may be cleared by writing 0 to SEG_CTR.

Note: See Table 7–4 on page 7-17 for typical values of timing parameters.

7.5 Erasing the Flash Array

The flash array is erased by code running on the DSP core. This code may originate from off-chip memory, on-chip ROM, or be loaded into on-chip RAM. The 'C24x core includes on-chip RAM to hold the erasing code. The XDS510 emulator may be used to scan the erasing program into the DSP core RAM.

Before the flash array may be erased, all of the bytes in each segment to be erased should be programmed to zeros. This assures that they are erased evenly and reduces the likelihood of a bit requiring reprogramming (see Table 7–2, Step 14). Next the segments are erased with pulses until all bits are erased. The WDATA must be loaded with the value FFFFh before the erase operation. If this register is not loaded with this value, then the erase operation will not begin. A special threshold voltage is used during the verification to ensure proper erasure. After erasing the memory segment, new data can be programmed as previously described. The sequence used to erase the flash array is shown in Table 7–3.

Table 7–3 assumes the array has previously been cleared (programmed to all zeros). The devices are erased before shipping. Therefore, you may choose to check for erasure before starting the erase operation. The erase operation sequence is as follows:

1) Check for erasure using VER1 mode. If erased then done.
2) Check for erasure using normal read mode. If erased then go to Step 4.
3) Program all bits to 0 as described in Section 7.4, *Programming the Flash Array*, on page 7-11.
4) Erase the array as described in Table 7–3.

This sequence avoids unnecessary clear and erase operations.

The erase operation includes a check for overerasure called inverse erase. After each erase pulse, the array is checked for overerasure. If the inverse erase test indicates overerasure, the program executes a flash–write operation to level the bits that erase faster.

Table 7–3. Steps for Erasing the Flash Array

Step	Action	Description
1	Power to the V_{CCP} terminal.	The V_{CCP} terminal does not need to be powered to read the array. You may choose to disconnect this terminal in normal operation to protect the array from inadvertent programming or erasure. However, you may choose to leave this signal connected in which case this step is not necessary.
2	Enable specific segment.	This is done by setting the corresponding bit in SEG_CTR for the segment that the programmed word resides.
3	Load WDATA with FFFF pattern.	This is done to fulfill the protection mechanism against inadvertent writes.
4	Activate the ERASE mode.	This is done by setting the ERASE bit in SEG_CTR. Steps 2 and 4 may be done in the same CPU write to SEG_CTR. For example: loading SEG_CTR with 0102h would initiate the erase setup operations and enable segment 0.
5	Wait for erase setup delay time.	CPU executes a delay loop to delay for $td_{(ERASE–SETUP)}$ time period.
6	Activate the EXEBIN mode.	This is done by loading the EXE, KEY1, and KEY0 bits with 1, 1, 0 respectively. All three bits must be loaded in the same write operation. Note also that the segment enable bits and ERASE bit must also be maintained. For example: loading SEG_CTR with 0143h would maintain the ERASE and segment bits but also initiate the erase execution.
7	Delay for erase operation time.	CPU executes a delay loop to delay for $td_{(ERASE)}$ time period.
8	Deactivate pending command mode.	This is done by clearing all the control bits in SEG_CTR. For example: loading SEG_CTR with 0100h would deactivate these two signals.
9	Delay for array stabilization time.	CPU executes a delay loop to delay for $td_{(BUSY–ERASE)}$ time period.
10	Activate inverse-erase mode.	This is done by setting both the VER0 and VER1 bits in SEG_CTR. For example: loading SEG_CTR with 0118h will initiate the inverse-erase operation.
11	Delay for inverse-erase voltage stabilization.	CPU executes a delay loop to delay for $td_{(BUSY–INVERSE)}$ time period.
12	Read all columns in erased segments to identify any bits that are overerasing.	The CPU reads and verifies every location in the erased segments (checked to 0) to verify against overerasure. Overerasure is identified as a column reading high. For this reason, only the first 32 words are checked. The flash module must be in array access mode (following an IN FF0Fh instruction).

Note: See Table 7–4 on page 7-17 for typical values of timing parameters.

Table 7–3. Steps for Erasing the Flash Array (Continued)

Step	Action	Description
13	Inverse-erase test: pass or fail?	Inverse-erase test passes. Deactivate inverse-erase mode in step 21. No overerased bits are found. Inverse-erase test fails. Proceed with flash write. If a column indicates an error then proceed with flash–write operation to level the bit.
14	Deactivate inverse-erase operation.	Clear out inverse erase operation. Deactivate inverse-erase mode by clearing the lower byte in SEG_CTR. For example: loading SEG_CTR with 0 would deactivate any pending operations Then proceed with verify 1 check.
15	Delay for array stabilization time.	CPU executes a delay loop to delay for $td_{(BUSY–INVERSE)}$ time period.
16	Activate flash–write signals.	This is done by setting the ERASE and WRITE bits in SEG_CTR. For example: loading SEG_CTR with 0106h would initiate the erase setup operations and enable segment 0.
17	Delay for flash–write operation time.	CPU executes a delay loop to delay for $td_{(FLASH–WRITE)}$ time period.
18	Activate the EXEBIN mode.	This is done by loading the EXE, KEY1, and KEY0 bits with 1, 1, 0 respectively. All three bits must be loaded in the same write operation. Note also that the segment enable bits as well as the ERASE and WRITE bits must also be maintained. For example: loading SEG_CTR with 0147h would maintain the WRITE, ERASE, and segment bits but also initiate the erase execution.
19	Delay for erase operation time.	CPU executes a delay loop to delay for $td_{(ERASE)}$ time period.
20	Recheck for overerased bits.	Return to step 8 to check if overerased bits have been successfully recovered.
21	Deactivate any pending operation.	Clear out pending operation. Deactivate inverse-erase or flash write modes by clearing the lower byte in SEG_CTR. For example: loading SEG_CTR with 0 would deactivate any pending operations Then proceed with verify 1 check.
22	Delay for array stabilization time.	CPU executes a delay loop to delay for $td_{(BUSY–INVERSE)}$ time period.
23	Activate verify 1 signal.	This is done by setting the VER1 bit in SEG_CTR. For example: loading SEG_CTR with 0108h will initiate the verify 1 operation.
24	Delay for verify 1 voltage stabilization.	CPU executes a delay loop to delay for $td_{(BUSY–VERIFY)}$ time period.

Note: See Table 7–4 on page 7-17 for typical values of timing parameters.

Table 7–3. Steps for Erasing the Flash Array (Continued)

Step	Action	Description
25	Read all locations in erased segments to verify erase operation.	The CPU reads and verifies every location in the erased segments to verify proper erasure. The flash module must be in array access mode (following an IN FF0Fh instruction).
26	Deactivate the verify 1 signal.	This is done by clearing the VER1 bit in SEG_CTR. For example: loading SEG_CTR with 0100h will deactivate the verify 1 operation.
27	Verify erased data: verification fails.	If the data read during the verify 1 operation is not the expected erased value, then the erase operation has not successfully completed. Return to Step 3. The erasing algorithm may have a program failure counter and trap to an error routine on too many erasure failures.
	Verify programmed data:verification passes.	If the data read during the verify 1 operation is correct, then the erase operation is complete. The segment enable bit may be cleared by writing 0 to SEG_CTR.

Note: See Table 7–4 on page 7-17 for typical values of timing parameters.

Table 7–4. Flash Programming Timing Information

Parameter	Description	MIN	NOM	MAX	UNIT
$td_{(BUSY–WRITE)}$	Post-programming operation stabilization	10			µs
$td_{(BUSY–VERIFY)}$	Sense voltage stabilization after setting verify bits.	10			µs
$td_{(BUSY\_INVERSE)}$	Setup to inverse-erase operation	10			µs
$td_{(BUSY–ERASE)}$	Post-erasing operation stabilization	10			µs
$td_{(BUSY–WAKEUP)}$	Voltage stabilization from low-power mode	10			µs
$td_{(ERASE–SETUP)}$	Erase setup time		10		µs
$td_{(FLASH\_WRITE–SETUP)}$	Flash–write setup time		10		µs
$td_{(WRITE–PUMP)}$	Write pump charge time		10		µs
$td_{(WRITE)}$	Programming initial duration for fast algorithm		100		µs
$td_{(ERASE)}$	Erase time initial duration		50		ms
$td_{(FLASH–WRITE)}$	Flash–write time duration		50		ms

Note: The timing information in this table is typical and is provided as a guide in understanding algorithm requirements. For the latest timing specifications, see the appropriate device data sheet.

7.6 Reading From the Flash Array

Once programmed, the flash array is read in the same manner as other memory on the DSP memory interface. It operates with zero wait states. When reading the flash module, the control register bits should all be zero and the flash array should be in the array access mode by issuing an IN FF0Fh instruction.

The flash array may be protected from inadvertent writes using any of the following mechanisms:

- ❑ Segment enable registers
- ❑ EXE, KEY0, KEY1 bits
- ❑ Connecting V_{CCP} pin to 0 V

The SEG_CTR, described in subsection 7.3.1 on page 7-7, can block writes or erasures to a given segment by clearing the corresponding protect bit. All segment bits are cleared by power-on reset. The flash array write or erase operations are initiated by setting the appropriate values in the EXE, KEY0, and KEY1 bits of SEG_CTR. If this is not done, the array will not be written. The voltage pumps are powered via the V_{CCP} pin. If this pin is driven low, the array may be read but cannot be erased or programmed. The flash logic supplies an additional security against erasure by requiring WDATA to be loaded with FFFFh before erasing.

Chapter 8

External Memory Interface

This chapter contains a general description of the External Memory Interface.

8.1 External Interface to Program Memory

The ‘C24x can address up to 64K words of program memory. While the ‘C24x is accessing the on–chip program memory blocks, the external memory signals $\overline{\text{PS}}$ and $\overline{\text{STRB}}$ are in high impedance. The external data and address bus is active only when the ‘C24x is accessing locations within the address ranges that are mapped to external memory locations. Table 8–1 lists the key signals for external memory interfacing.

Table 8–1. Key Signals for External Interfacing to Program Memory

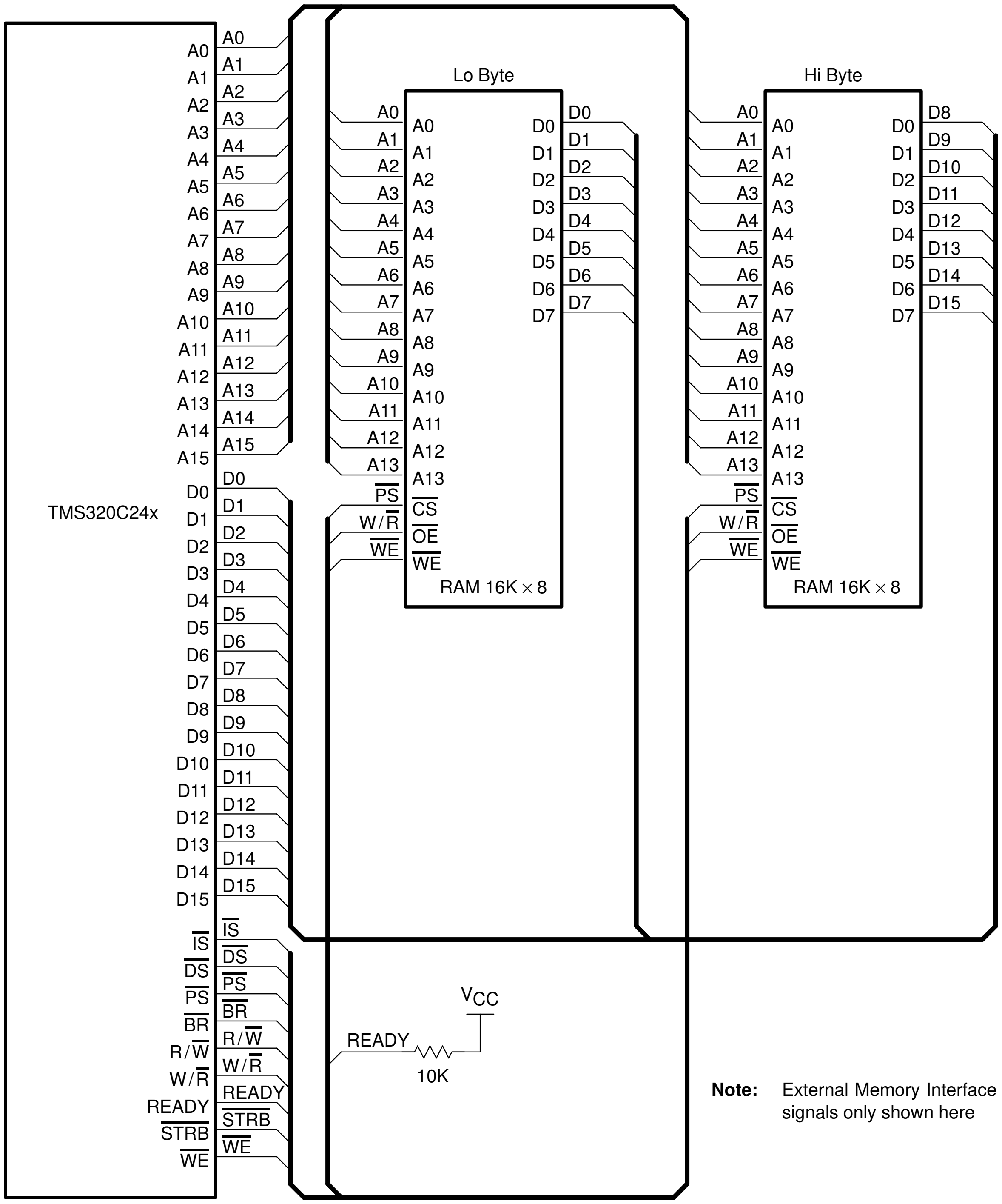
Signal	Description
A15–A0	16-bit bidirectional address bus
BR	Bus request
D15–D0	16-bit bidirectional data bus.
$\overline{\text{PS}}$	Program memory select
READY	Memory ready to complete cycle
R/$\overline{\text{W}}$	Read-not-write signal
$\overline{\text{STRB}}$	External memory access active strobe
$\overline{\text{WE}}$	Write enable signal
W/$\overline{\text{R}}$	Write-not-read signal

Figure 8–1 shows an example of a minimal external program memory interface. In this figure, the ’C24x device interfaces to two 16K $\times$ 8-bit SRAMs. Two 8-bit wide memories are used to implement the 16-bit word width required by the ’C24x. Although SRAMs are shown in Figure 8–1, the interface is equally valid for EPROMs, with the removal of the write enable ($\overline{\text{WE}}$) signal.

The interface shown in Figure 8–1 assumes a zero wait-state read/write cycle, that is, the memory access time has been appropriately chosen (see the ’C24x data sheet for exact bus timing parameters).

If slower memory is used, the on-chip wait-state generator can be used to insert one wait state to the access cycle. If more than one wait state is needed, then external wait-state logic is required which uses the READY signal to extend the bus cycle by the required number of wait states.

Figure 8–1. Interface to External Program Memory



Note: External Memory Interface signals only shown here

The program select ($\overline{\text{PS}}$) signal is connected directly to the chip select ($\overline{\text{CS}}$) to select the memory on any external program access. The memory is addressed in any 16K address block in program space. If multiple blocks of memory are to be interfaced in program space, a decode circuit that gates $\overline{\text{PS}}$ and the appropriate address bits can be used to drive the memory block chip selects.

The W/$\overline{\text{R}}$ signal is tied directly to the output enable ($\overline{\text{OE}}$) pin of the memory. The $\overline{\text{OE}}$ signal enables the output drivers of the memory. The drivers are turned off in time to guarantee that no data bus conflicts occur with an external write by the ’C24x device.

The ’C24x requires two cycles on all external writes, including a half cycle before the $\overline{\text{WE}}$ goes low and a half cycle after $\overline{\text{WE}}$ goes high. This prevents buffer conflicts on the external busses.

8.2 External Interface to Local Data Memory

The ’C24x device can address up to 32K words of off-chip local data memory. Table 8–2 lists the key signals necessary for this interface.

Table 8–2. Key Signals for External Interfacing to Local Data Memory

Signal	Description
A15–A0	16-bit bidirectional address bus
BR	Bus request
D15–D0	16-bit bidirectional data bus
$\overline{\text{DS}}$	Data memory select
READY	Memory ready to complete cycle
R/$\overline{\text{W}}$	Read-not-write signal
$\overline{\text{STRB}}$	External memory access active strobe
$\overline{\text{WE}}$	Write enable signal
W/$\overline{\text{R}}$	Write-not-read signal

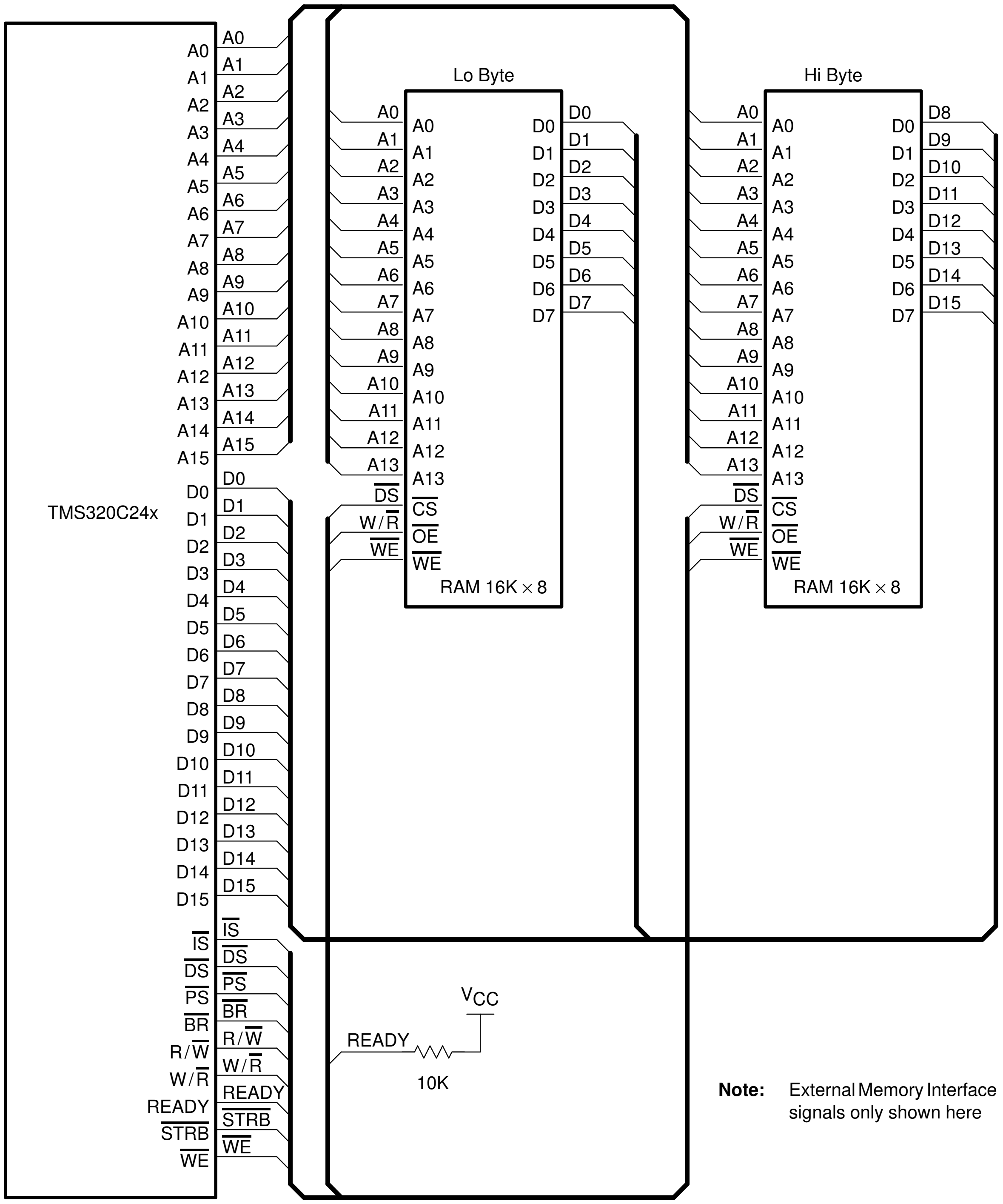
While the ‘C24x is accessing the on-chip data memory blocks, the external signals $\overline{\text{DS}}$ and $\overline{\text{STRB}}$ are in high impedance. The external data bus is active only when the ‘C24x is accessing locations within the address ranges that are mapped to external memory, 8000h–FFFFh. An active $\overline{\text{DS}}$ signal indicates that the external busses are being used for data memory. Whenever the external busses are active (when the external memory is being accessed), the ‘C24x drives the $\overline{\text{STRB}}$ signal low.

For fast interfacing, it is important to select external memory with fast access time. If fast memory access is not required, one can use the READY signal and/or on-chip wait-state generator to create wait states for interfacing with slow external memory devices.

Figure 8–2 shows an example of an external RAM interface. In this figure, the ’C24x device interfaces two 16K $\times$ 8-bit RAM devices. The data memory select ($\overline{\text{DS}}$) is directly connected to the chip select ($\overline{\text{CS}}$) of the devices. This means the external RAM block will be addressed in any of the two 16K banks of local data space, 8000h–FFFFh. If there are additional banks of off-chip data memory, a decode circuit that gates $\overline{\text{DS}}$ with the appropriate address bits can be used to drive the memory block chip set.

W/$\overline{\text{R}}$ signal is tied directly to the output enable ($\overline{\text{OE}}$) pin of the RAMs. This signal enables the output drivers of the RAM and turns them off in time to prevent data bus conflicts with an external write by the ’C24x. The $\overline{\text{WE}}$ signal of RAM is connected to the $\overline{\text{WE}}$ signal of ’C24x. The ’C24x requires at least two cycles on external writes, including a half cycle before $\overline{\text{WE}}$ goes low and a half cycle after $\overline{\text{WE}}$ goes high. This prevents buffer conflicts on the external buses. An additional wait state can be generated with the software wait-state generator.

Figure 8–2. Interface to External Data Memory


TMS320C24x
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15
IS DS PS BR R/W W/R READY STRB WE
Lo Byte
RAM 16K × 8
Hi Byte
RAM 16K × 8
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13
D0 D1 D2 D3 D4 D5 D6 D7
D8 D9 D10 D11 D12 D13 D14 D15
CS OE WE
VCC
READY
10K

Note: External Memory Interface signals only shown here

8.3 Interface to I/O Space

I/O space accesses are distinguished from program and data-memory accesses by $\overline{\text{IS}}$ going low. All 64K I/O words (external I/O ports and on-chip I/O registers) are accessed via the IN and OUT instructions. See Example 8–1 and Example 8–2.

Example 8–1. External I/O Port Access

```
IN DAT7, 0AFEEh ; Read data into data memory from external
                ; device on port 45038.
OUTDAT7, 0CFEFh ; Write data from data memory to external
                ; device on port 53231.
```

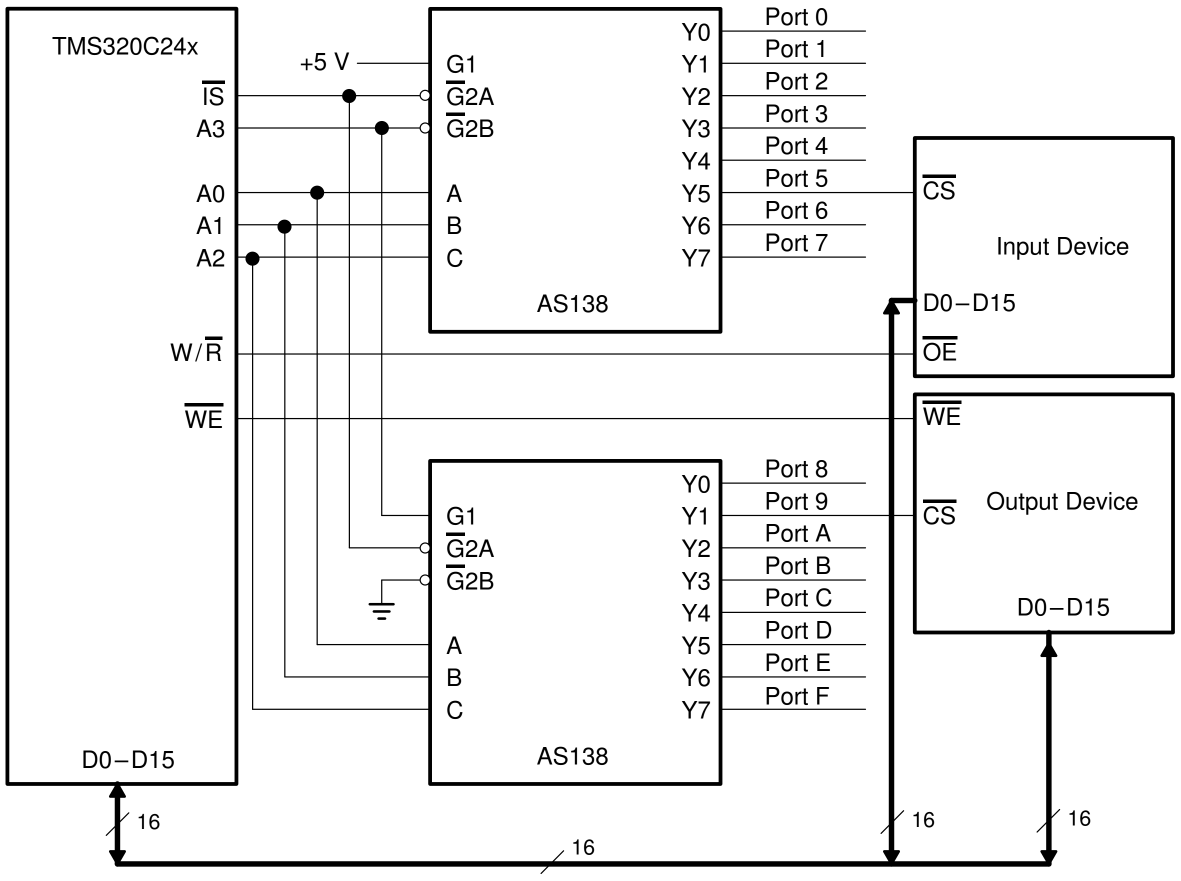
Example 8–2. I/O-Mapped Register Access

```
IN DAT7, FFFFh  ; Read data into data memory from ’C24x to
                ; wait state generator control register
OUTDAT8, FFFFh  ; Write data from data memory to ’C24x
                ; wait state generator control register
```

Access to external parallel I/O ports is multiplexed over the same address and data bus for program and data memory accesses. The data bus is 16 bits wide; however, if you are using 8-bit peripherals, you can use either the higher or lower byte of the data bus to suit a particular application.

W/$\overline{\text{R}}$ can be used with chip-select logic to generate an output enable signal for an external peripheral. The $\overline{\text{WE}}$ signal can be used with chip-select logic to generate a write enable signal for an external peripheral. Figure 8–3 shows an example of interface circuitry for 16 I/O ports. Note that the decode section can be simplified if fewer I/O ports are used.

Figure 8–3. I/O Port Interface



8.4 Memory Interface Timing Diagrams

Figure 8–4 shows the memory interface read waveforms and Figure 8–5 shows the memory interface write waveforms. Both figures are for demonstration purposes only. For accurate timing parameters, see the ’C24x data sheet.

Figure 8–4. Memory Interface Read Waveforms

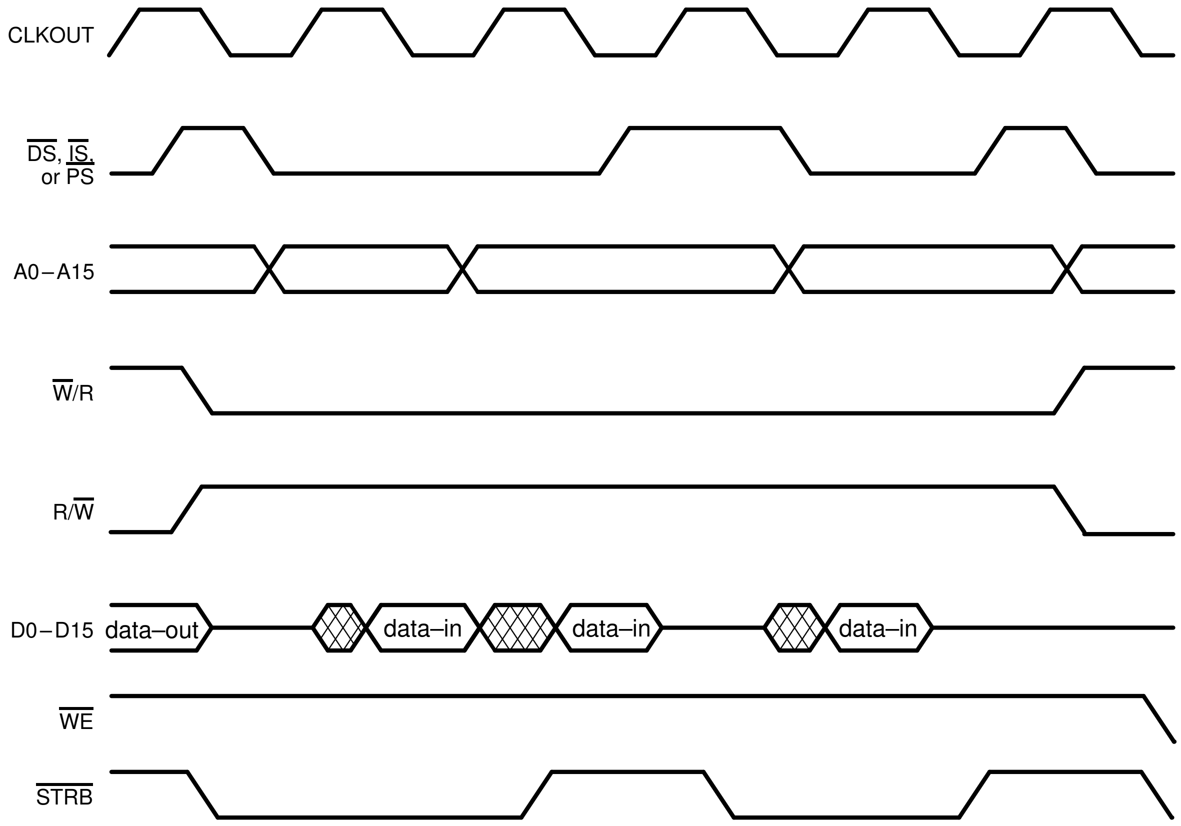
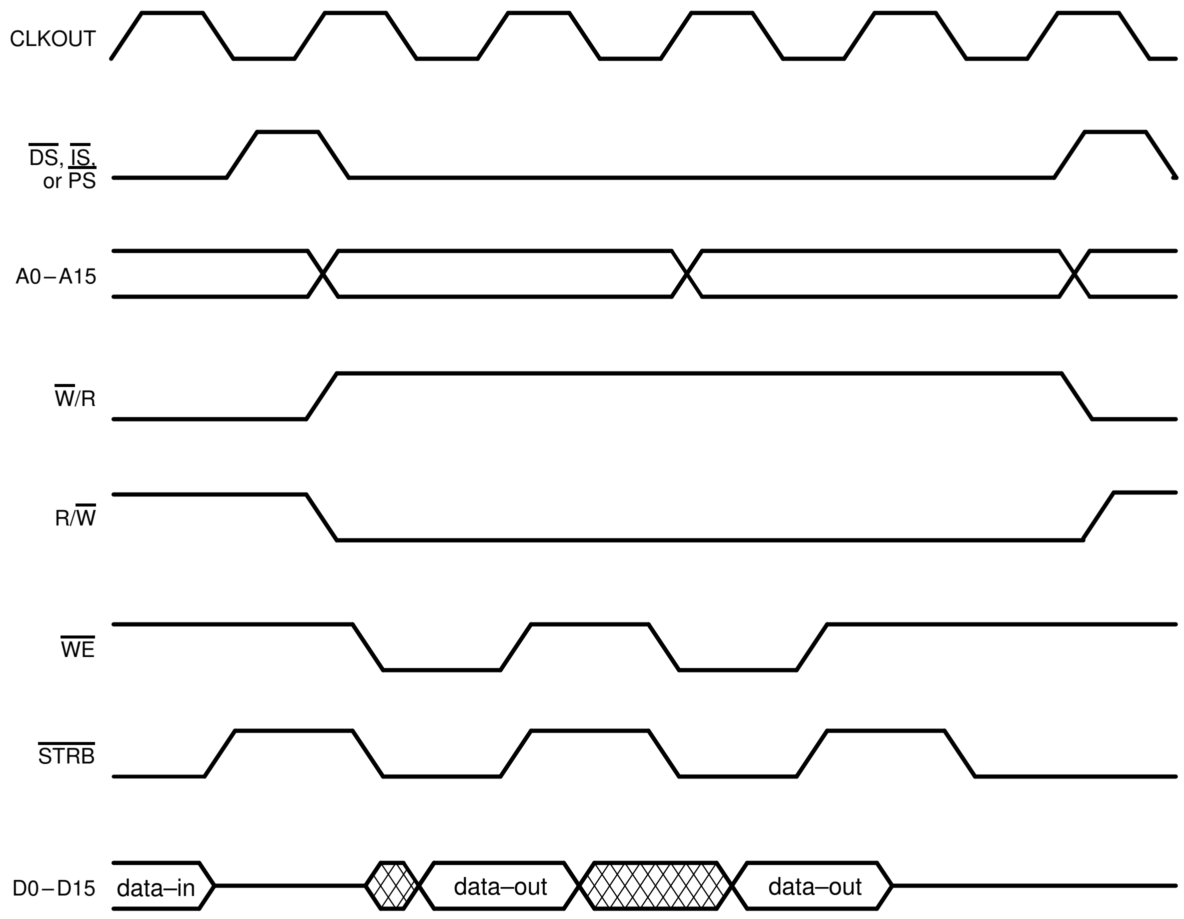


Figure 8–5. Memory Interface Write Waveforms



8.5 Wait-State Generator

Wait states can be generated when accessing slower external resources. Wait states operate on machine-cycle boundaries and are initiated either by using the ready signal or using the software wait-state generator. READY can be used to generate any number of wait states.

8.5.1 Generating Wait States with the READY Signal

By driving the READY signal high, an external device indicates that it is prepared for a bus transaction to be completed. If the external device is not ready, it can keep READY low for as long as it needs. When READY is low, the 'C24x waits one CLKOUT1 and checks READY again. The 'C24x will not continue executing until READY is driven high; therefore, if the READY signal is not used, it should be pulled high during external and internal accesses.

The READY pin can be used to generate any number of wait states. However, even when the 'C24x operates at full speed, it may not be able to respond fast enough to provide a READY-based wait state for the first cycle. In order to be absolutely sure to have a wait state or states immediately, you should use the on-chip wait-state generator first and then the additional wait states can be generated using the READY signal.

8.5.2 Generating Wait States with the Wait-State Generator

The wait-state generator can be programmed to generate the first wait state for a given off-chip memory space (data, program, or I/O), regardless of the state of the READY signal. To control the wait-state generator, read or write to the wait-state generator control register (WSGR), mapped to I/O memory location FFFFh. Figure 8–6 shows the register bit layout.

Figure 8–6. Wait-State Generator Control Register (WSGR)

15–4	3	2	1	0
Reserved	AVIS	ISWS	DSWS	PSWS
0	W–1	W–1	W–1	W–1

Note: W = Write, 0 = read by device as zeros, –1 = value after reset

Bits 15–4 **Reserved**. Always read as 0.

Bit 3	**AVIS**. Address visibility mode. At reset, this bit is set to 1. AVIS does not generate a wait state. 0 = Cleared; reduces power and noise. (for production systems) 1 = Enables the address visibility mode of the device. In this mode, the device provides a method of tracing the internal code operation: it passes the internal program address to the address bus when this bus is not used for an external access.
Bit 2	**ISWS**. I/O space wait state bit. At reset, this bit is set to 1. 0 = No wait states are generated for off-chip I/O space. 1 = One wait state will be applied to all reads from off-chip I/O memory space. (Writes always take two cycles, regardless of PSWS or READY.)
Bit 1	**DSWS**. Data space wait state bit. At reset, this bit is set to 1. 0 = No wait states are generated for off-chip data space. 1 = One wait state will be applied to all reads from off-chip data memory space. (Writes always take two cycles, regardless of DSWS or READY.)
Bit 0	**PSWS**. Program space wait state bit. This bit is set at the end of an A/D conversion (single or dual channel). At reset, this bit is set to 1. 0 = No wait states are generated for off-chip program space. 1 = One wait state will be applied to all reads from off-chip program memory space. (To avoid bus conflicts, a write always takes two cycles, regardless of PSWS or READY.)

In summary, the wait-state generator inserts a wait state to a given memory space (data, program, or I/O) if the corresponding bit in WSGR is set to 1, regardless of the condition of the READY signal. The READY signal can then be used to further extend wait states. The WSGR bits are all set to 1 by reset so that the device can operate from slow memory after reset. To avoid bus conflicts, writes from the wait-state generator always take two CLKOUT1 (internal clock) cycles each.

Chapter 9

Digital I/O Ports

This chapter contains a general description of the Digital I/O Ports module.

Topic **Page**

9.1 Digital I/O Ports Overview

The Digital I/O Ports module provides a flexible method for controlling both dedicated I/O (internal and external to the ’C24x) and shared pin functions. All I/O and shared pin functions are controlled using eight 16-bit registers mapped within Peripheral File 9. These registers are divided into three types:

❑ **Output Control registers** – Used to directly control O/P pins or, internally, to perform chip control functions.

❑ **Input Status registers** – Used to directly monitor the state of I/P pins or, internally, to monitor the status of chip events or conditions.

❑ **Data and Direction Control registers** – Used to control the data and the direction of the data to bidirectional I/O pins. The registers are directly connected to the bidirectional I/O pins.

The Digital I/O Ports module allows a maximum of 32 output/internal control functions, 32 input/internal status functions, and 32 bidirectional I/O pin functions. The total number of available digital I/O pins, control/status functions, and associated registers are device specific. You should refer to the appropriate device specific configuration, Chapter 11, *TMS320C240 DSP Controller*, for the exact number of I/O pins available, pin locations, whether pin functions are shared, naming conventions, and control registers.

9.2 Digital I/O Ports Registers

Table 9–1 lists the registers available to the digital I/O module. As with other 'C24x peripherals, the registers are memory mapped to the data space. Specifically, these registers reside in peripheral frame 7090h through 709Fh.

Table 9–1. Addresses of Digital I/O Ports Registers

			Described in	
Address	**Register**	**Name**	**Subsection**	**Page**
7090h	OCRA	Output Control Register A	9.2.1	9-3
7092h	OCRB	Output Control Register B	9.2.2	9-4
7094h	ISRA	Input Status Register A	9.2.3	9-4
7096h	ISRB	Input Status Register B	9.2.4	9-5
7098h	PADATDIR	I/O Port A Data and Direction Register	9.2.5	9-5
709Ah	PBDATDIR	I/O Port B Data and Direction Register	9.2.5	9-5
709Ch	PCDATDIR	I/O Port C Data and Direction Register	9.2.5	9-5
709Eh	PDDATDIR	I/O Port D Data and Direction Register	9.2.5	9-5

9.2.1 Output Control Register A (OCRA)

Figure 9–1. Output Control Register A (OCRA) — Address 7090h

15	14	13	12	11	10	9	8
OPA15	OPA14	OPA13	OPA12	OPA11	OPA10	OPA9	OPA8
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
OPA7	OPA6	OPA5	OPA4	OPA3	OPA2	OPA1	OPA0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

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

Bits 15–0 OPA15–OPA0

0 = Set corresponding output pin or internal control line LOW.
1 = Set corresponding output pin or internal control line HIGH.

9.2.2 Output Control Register B (OCRB)

Figure 9–2. Output Control Register B (OCRB) — Address 7092h

15	14	13	12	11	10	9	8
OPB15	OPB14	OPB13	OPB12	OPB11	OPB10	OPB9	OPB8
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
OPB7	OPB6	OPB5	OPB4	OPB3	OPB2	OPB1	OPB0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

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

Bits 15–0 OPB15–OPB0

0 = Set corresponding output pin or internal control line LOW.
1 = Set corresponding output pin or internal control line HIGH.

9.2.3 Input Status Register A (ISRA)

Figure 9–3. Input Status Register A (ISRA) — Address 7094h

15	14	13	12	11	10	9	8
IPA15	IPA14	IPA13	IPA12	IPA11	IPA10	IPA9	IPA8
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
IPA7	IPA6	IPA5	IPA4	IPA3	IPA2	IPA1	IPA0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

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

Bits 15–0 IPA15–IPA0

0 = Corresponding input pin or internal signal is LOW.
1 = Corresponding input pin or internal signal is HIGH.

9.2.4 Input Status Register B (ISRB)

Figure 9–4. Input Status Register B (ISRB) — Address 7096h

15	14	13	12	11	10	9	8
IPB15	IPB14	IPB13	IPB12	IPB11	IPB10	IPB9	IPB8
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
IPB7	IPB6	IPB5	IPB4	IPB3	IPB2	IPB1	IPB0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

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

Bits 15–0 IPB15–IPB0

0 = Corresponding input pin or internal signal is LOW.
1 = Corresponding input pin or internal signal is HIGH.

9.2.5 Data and Direction Control Registers

Figure 9–5. Data and Direction Control Registers (PxDATDIR; x = A, B, C, or D)

15	14	13	12	11	10	9	8
x7DIR	x6DIR	x5DIR	x4DIR	x3DIR	x2DIR	x1DIR	x0DIR
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
IOPx7	IOPx6	IOPx5	IOPx4	IOPx3	IOPx2	IOPx1	IOPx0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

Note: R = Read access; W = Write access; –0 = value after reset; x=ports A, B, C, or D; n = 0–7. Refer to Table 9–1 on page 9-3 for address locations of each register.

Bits 15–8 x7DIR–x0DIR

0 = Configure corresponding pin as an INPUT.
1 = Configure corresponding pin as an OUTPUT.

Bits 7–0 IOPx7–IOPx0

If xnDIR = 0, then:

0 = Corresponding I/O pin is read as a LOW.
1 = Corresponding I/O pin is read as a HIGH.

If xnDIR = 1, then:

0 = Set corresponding I/O pin LOW.
1 = Set corresponding I/O pin HIGH.

Chapter 10

PLL Clock Module

This chapter describes the architecture, functions, and programming of the PLL clock module.

Note: 8-Bit Peripheral

This module is interfaced to the 16-bit peripheral bus as an 8-bit peripheral. Therefore, reads from bits 15–8 are undefined; writes to bits 15–8 have no effect.

Topic **Page**

10.1 PLL Clock Module Overview

The PLL clock module provides all the necessary clock signals for C24x devices.

The clock module interfaces to the peripheral bus and provides all of the clocks required for the entire device (see Figure 10–1). There are four sets of clocks, all running at different frequencies:

❑ **CPUCLK** – This is the highest frequency clock provided by the module and is used by the CPU, all memories and any peripherals tied directly to the CPUs buses, including an external memory interface if used. All other clocks are derived by dividing this clock down to a lower frequency.

❑ **SYSCLK** – This clock is a half or a quarter the rate of CPUCLK. It is used to clock all the Prism bus peripherals.

❑ **ACLK** – This clock is used to clock analog modules and has a nominal frequency of 1.0 MHz $\pm$10% if one of the recommended input frequencies is used and the CKINF(3:0) bits are programmed correctly and f_{CPUCLK} is an even number of MHz.

❑ **WDCLK** – This is the low power clock used by the Watchdog Timer/Real Time Interrupt module. It has a nominal frequency of 16 kHz with a 25% duty cycle.

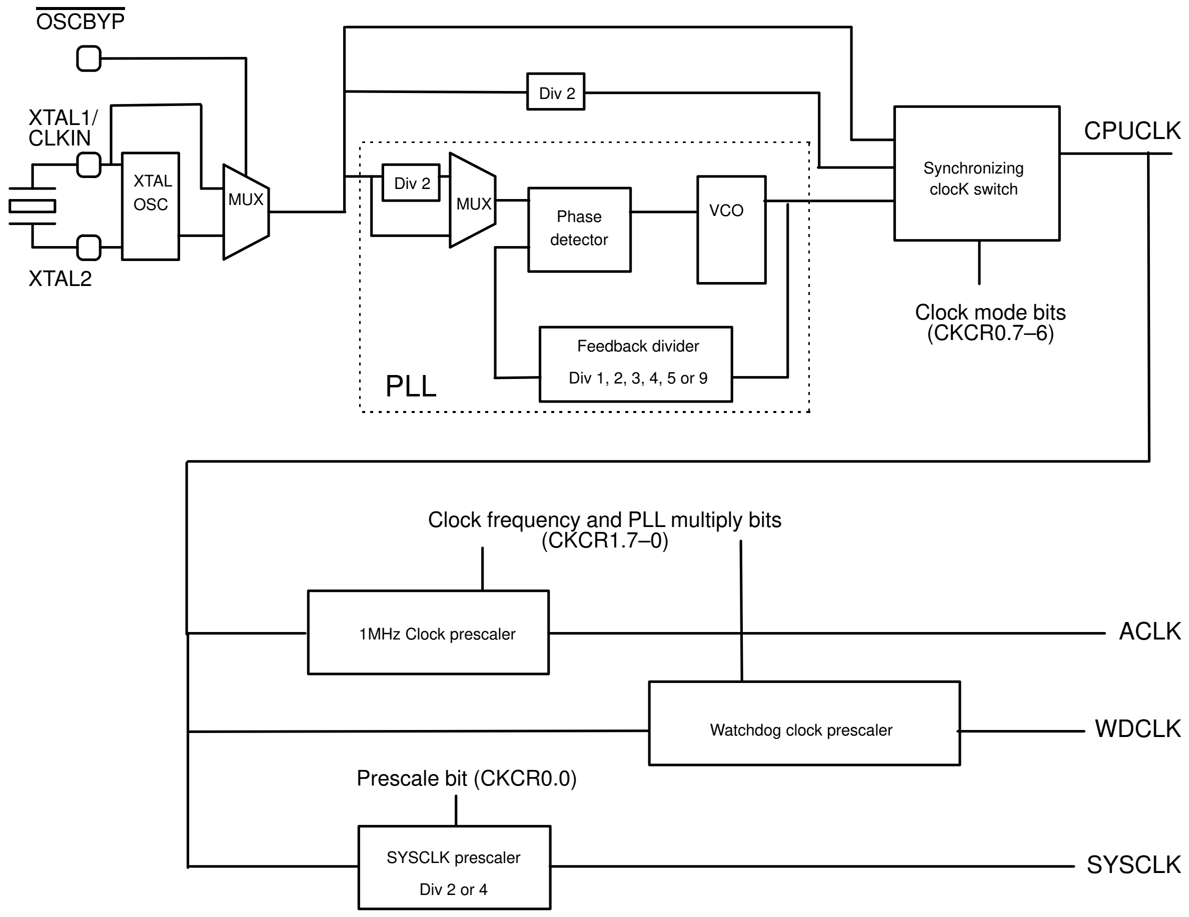
The clock module operates with a 4, 6, or 8 MHz reference crystal in conjunction with it's on chip oscillator circuit, or an external oscillator bypass clock in the range 2–32 MHz. The PLL can multiply the input frequency by factors of 1, 2, 3, 4, 5, and 9. The input clock can also be divided by two before this multiplication to give additional factors of 1.5, 2.5, and 4.5. The actual CPU clock frequency is software selectable from 2 MHz up to the maximum operating frequency of the device. The PLL can also be bypassed with a 1$\times$ or 2$\times$ (CPUCLK frequency) Clock-In input.

Note:

If Clock-In has a higher frequency than 32 MHz, the ACLK and WDCLK frequencies will not be correct.

The clock module contains all necessary control registers. It also contains low power mode control bits which determine which clocks are switched off when the CPU goes into idle mode.

Figure 10–1. PLL Clock Module Block Diagram



Two registers (listed in Table 10–1) control the PLL Clock Module operations:

❑ **CKCR0** (Clock control register 0)

This register contains bits used for general control of the clock module, such as clock mode, low-power mode selection, SYSCLK prescale selection, and status flags.

❑ **CKCR1** (Clock control register 1)

This register specifies the PLL multiplication factor (if enabled) and the frequency of the input clock.

Table 10–1. Addresses of PLL Clock Module Control Registers

			Described in	
Address	**Register**	**Name**	**Subsection**	**Page**
7020h		Reserved†		
7022h		Reserved†		
7024h		Reserved†		
7026h		Reserved†		
7028h		Reserved†		
702Ah‡	CKCR0	Clock Control Register 0	10.3.1	10-15
702Ch‡	CKCR1	Clock Control Register 1	10.3.2	10-17
702Eh		Reserved		

† Reserved for the Watchdog and Real-Time Interrupt Module control registers, see Chapter 6, *Watchdog (WD) and Real-Time Interrupt (RTI) Module*.
‡ Each register also appears at the next, odd address location, that is CKCR0 appears at 702Bh and CKCR1 appears at 702Dh.

10.2 PLL Clock Operation

This section describes the operation and functionality of the PLL clock module. Included are these topics:

- Pin description
- Oscillator operation modes
- PLL operation modes
- CPU clock (CPUCLK) signal frequency selection
- System clock (SYSCLK) signal prescale selection
- Analog module 1 MHz clock (ACLK) signal
- Watchdog counter clock (WDCLK) signal
- PLL Startup
- Low Power Modes

10.2.1 Pin Description

The PLL module has three associated pins:

- $\overline{\text{OSCBYP}}$

  The oscillator bypass ($\overline{\text{OSCBYP}}$) pin is used to select whether the oscillator is bypassed or not. If the device is used with an external clock input (that is, not used with a reference crystal), this signal should be tied to 0V to bypass the crystal reference oscillator circuit.

- XTAL1/CLKIN

  The oscillator in (XTAL1/CLKIN) pin is typically tied to one side of a 4, 6, or 8 MHz reference crystal. This pin may also be used as a clock in pin for an external signal. See subsection 10.2.2, *Oscillator Operation Modes*, for details.

- XTAL2

  The oscillator out (XTAL2) pin is:

  - Tied to the other side of a 4, 6, or 8 MHz reference crystal, or
  - Left open when an external clock is provided via XTAL1/CLKIN.

10.2.2 Oscillator Operation Modes

The Oscillator has two operation modes, Oscillator and Oscillator Bypass (Clock-In) modes (see Table 10–2):

❑ Oscillator Mode

This is the normal operation mode when you use an external reference crystal. This mode is entered when the $\overline{\text{OSCBYP}}$ pin is tied high (V_{IH}) and a 4, 6, or 8 MHz crystal is connected between XTAL1 and XTAL2 to provide a reference crystal frequency. Following device power up it takes about 1 ms for the crystal oscillator circuitry to power up and start generating a good clock.

❑ Clock-In Mode

You can bypass the Oscillator circuitry by tying the $\overline{\text{OSCBYP}}$ pin low (V_{IL}). This allows the device to be clocked by an external signal input on the XTAL1/CLKIN pin. The oscillator circuitry is powered down when bypassed.

Table 10–2. Oscillator Operation Mode Selection

$\overline{\text{OSCBYP}}$ Pin Levels	Oscillator Operation Mode
V_{IH}	Oscillator mode
V_{IL}	Oscillator Bypass (Clock In) mode

10.2.3 PLL Operation Modes

The clock module can operate with the PLL as the clock source or with a divide-by-1 or a divide-by-2 bypass clock.

The CLKMD(1:0) (CKCR0.7–6) bits set the clock source as follows:

CLKMD(1:0)	Mode
00	CLKIN / 2
01	CLKIN
10	PLL
11	PLL

The PLL is only powered up when enabled, CLKMD(1) = 1.

This PLL has a counter to ensure that enough time has elapsed for the PLL to lock at all frequencies before the device is switched over to run from PLL clocks. This lock counter is cleared by a Power-On-Reset. The PLLOCK(1) bit in CKCR0 indicates that the PLL counter has rolled over, the PLL has locked and the device is running on PLL clocks.

PLL multiplication factor can be set to multiply-by-1, 2, 3, 4, 5, and 9. It is controlled by the PLLFB(2:0) bits in CKCR1. Additionally, the clock input to the PLL can be divide-by-2 before use to give additional multiplication factors of 1.5, 2.5, and 4.5. This is controlled by the PLLDIV2 bit in CKCR1.

10.2.4 CPU Clock (CPUCLK) Frequency Selection

The PLL clock module gives you the option of generating one of many possible software-selectable CPU clock (CPUCLK) frequencies for a given crystal or clock in frequency. The selection of the actual CPUCLK frequency is controlled by four bits in the CKCR1 control register:

- ❑ The PLL multiplication ratio select bits, PLLFB(2:0) (CKCR1.2–0). These bits control the PLL multiplication factor.
- ❑ The PLL input divide-by-2 control bit, PLLDIV2 (CKCR1.3). This bit controls whether or not the clock input to the PLL is divided by 2.

The following formulas may be used to calculate the CPU clock frequency given the crystal frequency and the register values:

$$f_{CPUCLK} = f_{CKIN} * \text{(PLL Multiply Ratio)} / 2^{PLLDIV2}$$

where,

$$2\text{ MHz} < f_{CKIN} < 32\text{ MHz}$$

Table 10–3 shows all locked CPU clock frequencies possible, with the 16 different crystal or clock-in frequencies (which will give a true 1 MHz analog clock ACLK), by using different settings of the Feedback bits. Depending on the speed sort of a particular device, some table values will not be applicable. That is, a device characterized to run at or below 20 MHz should not be used with register settings and crystal or clock-in values which yield CPUCLK frequencies above that value. It is not expected that ’C24x devices will be available which run above 40 MHz (the shaded values in Table 10–3).

Table 10–3. Selectable CPU Clock Frequencies in MHz

Crystal or Clock-In Frequency (MHz)	PLL Multiply Ratio * $2^{PLLDIV2}$								
	1	**1.5**	**2**	**2.5**	**3**	**4**	**4.5**	**5**	**9**
2	2	3	4	5	6	8	9	10	18
4	4	6	8	10	12	16	18	20	36
6	6	9	12	15	18	24	27	30	54
8	8	12	16	20	24	32	36	40	72
10	10	15	20	25	30	40	45	50	90
12	12	18	24	30	36	48	54	60	108
14	14	21	28	35	42	56	63	70	126
16	16	24	32	40	48	64	72	80	144
18	18	27	36	45	54	72	81	90	162
20	20	30	40	50	60	80	90	100	180
22	22	33	44	55	66	88	99	110	198
24	24	36	48	60	72	96	108	120	216
26	26	39	52	65	78	104	117	130	234
28	28	42	56	70	84	112	126	140	252
30	30	45	60	75	90	120	135	150	270
32	32	48	64	80	96	128	144	160	288

10.2.5 System Clock (SYSCLK) Frequency Selection

The System Clock (SYSCLK) frequency is generated by dividing the CPUCLK by 2 or by 4. The SYSCLK divider is controlled by the prescale select bit, PLLPS (CKCR0.0). This bit controls two possible prescale options, as described in Table 10–4.

Table 10–4. PLL Prescale Selection Options

PLLPS (CKCR0.0)	SYSCLK Prescale Selection Options
0	CPUCLK divided by 4
1	CPUCLK divided by 2

10.2.6 Analog Module 1 MHz Clock (ACLK)

The clock module provides an analog module clock (ACLK) signal to certain analog peripheral modules. This clock has a nominal frequency of 1 MHz. ACLK is produced by a divider circuit which is controlled by the contents of the PLLFB(2:0), PLLDIV2, CKINF(3:0), CLKMD(1:0), and PLLOCK(1) bits. This circuit corrects the frequency division of the CPUCLK to produce ACLK as close as possible to 1.0 MHz $\pm$10%, independent of the CPUCLK frequency selected, but only as long as one of the recommended Clock-In frequencies is used.

ACLK is synchronously started/stopped with respect to CPUCLK when entering/exiting low-power modes. During an emulator suspend, ACLK continues to run to avoid damage to the analog modules under load.

There is a control register bit, ACLKENA (CKCR0.1), which is used to turn ACLK on and off. This was included for devices which do not need the 1 MHz ACLK, to reduce power consumption and EMI emissions. The ACLKENA bit is cleared to 0 after Power-On Reset which causes the device to power up with the 1 MHz clock turned off.

Note that if the CPUCLK frequency is an odd number of megahertz, ACLK will actually have a frequency of 0.5 MHz.

10.2.7 Watchdog Counter Clock (WDCLK)

The PLL clock module provides a watchdog counter clock (WDCLK) signal to the WD/RTI (if available on the device). The WDCLK is generated by dividing CPUCLK to yield a WDCLK signal of about 16384 Hz. WDCLK is produced by a divider circuit which is controlled by the contents of the PLLFB(2:0), PLLDIV2, CKINF(3:0), CLKMD(1:0), and PLLOCK(1) bits. If the CKINF(3:0) bits are not programmed correctly the WDCLK frequency will be incorrect, similar to ACLK.

Note that WDCLK is only 16 384 Hz (2^{14} Hz) when CLKIN is a power of 2 Hz, see Table 10–5.

Table 10–5. Watchdog Counter Clock Frequencies

CLKIN (Hz)	WDCLK (Hz)
4 000 000	15 625
4 194 304 (2^{22})	16 384
8 000 000	15 625
8 388 608 (2^{23})	16 384

A good way to obtain a higher or lower WDCLK frequency is to put an incorrect value in the CKINF bits. For example, if CLKIN = 4.194304 MHz, but CKINF is set to 1111 (2 MHz), WDCLK will be 32.768 kHz. Note that ACLK will also be affected, so this technique should not be used on devices which use ACLK.

10.2.8 PLL Startup

When the device first powers up the PLL is neither selected nor powered, the device is running off the oscillator (or oscillator bypass) clocks divided-by-2 (CLKMD = 00). The CLKMD(1:0) bits are cleared to 0 by Power-On Reset, as are the PLLFB(2:0) and PLLDIV2 bits.

If PLL clocks are required the PLLFB(2:0) and PLLDIV2 bits should be set to the desired values and the CLKMD(1) bit set to 1. The PLL will be powered up and will start to lock. This will take about 100 µs. The device will continue to run off the divide-by-2 (or 1) clocks until the PLL lock counter has rolled over, indicating that the PLL has reached lock with it's new settings. At this time, the clock module will automatically do a glitch free switch over to the PLL clocks. If the user needs to prevent some code from being executed before the switch to the (higher frequency) PLL clocks has occurred, the PLLOCK(1) bit in CKCR0 can be polled. This bit indicates that the PLL has locked and the device is now running on PLL clocks.

Subsequent changes to the PLLFB and DIV2 bits do not have an immediate effect on the PLL if it is selected as the clock source. If these bits are changed, the changes do not start to take effect until the PLL is deselected by clearing the CLKMD(1) bit. The CLKMD(1) bit should then be immediately set back to 1, the PLL will be powered back up and will start to lock with the new settings. The device will continue to run on the oscillator clocks. When the PLL has relocked, the device will switch back to PLL clocks.

10.2.9 Low-Power Modes

When the IDLE instruction is executed, power is saved by shutting off some or all of the on-chip clocks sources. For the purposes of low power modes, there are three different clock domains which can be shut down independently:

- **CPU Clock Domain.** All clocks in CPU memory except for the interrupt registers.
- **System Clock Domain.** All peripheral clocks (CPUCLK or SYSCLK), the clocks for the CPU's interrupt register, and ACLK.
- **Watch Dog Clock.** The nominally 16 kHz clock used to increment the Watch Dog Timer (WDCLK).

Note:

The terms CPUCLK and CPU Clock Domain, SYSCLK and System Clock Domain are not interchangeable.

Executing the IDLE instruction causes the device to enter one of the four low-power modes, which mode depends on the PLLPM(1:0) (CKCR0.3–2) bits. The selection bits are summarized in Table 10–6.

Table 10–6. Low Power Modes

Low Power Mode	PLLPM(1:0) bits	CPU Clock Domain	System Clock Domain/ ACLK	WDCLK	PLL State†	Osc. State†	Exit Condition	Description
X + not IDLE	XX	On	On	On	On	On	—	Normal Run Mode
0 + IDLE LPM0 (IDLE1)	00	Off	On	On	On	On	Interrupt, reset	Idle1
1 + IDLE LPM1 (IDLE2)	01	Off	Off	On	On	On	Wake-up interrupt, reset	Idle2
2 + IDLE LPM2 (PLL Power Down)	10	Off	Off	On	Off	On	Wake-up interrupt, reset	PLL Power Down
3 + IDLE LPM3 (Oscillator Power Down)	11	Off	Off	Off	Off	Off	Wake-up interrupt, reset	Oscillator Power Down

† If enabled

The low power mode may be exited by a reset or any individually and globally enabled wake-up interrupt. The actual wake-up interrupts available are device-specific, but usually include the Real Time Interrupt (RTI) and the external interrupts (XINTn). See the specific device data sheet to determine the available wake-up interrupts on the device being used.

When exiting LPM2 or LPM3 there will be a delay of up to 100 µs before the clocks start whilst the PLL locks, if the PLL is selected. In addition, when exiting LPM3 with the oscillator connected to a crystal, there will be a delay of about 1ms whilst the oscillator powers up. If the oscillator is bypassed, there is no additional delay.

When entering LPM3, WDCLK is shut down synchronously. There may be a delay while the device waits for WDCLK to enter it's internal master phase before the CPU Clock Domains and System Clock Domains are shut down.

LPM2 stops the clocks to all modules, except that WDCLK is still active. This means that the WD counter will be active in LPM2. Since the CPU is not active, the WD will not be serviced and will effectively bring the device out of LPM2 with a WD reset when a WD overflow occurs. If the RTI is enabled, the RTI will interrupt the CPU with a wake-up interrupt, causing the device to exit standby mode. At this time, the WD could be serviced to prevent the device from being reset.

The LPM3 mode stops all internal clock signals and powers down the PLL and the oscillator. This stops all modules, including the WD/RTI, resulting in the lowest power consumption possible.

Note:

Entering LPM2 makes no sense if the PLL is not enabled.

LPMODE 0 Entry/Exit Sequence.

When entering this mode:

1) The CPU Clock Domain is shut down immediately.
2) All other chip clocks continue running.

When exiting this mode with an interrupt or reset:

1) The CPU Clock Domain starts running again immediately.

LPMODE 1 Entry/Exit Sequence.

When entering this mode:

1) The CPU Clock Domain is shut down immediately.
2) Wait until the System Clock Domain and ACLK are both in a HIGH state and then stop in that state.
3) WDCLK continues to run.

When exiting this mode with a wake-up interrupt or reset:

1) The clock module internal clocks start running immediately.
2) A few cycles later the CPU Clock Domain, the System Clock Domain, and ACLK start running again.

LPMODE 2 Entry/Exit Sequence.

When entering this mode:

1) The CPU Clock Domain is shut down immediately.
2) Wait until the System Clock Domain and ACLK are both in a HIGH state and then stop in that state.
3) WDCLK continues to run.
4) Clocks to switch from PLL to by 1 or by 2 mode.
5) PLL is powered down.

When exiting this mode with a wake-up interrupt or reset:

1) The PLL is powered up and the lock counter begins counting. Clock module internal clocks start running in by 1 or by 2 mode.
2) The CPU Clock Domain, the System Clock Domain, and ACLK start running again.
3) When PLL has locked (lock counter rolled over), the clocks automatically switch back to the PLL.

LPMODE 3 Entry/Exit Sequence.

When entering this mode:

1) The CPU Clock Domain is shut down immediately.
2) Wait until the System Clock Domain and ACLK are both in a HIGH state and then stop in that state.
3) WDCLK continues to run.
4) Clocks to switch from PLL to by 1 or by 2 mode.
5) PLL is powered down.
6) WDCLK keeps running until the internal WDCLK mater phase is LOW.
7) Clock switching logic goes to “all off” state — that is, all internal clocks are stopped.
8) Oscillator (if used) is powered down.

When exiting this mode with a wake-up interrupt or reset:

1) The oscillator is reenabled but the oscillator output is not good for about 1 ms.

2) Clock switching logic switches to by-1 or by-2 state, depending on value of CKMD(0) bit.

3) The PLL is powered up and begins to lock.

4) The CPU Clock Domain, the System Clock Domain, and ACLK start running again.

5) When PLL has locked, clocks automatically switch back to the PLL.

10.3 PLL Clock Control Registers

The PLL clock module is controlled and accessed through control registers. These registers are illustrated and described in the following sections.

The address shown for each register is the typical address used for these modules where the offset is 7020h. A different offset may be used on some devices, see the device data sheet for details.

10.3.1 Clock Control Register 0 (CKCR0)

Figure 10–2. Clock Control Register 0 (CKCR0) — Address 702Bh

7	6	5	4	3	2	1	0
CLKMD(1)	CLKMD(0)	PLLOCK(1)	PLLOCK(0)	PLLPM(1)	PLLPM(0)	ACLKENA	PLLPS
RW–x	RW–x	R–x	R–x	RW–0	RW–0	RW–x	RW–0

Note: R = Read access; W = Write access; –x = not affected by system reset, cleared to 0 by power on reset

Bits 7–6 **CLKMD(1), CLKMD(0)**. Read/Write bits. These bits select the operational mode of the clock module (Table 10–7).

Table 10–7. CLKMD(1:0) Bits vs. Clock Mode

CLKMD(1:0)	Mode
00	CLKIN / 2
01	CLKIN
10	PLL Enabled
11	PLL Enabled

If the device enters a low-power mode which shuts down the PLL, on exiting that low power mode the device will run on CLKIN/2 until the PLL locks if CLKMD = 10, or it will run on CLKIN if CLKMD = 11.

Bits 5–4 **PLLOCK(1), PLLOCK(0)**. Read only bits. These bits indicate when the PLL has entered the mode selected by the CLKMD(1:0) bits. Bit 0 is really only required for device test. Bit 1 can be used to determine if the PLL has locked. This bit can be software polled after the PLL has been enabled to prevent the execution of any time critical code prior to the module switching over to PLL clocks. This bit is unaffected by System Reset and is cleared to 0 by Power-On Reset.

0 = PLL not locked – running off Clock In
1 = PLL locked and running off PLL clocks

Bits 3–2	**PLLPM(1), PLLPM(0)**. Read/Write bits. These bits specify which low power mode will be entered upon execution of an IDLE instruction. These bits are cleared (00b) during Power-On and System Reset, making LPM0 the default. See subsection 10.2.9, *Low-Power Modes*, on page 10-10.
Bit 1	**ACLKENA.** Read/Write bit. Enables the 1 MHz ACLK if set to 1, stops ACLK if cleared to 0. This bit is unaffected by System Reset and is cleared to 0 by Power-On Reset. 0 = ACLK disabled (stopped) 1 = ACLK enabled
Bit 0	**PLLPS.** Read/Write bit. This bit specifies which of two prescale values will be selected for the System clocks. This bit is cleared (0b) during Power-On and System Reset, making CPUCLK/4 the default System clock (SYSCLK) frequency. See subsection 10.2.5, *System Clock (SYSCLK) Frequency Selection*, on page 10-8. 0 = $f_{(SYSCLK)} = f_{(CPUCLK)} / 4$ 1 = $f_{(SYSCLK)} = f_{(CPUCLK)} / 2$

10.3.2 Clock Control Register 1 (CKCR1)

Figure 10–3. Clock Control Register 1 (CKCR1) — Address 702Dh

7	6	5	4	3	2	1	0
CKINF(3)	CKINF(2)	CKINF(1)	CKINF(0)	PLLDIV(2)	PLLFB(2)	PLLFB(1)	PLLFB(0)
RW–x	RW–x	RW–x	RW–x	RW–x	RW–x	RW–x	RW–x

Note: R = Read access; W = Write access; –x = not affected by system reset, cleared to 0 by power on reset

Bits 7–4 **CKINF(3)–CKINF(0)**. Read/Write bits. These bits indicate the crystal or clock-in frequency being used (Table 10–8). This is used by the ACLK divider to ensure that a 1.0 ±10% MHz clock is generated. If ACLK is not used by any module on the device any frequencies can be used (within the range of the oscillator), but, if a 1 MHz clock is required, one of the following crystal frequencies must be used. These bits are unaffected by System Reset and are cleared to 0 by Power-On Reset.

Table 10–8. CKINF(3:0) Bits vs. Clock-In Frequency

CKINF(3:0)	Frequency (MHz)	CKINF(3:0)	Frequency (MHz)
0000	32	1000	16
0001	30	1001	14
0010	28	1010	12
0011	26	1011	10
0100	24	1100	8
0101	22	1101	6
0110	20	1110	4
0111	18	1111	2

Bit 3 **PLLDIV(2)**. Read/Write bit. This bit specifies whether the input to the PLL is divide-by-2. Writing to this bit has no effect on the PLL until the CLKMD(1:0) bits are changed from 1×. This bit is unaffected by System Reset and is cleared to 0 by Power-On-Reset.

0 = Do not divide PLL input
1 = Divide PLL input by 2

Bits 2–0 **PLLFB(2)–PLLFB(0)**. Read/Write bits. These bits specify one of 6 possible PLL multiplication (feedback) ratios (Table 10–9). Writing to this bit has no effect on the PLL until the CLKMD(1:0) bits are changed from 1 ×. These bits are unaffected by System Reset and are cleared to 0 by Power-On-Reset.

Table 10–9. PLLFB(2:0) Bits vs. PLL Multiplication Ratio

PLLFB(2:0)	PLL Multiplication Ratio
000	1
001	2
010	3
011	4
100	5
101	9
110	1
111	1

Section I
Peripheral Descriptions

Section II
Specific TMS320C24x Information

Chapter 11

TMS320C240 DSP Controller

This chapter contains a general description of the ’C240 DSP Controller.

11.1 TMS320C240 DSP Controller Overview

The TMS320C240 and TMS320F240 devices are the first members of a new family of DSP controllers based on the TMS320C2xx generation of 16-bit fixed-point digital signal processors (DSPs). Unless otherwise noted, the term ’x240 refers to both the TMS320C240 and the TMS320F240. Subsection 11.1.1 on page 11-5 provides a comparison of the features of each device. The only difference between these two devices is the type of program memory (see Table 11–1): the ’C240 contains 16K words of ROM and the ’F240 contains 16K words of flash EEPROM. This new family is optimized for digital motor and motion control applications. The DSP controllers combine the enhanced TMS320 architectural design of the ’C2xx core CPU for low-cost, high-performance processing capabilities and several advanced peripherals optimized for motor/motion control applications. These peripherals include the event manager module, which provides general-purpose timers and compare registers to generate up to 12 PWM outputs; and a dual, 10-bit analog-to-digital converter (ADC), which can perform two simultaneous conversions within 10 μs.

Figure 11–1 shows an overview of the ’x240 signals and Figure 11–2 provides a pin out diagram.

Table 11–1. Characteristics of the TMS320x240 DSP Controllers

TMS320x240 Devices	On-chip Memory (Words)				Power Supply (V)	Cycle Time (ns)	Package Type Pin Count
	RAM						
	Data	Data/Program	ROM	Flash EEPROM			
TMS320C240	288	256	16K	0	5	50	PQ 132–P
TMS320F240	288	256	0	16K	5	50	PQ 132–P

Figure 11–1. TMS320C240 and TMS320F240 Device Overview

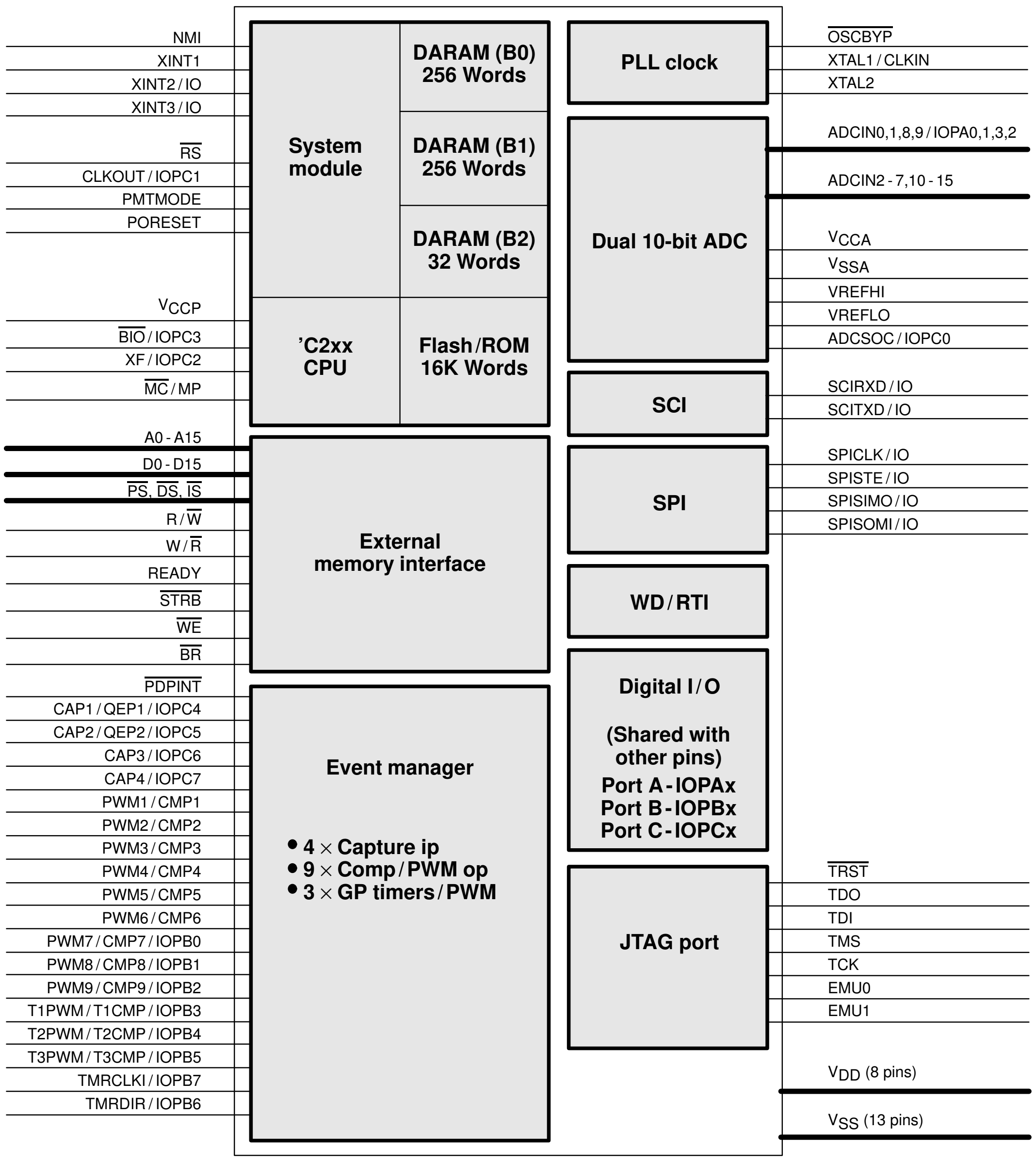
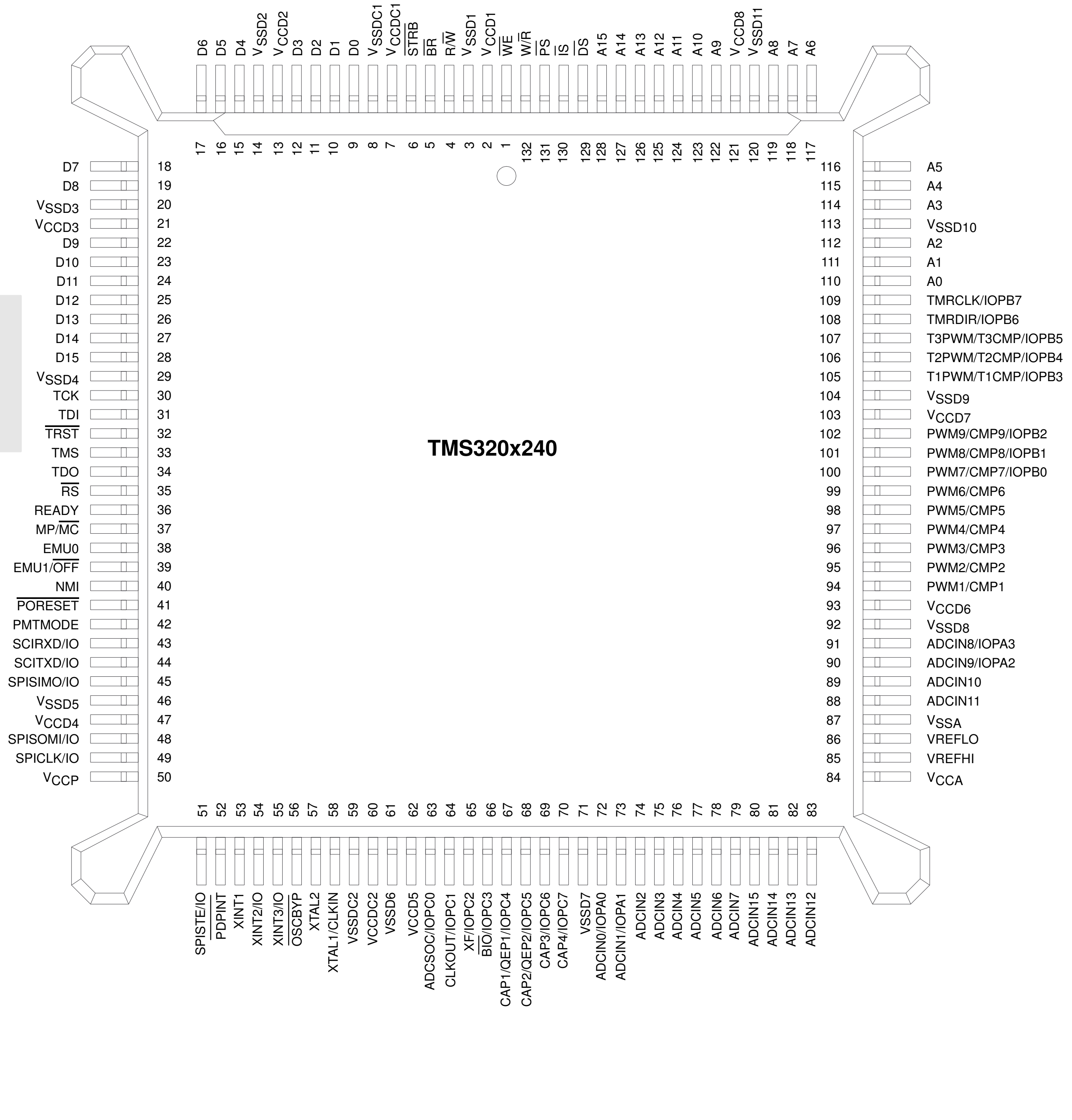


Figure 11–2. TMS320C240 and TMS320F240 Pin Out Assignment



11.1.1 Features of the ’C240

The ’C240 device, shown in Figure 11–1, consists of the following:

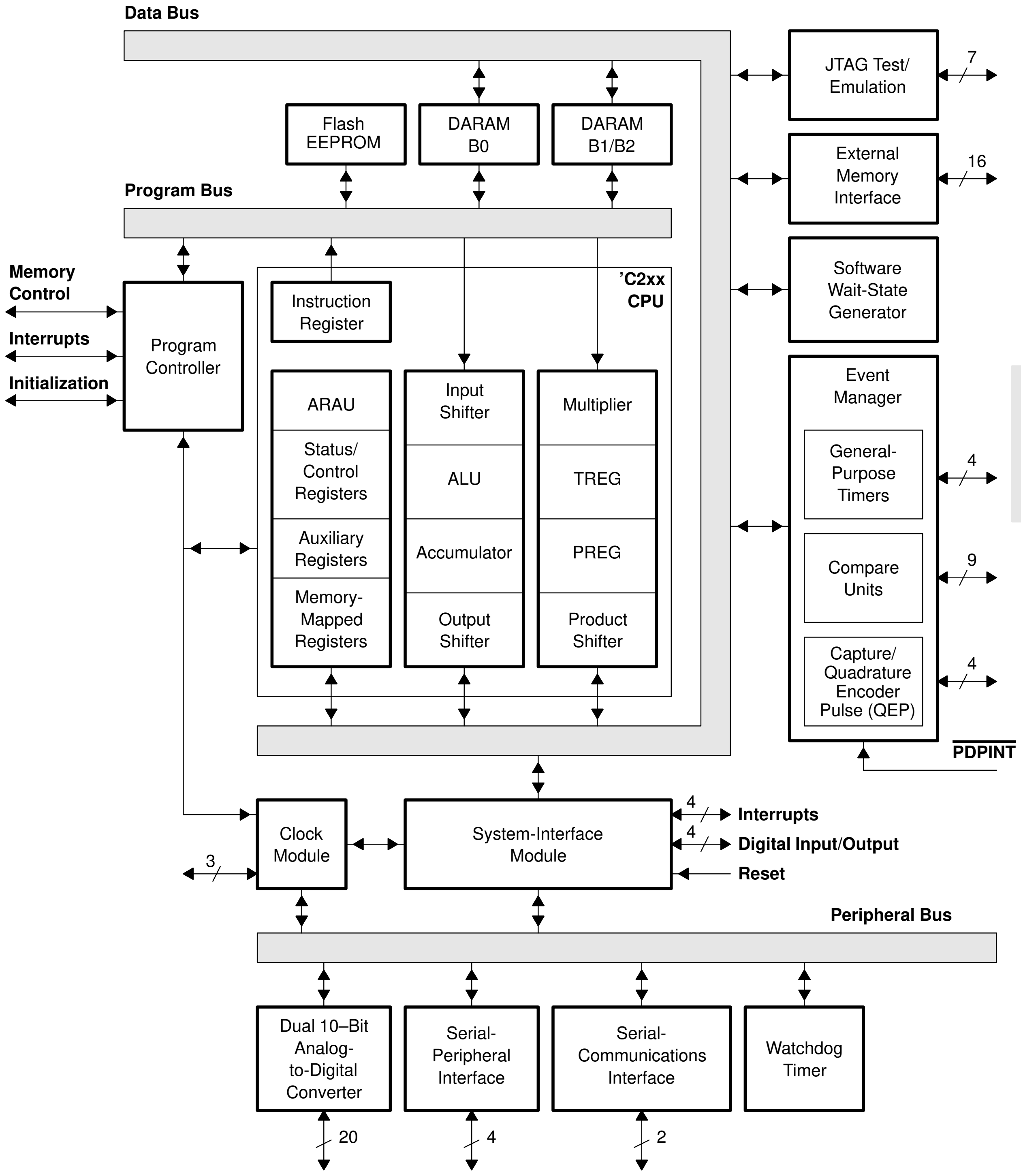
- ❑ High-performance static CMOS technology
- ❑ Includes the ’C2xx core CPU
  - ■ Source code compatible with TMS320C25
  - ■ Upwardly compatible with TMS320C5x
  - ■ 132-pin plastic quad flat package
  - ■ 50-ns instruction cycle time
- ❑ Industrial temperature standard, automotive temperature available
- ❑ Memory
  - ■ 544 words × 16 bits of on-chip data/program dual-access RAM
  - ■ 16K words × 16 bits of on-chip program ROM (’C240)/Flash EEPROM (’F240)
  - ■ 224K words × 16 bits of total memory address reach, (64K data, 64K program and I/O, and 32K global memory space)
- ❑ Event manager module
  - ■ 12 compare/pulse-width modulation (PWM) channels (9 independent)
  - ■ Three 16-bit general-purpose timers with six modes, including continuous up and up/down counting
  - ■ Three 16-bit full compare units with deadband capability
  - ■ Three 16-bit simple compare units
  - ■ Four capture units (two with quadrature encoder-pulse interface capability)
- ❑ Dual 10-bit analog-to-digital converter module
- ❑ 28 individually programmable, multiplexed I/O pins
- ❑ Phase-locked loop (PLL)-based clock module
- ❑ Watchdog timer module with real-time interrupt
- ❑ Serial communication interface (SCI) module
- ❑ Serial peripheral interface (SPI) module
- ❑ Six external interrupts (power drive protect, reset, NMI, and three maskable interrupts)
- ❑ Four power-down modes for low-power operation
- ❑ Scan-based emulation

- Development tools available:
  - TI ANSI C compiler, assembler/linker, and C-source debugger
  - Full range of emulation products: self-emulation (XDS510™), ROM replacement (XDS511™), break-point, trace, and timing (XDS522A™)
  - Evaluation Module (EVM) with JTAG emulation
  - Third-party digital motor control and fuzzy-logic development support

11.1.2 Architectural Overview

The functional block diagram (Figure 11–3) provides a high level description of each component in the 'x240 DSP controller device. The 'x240 devices are composed of three main functional units: a 'C2xx DSP core, internal memory, and peripherals. In addition to these three functional units, there are several system-level features of the 'x240 that are distributed. These system features include the memory map, device reset, interrupts, digital input/output (I/O), clock generation, and low-power operation.

Figure 11–3. TMS320x240 Functional Block Diagram



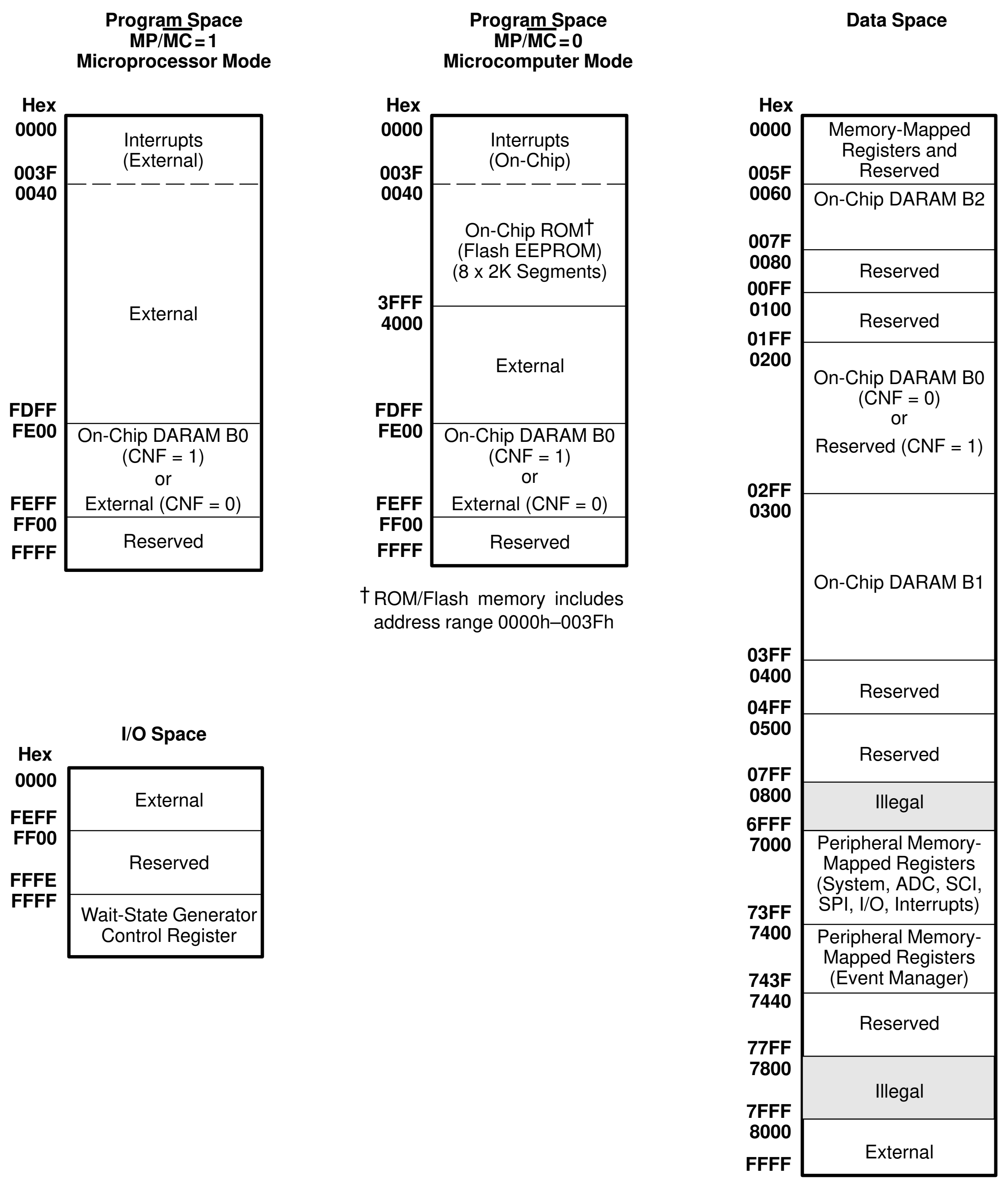
11.2 Memory Map

The TMS320x240 implements three separate address spaces for program memory, data memory, and I/O. Each space accommodates a total of 64K 16-bit words. Within the 64K words of data space, the 256 to 32K words at the top of the address range can be defined to be external global memory in increments of powers of two, as specified by the contents of the global memory allocation register (GREG). Access to global memory is arbitrated using the global memory bus request ($\overline{\text{BR}}$) signal.

On the ’x240, the first 96 data memory locations (0−5Fh) are either allocated for memory-mapped registers or reserved. This memory-mapped register space contains various control and status registers including those for the CPU.

All the on-chip peripherals of ’x240 device are mapped into data memory space. Access to these registers is made by the CPU instructions addressing their data memory locations. Figure 11–4 shows the memory map.

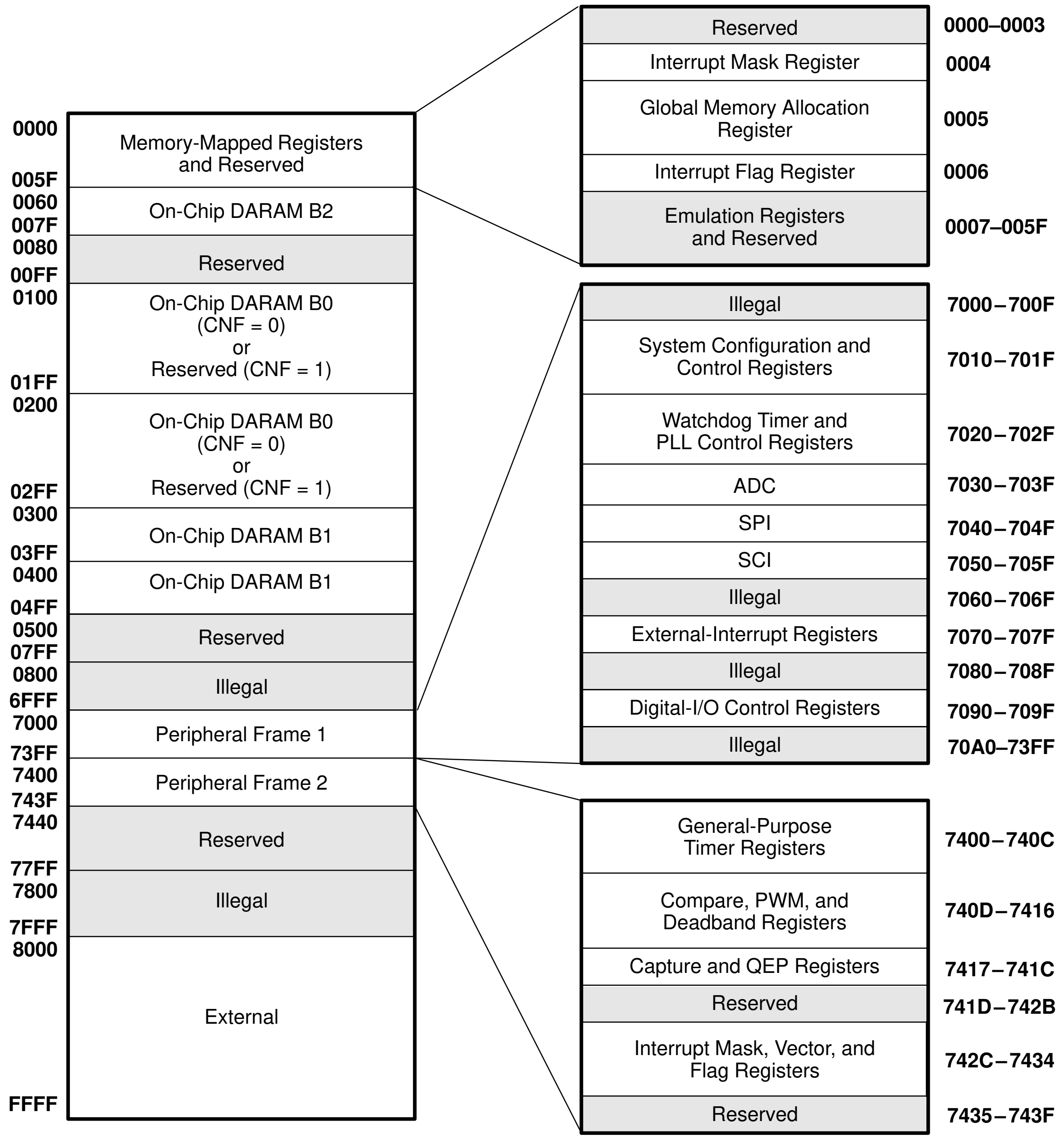
Figure 11–4. TMS320x240 Memory Map


Program Space MP/MC = 1 Microprocessor Mode
Hex
0000 Interrupts (External)
003F
0040 External
FDFF
FE00 On-Chip DARAM B0 (CNF = 1) or External (CNF = 0)
FEFF
FF00 Reserved
FFFF
Program Space MP/MC = 0 Microcomputer Mode
Hex
0000 Interrupts (On-Chip)
003F
0040 On-Chip ROM† (Flash EEPROM) (8 x 2K Segments)
3FFF
4000 External
FDFF
FE00 On-Chip DARAM B0 (CNF = 1) or External (CNF = 0)
FEFF
FF00 Reserved
FFFF
† ROM/Flash memory includes address range 0000h–003Fh
Data Space
Hex
0000 Memory-Mapped Registers and Reserved
005F
0060 On-Chip DARAM B2
007F
0080 Reserved
00FF
0100 Reserved
01FF
0200 On-Chip DARAM B0 (CNF = 0) or Reserved (CNF = 1)
02FF
0300 On-Chip DARAM B1
03FF
0400 Reserved
04FF
0500 Reserved
07FF
0800 Illegal
6FFF
7000 Peripheral Memory-Mapped Registers (System, ADC, SCI, SPI, I/O, Interrupts)
73FF
7400 Peripheral Memory-Mapped Registers (Event Manager)
743F
7440 Reserved
77FF
7800 Illegal
7FFF
8000 External
FFFF
I/O Space
Hex
0000 External
FEFF
FF00 Reserved
FFFE
FFFF Wait-State Generator Control Register

11.3 Peripheral Memory Map

The TMS320x240 system and peripheral control register frame contains all the data, status, and control bits to operate the system and peripheral modules on the device (excluding the event manager). Figure 11–5 shows the peripheral memory map.

Figure 11–5. TMS320x240 Peripheral Memory Map


0000
005F
Memory-Mapped Registers and Reserved
0060
007F
On-Chip DARAM B2
0080
00FF
Reserved
0100
01FF
On-Chip DARAM B0 (CNF = 0) or Reserved (CNF = 1)
0200
02FF
On-Chip DARAM B0 (CNF = 0) or Reserved (CNF = 1)
0300
03FF
On-Chip DARAM B1
0400
04FF
On-Chip DARAM B1
0500
07FF
Reserved
0800
6FFF
Illegal
7000
73FF
Peripheral Frame 1
7400
743F
Peripheral Frame 2
7440
77FF
Reserved
7800
7FFF
Illegal
8000
FFFF
External
Reserved 0000–0003
Interrupt Mask Register 0004
Global Memory Allocation Register 0005
Interrupt Flag Register 0006
Emulation Registers and Reserved 0007–005F
Illegal 7000–700F
System Configuration and Control Registers 7010–701F
Watchdog Timer and PLL Control Registers 7020–702F
ADC 7030–703F
SPI 7040–704F
SCI 7050–705F
Illegal 7060–706F
External-Interrupt Registers 7070–707F
Illegal 7080–708F
Digital-I/O Control Registers 7090–709F
Illegal 70A0–73FF
General-Purpose Timer Registers 7400–740C
Compare, PWM, and Deadband Registers 740D–7416
Capture and QEP Registers 7417–741C
Reserved 741D–742B
Interrupt Mask, Vector, and Flag Registers 742C–7434
Reserved 7435–743F

11.4 Digital I/O and Shared Pin Functions

The 'C240 has a total of 28 pins shared between Primary functions and I/Os. These pins are divided into two groups:

- **Group1** – Primary functions shared with I/Os belonging to dedicated I/O ports, Port A, Port B, and Port C.
- **Group2** – Primary functions belonging to Peripheral Modules which also have an in-built I/O feature as a secondary function, for example SCI, SPI, external interrupts, and PLL clock module.

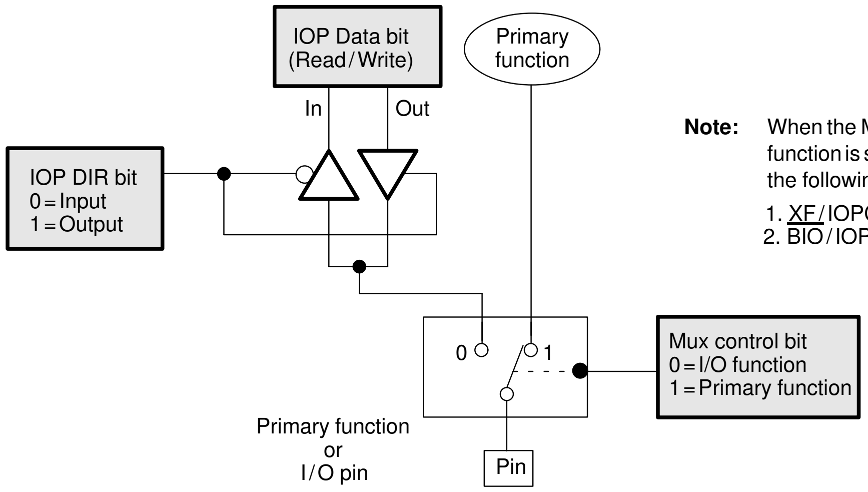
11.4.1 Description of Group1 Shared I/O pins

The control structure for Group1 type shared I/O pins is shown in Figure 11–6. The only exception to this configuration is the CLKOUT/IOPC1 pin, which is described later in this section. In Figure 11–6, each pin has three bits which define its operation:

- Mux control bit – this bit selects between the primary function (1) and I/O function (0) of the pin.
- I/O direction bit – if the I/O function is selected for the pin (mux control bit is set to 0), this bit determines whether the pin is an input (0) or output (1).
- I/O data bit – if the I/O function is selected for the pin (mux control bit is set to 0) and the direction selected is an input, data is read from this bit; if the direction selected is an output, data is written to this bit.

The mux control bit, I/O direction bit, and I/O data bit are in the I/O control registers described in subsection 11.4.3 on page 11-13.

Figure 11–6. Shared Pin Configuration



Note: When the Mux control bit = 1, the primary function is selected in all cases except for the following pins:

1. $\overline{XF}$/IOPC2 (0 = Primary function)
2. $\overline{BIO}$/IOPC3 (0 = Primary function)

A summary of Group1 pin configurations and associated bits is shown in Table 11–2.

Table 11–2. TMS320C240 Shared Pin Configuration

Pin #	Mux Control Register (name.bit #)	Pin function selected (CRx.n = 1)	Pin function selected (CRx.n = 0)	IO Port Data & Direction† Register	Data bit #	Dir bit #
72	CR**A**.0	ADCIN0	IOPA0	P**A**DATDIR	0	8
73	CR**A**.1	ADCIN1	IOPA1	P**A**DATDIR	1	9
91	CR**A**.2	ADCIN9	IOPA2	P**A**DATDIR	2	10
90	CR**A**.3	ADCIN8	IOPA3	P**A**DATDIR	3	11
100	CR**A**.8	PWM7/CMP7	IOPB0	P**B**DATDIR	0	8
101	CR**A**.9	PWM8/CMP8	IOPB1	P**B**DATDIR	1	9
102	CR**A**.10	PWM9/CMP9	IOPB2	P**B**DATDIR	2	10
105	CR**A**.11	T1PWM/T1CMP	IOPB3	P**B**DATDIR	3	11
106	CR**A**.12	T2PWM/T2CMP	IOPB4	P**B**DATDIR	4	12
107	CR**A**.13	T3PWM/T3CMP	IOPB5	P**B**DATDIR	5	13
108	CR**A**.14	TMRDIR	IOPB6	P**B**DATDIR	6	14
109	CR**A**.15	TMRCLK	IOPB7	P**B**DATDIR	7	15
63	CR**B**.0	ADCSOC	IOPC0	P**C**DATDIR	0	8
64	SCR.7–6‡					
	0 0	IOPC1		P**C**DATDIR	1	9
	0 1	CLKOUT (Watchdog clock)		—	—	—
	1 0	CLKOUT (SYSCLK)		—	—	—
	1 1	CLKOUT (CPUCLK)		—	—	—
65	CR**B**.2	IOPC2	XF	P**C**DATDIR	2	10
66	CR**B**.3	IOPC3	$\overline{\text{BIO}}$	P**C**DATDIR	3	11
67	CR**B**.4	CAP1/QEP1	IOPC4	P**C**DATDIR	4	12
68	CR**B**.5	CAP2/QEP2	IOPC5	P**C**DATDIR	5	13
69	CR**B**.6	CAP3	IOPC6	P**C**DATDIR	6	14
70	CR**B**.7	CAP4	IOPC7	P**C**DATDIR	7	15

† Valid only if the I/O function is selected on the pin.
‡ SCR.7–6 is bits 7 and 6 in the system control register (see System Functions chapter in *TMS320C24x DSP Controllers Reference Set, Volume 1: CPU, System, and Instruction Set*).

11.4.2 Description of Group2 Shared I/O Pins

Group2 shared pins belong to peripherals which have in-built general purpose I/O capability. Control and configuration for these pins is achieved by setting appropriate bits within the control and configuration registers of the peripherals. Table 11–3 lists the Group2 shared pins.

Table 11–3. Group2 Shared Pins

Pin #	Primary Function	Peripheral Module
43	SCIRXD	SCI
44	SCITXD	SCI
45	SPISIMO	SPI
48	SPISOMI	SPI
49	SPICLK	SPI
51	SPISTE	SPI
54	XINT2	External Interrupts
55	XINT3	External Interrupts

For information on:

- ❑ Serial Communications Interface (SCI) – see Chapter 4
- ❑ Serial Peripheral Interface (SPI) – see Chapter 5
- ❑ External Interrupts – see subsection 11.5.3

11.4.3 Digital I/O Control Registers

Table 11–4 lists the registers available to the digital I/O module. As with other 'C24x peripherals, the registers are memory mapped to the data space.

Table 11–4. Addresses of Digital I/O Control Registers

Address	Register	Name	Page
7090h	OCRA	I/O mux control register A	11-14
7092h	OCRB	I/O mux control register B	11-15
7098h	PADATDIR	I/O port A data and direction register	11-16
709Ah	PBDATDIR	I/O port B data and direction register	11-17
709Ch	PCDATDIR	I/O port C data and direction register	11-18

I/O Mux Control Registers

Figure 11–7. I/O Mux Control Register A (OCRA) — Address 7090h

15	14	13	12	11	10	9	8
CRA.15	CRA.14	CRA.13	CRA.12	CRA.11	CRA.10	CRA.9	CRA.8
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
Reserved				CRA.3	CRA.2	CRA.1	CRA.0
RW–0				RW–0	RW–0	RW–0	RW–0

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

Table 11–5. I/O Mux Control Register A (OCRA) Configuration

		Pin function selected	
Bit #	**Name.bit #**	**(CRA.n = 1)**	**(CRA.n = 0)**
0	CR**A**.0	ADCIN0	IOPA0
1	CR**A**.1	ADCIN1	IOPA1
2	CR**A**.2	ADCIN9	IOPA2
3	CR**A**.3	ADCIN8	IOPA3
8	CR**A**.8	PWM7/CMP7	IOPB0
9	CR**A**.9	PWM8/CMP8	IOPB1
10	CR**A**.10	PWM9/CMP9	IOPB2
11	CR**A**.11	T1PWM/T1CMP	IOPB3
12	CR**A**.12	T2PWM/T2CMP	IOPB4
13	CR**A**.13	T3PWM/T3CMP	IOPB5
14	CR**A**.14	TMRDIR	IOPB6
15	CR**A**.15	TMRCLK	IOPB7

Figure 11–8. I/O Mux Control Register B (OCRB) — Address 7092h

15	14	13	12	11	10	9	8
Reserved							
RW–0							

7	6	5	4	3	2	1	0
CRB.7	CRB.6	CRB.5	CRB.4	CRB.3	CRB.2	CRB.1	CRB.0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

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

Table 11–6. I/O Mux Control Register B (OCRB) Configuration

		Pin function selected	
Bit #	**Name.bit #**	**(CRB.n = 1)**	**(CRB.n = 0)**
0	CR**B**.0	ADCSOC	IOPC0
2	CR**B**.2	IOPC2	XF
3	CR**B**.3	IOPC3	$\overline{\text{BIO}}$
4	CR**B**.4	CAP1/QEP1	IOPC4
5	CR**B**.5	CAP2/QEP2	IOPC5
6	CR**B**.6	CAP3	IOPC6
7	CR**B**.7	CAP4	IOPC7

I/O Port Data and Direction Registers

Figure 11–9. I/O Port A Data and Direction Register (PADATDIR) — Address 7098h

15	14	13	12	11	10	9	8
Reserved				A3DIR	A2DIR	A1DIR	A0DIR
				RW–0	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
Reserved				IOPA3	IOPA2	IOPA1	IOPA0
				RW–0	RW–0	RW–0	RW–0

Note: R = Read access, W = Write access, –0 = value after reset, n = 0–3

Bits 15–12 **Reserved**

Bits 11–8 **A3DIR–A0DIR**. Port A direction control bits.

0 = Configure corresponding pin as an INPUT.
1 = Configure corresponding pin as an OUTPUT.

Bits 7–4 **Reserved**

Bits 3–0 **IOPA3–IOPA0**. Port A data bits.

If AnDIR = 0, then:

0 = Corresponding I/O pin is read as a LOW.
1 = Corresponding I/O pin is read as a HIGH.

If AnDIR = 1, then:

0 = Set corresponding I/O pin to an output LOW level.
1 = Set corresponding I/O pin to an output HIGH level.

Figure 11–10. I/O Port B Data and Direction Register (PBDATDIR) — Address 709Ah

15	14	13	12	11	10	9	8
B7DIR	B6DIR	B5DIR	B4DIR	B3DIR	B2DIR	B1DIR	B0DIR
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
IOPB7	IOPB6	IOPB5	IOPB4	IOPB3	IOPB2	IOPB1	IOPB0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

Note: R = Read access, W = Write access, –0 = value after reset, n = 0–7

Bits 15–8 **B7DIR–B0DIR**. Port B direction control bits.

0 = Configure corresponding pin as an INPUT.
1 = Configure corresponding pin as an OUTPUT.

Bits 7–0 **IOPB7–IOPB0**. Port B data bits.

If BnDIR = 0, then:

0 = Corresponding I/O pin is read as a LOW.
1 = Corresponding I/O pin is read as a HIGH.

If BnDIR = 1, then:

0 = Set corresponding I/O pin to an output LOW level.
1 = Set corresponding I/O pin to an output HIGH level.

Figure 11–11.I/O Port C Data and Direction Register (PCDATDIR) — Address 709Ch

15	14	13	12	11	10	9	8
C7DIR	C6DIR	C5DIR	C4DIR	C3DIR	C2DIR	C1DIR	C0DIR
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
IOPC7	IOPC6	IOPC5	IOPC4	IOPC3	IOPC2	IOPC1	IOPC0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

Note: R = Read access, W = Write access, –0 = value after reset, n = 0–7

Bits 15–8 **C7DIR–C0DIR**. Port C direction control bits.

0 = Configure corresponding pin as an INPUT.
1 = Configure corresponding pin as an OUTPUT.

Bits 7–0 **IOPC7–IOPC0**. Port C data bits.

If CnDIR = 0, then:

0 = Corresponding I/O pin is read as a LOW.
1 = Corresponding I/O pin is read as a HIGH.

If CnDIR = 1, then:

0 = Set corresponding I/O pin to an output LOW level.
1 = Set corresponding I/O pin to an output HIGH level.

11.5 Device Reset and Interrupts

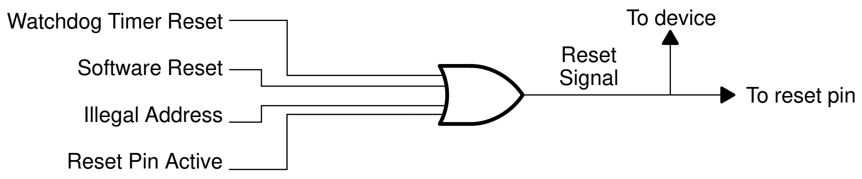
The ’C240 software-programmable interrupt structure supports flexible on-chip and external interrupt configurations to meet real-time interrupt-driven application requirements. The ’x240 recognizes four types of interrupt sources:

- **Reset** (hardware- or software-initiated) is unarbitrated by the CPU and takes immediate priority over any other executing functions. All maskable interrupts are disabled until the reset service routine enables them.
- **Hardware-generated interrupts** are requested by external pins or by on-chip peripherals. There are two types:
  - *External interrupts* are generated by one of five external pins corresponding to the interrupts XINT1, XINT2, XINT3, PDPINT, and NMI. The first four can be masked both by dedicated enable bits and by the CPU’s interrupt mask register (IMR) register, which can mask each maskable interrupt line at the DSP core. NMI, which is not maskable, takes priority over peripheral interrupts and software-generated interrupts. It can be locked out only by an already executing NMI or a reset.
  - *Peripheral interrupts* are initiated internally by these on-chip peripheral modules: the event manager, SPI, SCI, WD/RTI, and ADC. They can be masked both by enable bits for each event in each peripheral and by the CPU’s interrupt mask register (IMR) register, which can mask each maskable interrupt line at the DSP core.
- **Software-generated interrupts** for the ’x240 device include:
  - *INTR instruction.* This instruction allows initialization of any ’x240 interrupt with software. Its operand indicates to which interrupt vector location the CPU branches. This instruction globally disables maskable interrupts (sets the INTM bit to 1).
  - *NMI instruction.* This instruction forces a branch to interrupt vector location 24h, the same location used for the nonmaskable hardware interrupt NMI. NMI can be initiated by driving the NMI pin low or by executing an NMI instruction. This instruction globally disables maskable interrupts.
  - *TRAP instruction.* This instruction forces the CPU to branch to interrupt vector location 22h. The TRAP instruction does not disable maskable interrupts (INTM is not set to 1); thus when the CPU branches to the interrupt service routine, that routine can be interrupted by the maskable hardware interrupts.
  - *An emulator trap.* This interrupt can be generated with either an INTR instruction or a TRAP instruction.

11.5.1 Reset

The reset operation ensures an orderly startup sequence for the device. There are four possible causes of a reset, as shown in Figure 11–12. Three of these causes are internally generated; the other cause, the RS pin, is controlled externally.

Figure 11–12. Reset Signals



The four possible reset signals are generated as follows:

- ❑ **Watchdog timer reset.** A watchdog timer generated reset occurs if the watchdog timer overflows or an improper value is written to either the watchdog key register or the watchdog control register. (Note that when the device is powered on, the watchdog timer is automatically active.)
- ❑ **Software-generated reset.** This is implemented with the system control register (SCR). Clearing the RESET0 bit (bit14) or setting the RESET1 bit (bit15) causes a system reset.
- ❑ **Illegal Address.** The system and peripheral module control register frame address map contains unimplemented address locations in the ranges labeled reserved. Any access to an address located in the Reserved ranges will generate an illegal-address reset.
- ❑ **Reset pin active.** To generate an external reset pulse on the $\overline{\text{RS}}$ pin, a low-level pulse duration of as little as a few nanoseconds is usually effective; however, pulses of one SYSCLK cycle are necessary to ensure that the device recognizes the reset signal. A typical reset circuit required for the ’x240 device consists of a 10-kilohm pullup resistor from the $\overline{\text{RS}}$ pin to V_{CC}.

Once a reset source is activated, the external $\overline{\text{RS}}$ pin is driven (active) low for a minimum of eight SYSCLK cycles. This allows the ’x240 to reset external devices connected to the $\overline{\text{RS}}$ pin. (The $\overline{\text{RS}}$ pin is an open-collector I/O pin and must have a pullup resistor attached.) Additionally, if the $\overline{\text{RS}}$ pin is held low, the reset logic holds the device in a reset state for as long as the $\overline{\text{RS}}$ pin is held low.

When a reset signal is received, the program determines the source of the reset by reading the contents of the system status register (SSR). The SSR contains one status bit for each of the four internal sources that can cause a reset. During a reset, RAM contents remain unchanged, and all control bits that are affected by a reset are initialized.

11.5.2 Hardware-Generated Interrupts

All the hardware interrupt lines of the DSP core are given a priority rank from 1 to 10 (1 being highest). When more than one of these hardware interrupts is pending acknowledgment, the interrupt of highest rank gets acknowledged first. The others are acknowledged in order after that. Of those ten lines, six are for maskable interrupt lines (INT1–INT6) and one is for the nonmaskable interrupt (NMI) line. INT1–INT6 and NMI have the priorities shown in Table 11–7.

Table 11–7. Maskable Interrupt Priorities at the Level of the DSP Core

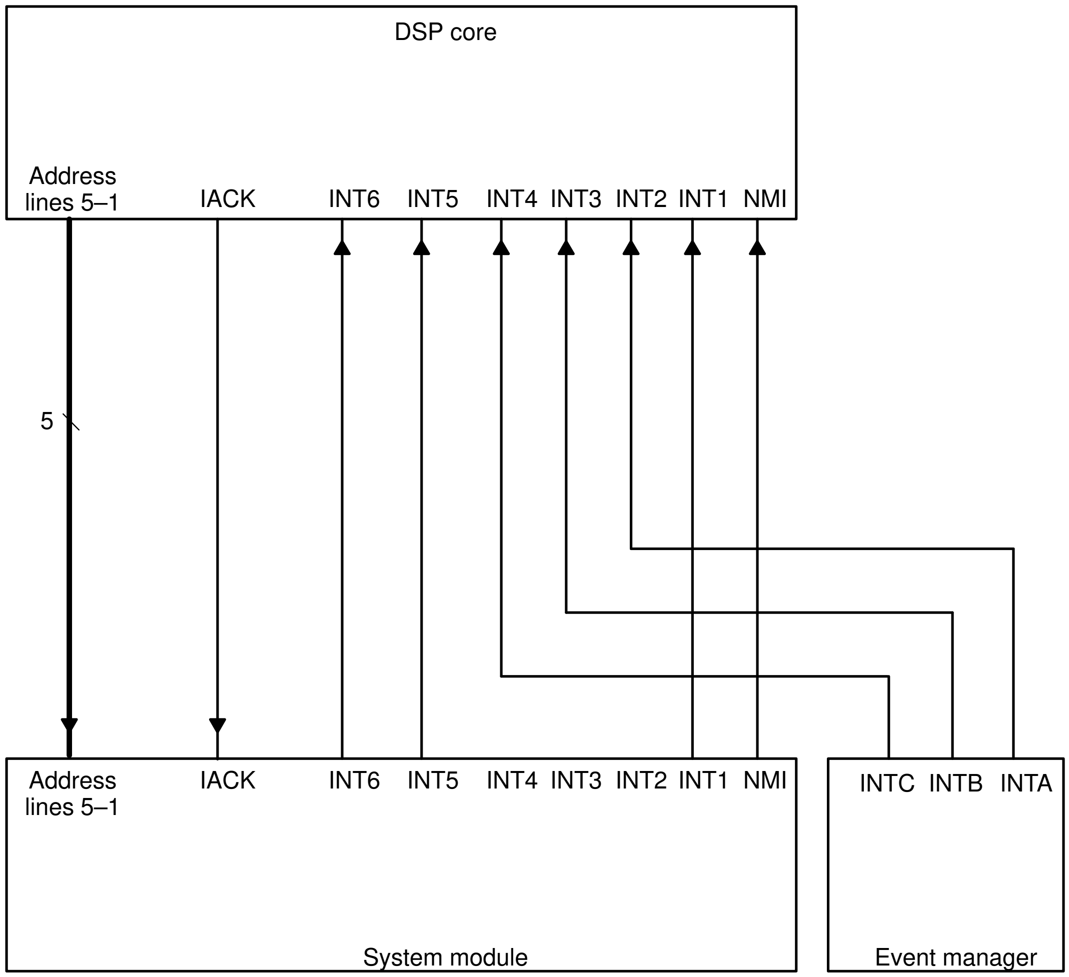
Priority at the DSP Core	Maskable Interrupt
3	NMI
4	INT1
5	INT2
6	INT3
7	INT4
8	INT5
9	INT6

The inputs to these lines are controlled by the system module and the event manager as summarized in Table 11–8 and shown in Figure 11–13.

Table 11–8. Interrupt Lines Controlled by the System Module and Event Manager

Interrupt Line	Controlled by
INT1 INT5 INT6 NMI	System Module
INT2 INT3 INT4	Event Manager

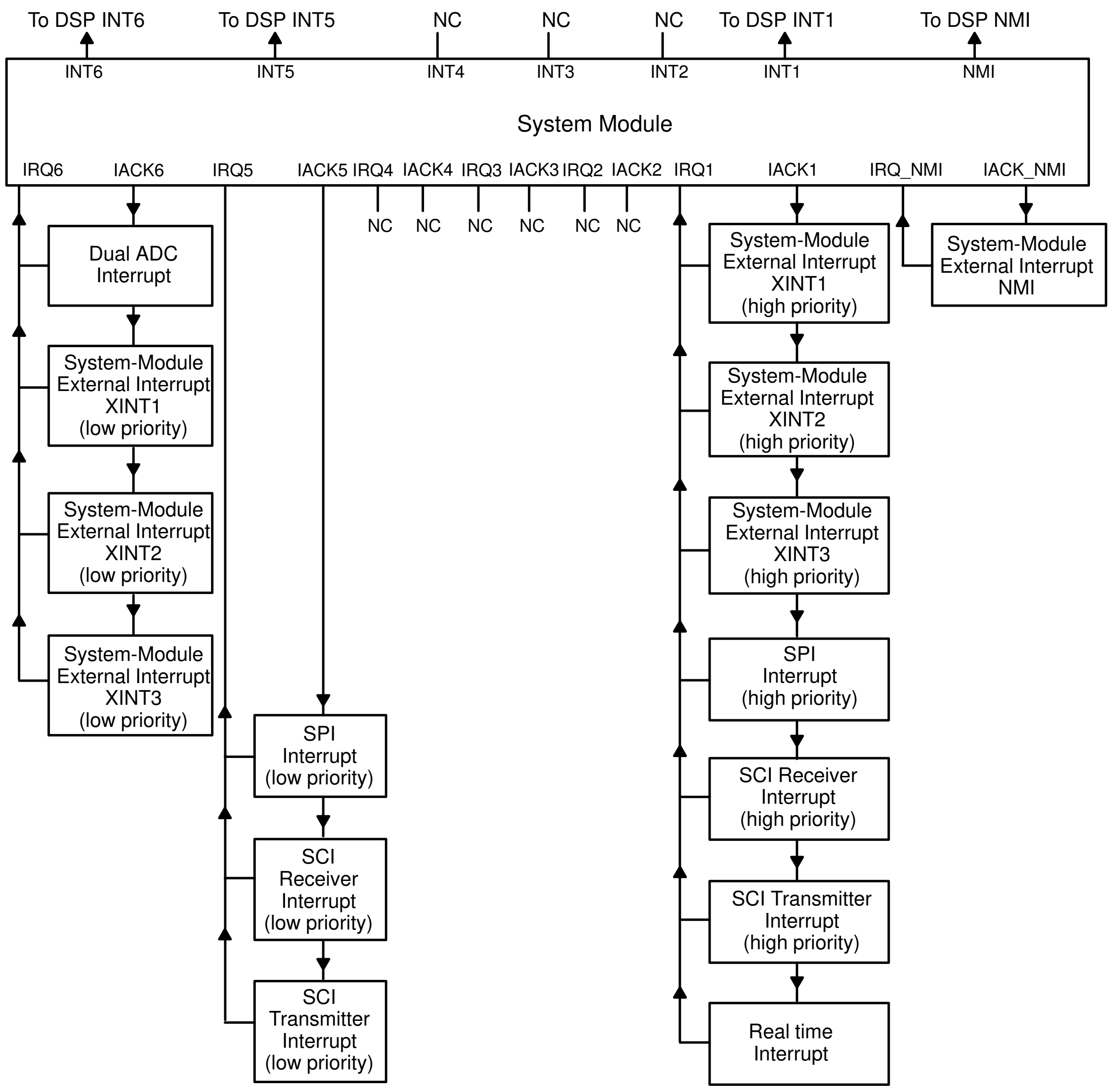
Figure 11–13. DSP Interrupt Structure



At the level of the system module and the event manager, each of the maskable interrupt lines (INT1–INT6) is connected to multiple maskable interrupt sources. Sources connected to interrupt line INT1 are called Level 1 interrupts; sources connected to interrupt line INT2 are called Level 2 interrupts; and so on. For each interrupt line, the multiple sources also have a set priority ranking. The source with highest priority has its interrupt request responded to by the DSP core first.

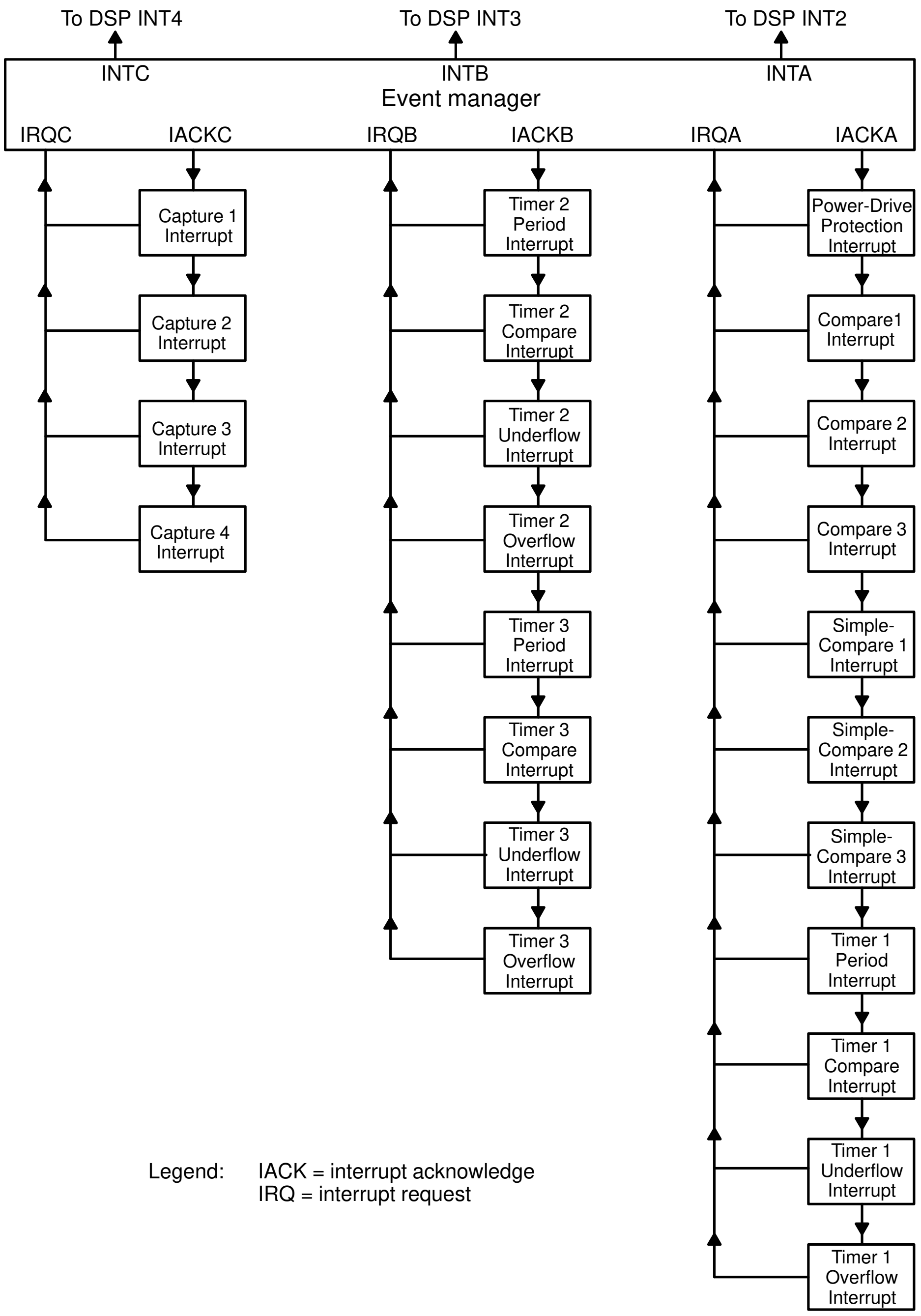
Figure 11–14 shows the sources and priority ranking for the interrupts controlled by the system module. Note that for each interrupt chain, the interrupt source of highest priority is at the top. Priority decreases from the top of the chain to the bottom. Figure 11–15 shows the interrupt sources and priority ranking for the event manager interrupts.

Figure 11–14. System–Module Interrupt Structure


To DSP INT6
To DSP INT5
NC
NC
NC
To DSP INT1
To DSP NMI
INT6
INT5
INT4
INT3
INT2
INT1
NMI
System Module
IRQ6
IACK6
IRQ5
IACK5
IRQ4
IACK4
IRQ3
IACK3
IRQ2
IACK2
IRQ1
IACK1
IRQ_NMI
IACK_NMI
NC
NC
NC
NC
NC
NC
Dual ADC Interrupt
System-Module External Interrupt XINT1 (low priority)
System-Module External Interrupt XINT2 (low priority)
System-Module External Interrupt XINT3 (low priority)
SPI Interrupt (low priority)
SCI Receiver Interrupt (low priority)
SCI Transmitter Interrupt (low priority)
System-Module External Interrupt XINT1 (high priority)
System-Module External Interrupt XINT2 (high priority)
System-Module External Interrupt XINT3 (high priority)
SPI Interrupt (high priority)
SCI Receiver Interrupt (high priority)
SCI Transmitter Interrupt (high priority)
Real time Interrupt
System-Module External Interrupt NMI

Legend:	
	NC = no connection
	IACK = interrupt acknowledge
	IRQ = interrupt request

Figure 11–15. Event Manager Interrupt Structure


To DSP INT4
To DSP INT3
To DSP INT2
INTC
INTB
INTA
Event manager
IRQC
IACKC
IRQB
IACKB
IRQA
IACKA
Capture 1 Interrupt
Capture 2 Interrupt
Capture 3 Interrupt
Capture 4 Interrupt
Timer 2 Period Interrupt
Timer 2 Compare Interrupt
Timer 2 Underflow Interrupt
Timer 2 Overflow Interrupt
Timer 3 Period Interrupt
Timer 3 Compare Interrupt
Timer 3 Underflow Interrupt
Timer 3 Overflow Interrupt
Power-Drive Protection Interrupt
Compare1 Interrupt
Compare 2 Interrupt
Compare 3 Interrupt
Simple-Compare 1 Interrupt
Simple-Compare 2 Interrupt
Simple-Compare 3 Interrupt
Timer 1 Period Interrupt
Timer 1 Compare Interrupt
Timer 1 Underflow Interrupt
Timer 1 Overflow Interrupt

Legend:	IACK = interrupt acknowledge
	IRQ = interrupt request

Each of the interrupt sources has its own control register with a flag bit and an enable bit. When an interrupt request is received, the flag bit in the corresponding control register is set. If the enable bit is also set, a signal is sent to arbitration logic, which may simultaneously receive similar signals from one or more other control registers. The arbitration logic compares the priority level of competing interrupt requests, and it passes the interrupt of highest priority to the CPU. The corresponding flag is set in the interrupt flag register (IFR), indicating that the interrupt is pending. The CPU then must decide whether to acknowledge the request. Maskable hardware interrupts are acknowledged only after certain conditions are met:

- ❑ **Priority is highest.** When more than one hardware interrupt is requested at the same time, the 'x240 services them according to the set priority ranking.
- ❑ **INTM bit is 0.** The interrupt mode (INTM) bit, bit 9 of status register ST0, enables or disables all maskable interrupts:
  - ■ When INTM = 0, all unmasked interrupts are enabled.
  - ■ When INTM = 1, all unmasked interrupts are disabled.
- ❑ INTM is set to 1 automatically when the CPU acknowledges an interrupt (except when initiated by the TRAP instruction) and at reset. It can be set and cleared by software.
- ❑ **IMR mask bit is 1.** Each of the maskable interrupt lines has a mask bit in the interrupt mask register (IMR). To unmask an interrupt line, set its IMR bit to 1.

When the CPU acknowledges a maskable hardware interrupt, it jams the instruction bus with the INTR instruction. This instruction forces the PC to the appropriate address from which the CPU fetches the software vector. This vector leads to an interrupt service routine.

Usually, the interrupt service routine reads the peripheral-vector-address offset from the peripheral-vector-address register (see Table 11–10) to branch to code that is meant for the specific interrupt source that initiated the interrupt request. The 'x240 includes a phantom-interrupt vector offset (0000h), which is a system interrupt integrity feature that allows a controlled exit from an improper interrupt sequence. If the CPU acknowledges a request from a peripheral when, in fact, no peripheral has requested an interrupt, the phantom-interrupt vector is read from the interrupt-vector register.

Table 11–9 summarizes the interrupt sources, overall priority, vector address/offset, source, and function of each interrupt available on the 'x240.

Table 11–9. TMS320x240 Interrupt Locations and Priorities

Overall Priority	Interrupt Name	DSP-Core Interrupt and Address	Peripheral Vector Address	Peripheral Vector Address Offset	Maskable	’x240 Module	Function Interrupt
1 Highest	RS	$\overline{\text{RS}}$ 0000h	N/A		N	Core, SD	External, system reset (RESET)
2	Reserved	INT7 0026h	N/A	N/A	N	DSP Core	Emulator trap
3	NMI	NMI 0024h	N/A	0002h	N	Core, SD	External user interrupt
4 5 6	XINT1 XINT2 XINT3	INT1 0002h	SYSIVR	0001h 0011h 001Fh	Y Y Y	SD	High-priority external user interrupts
7	SPIINT			0005h	Y	SPI	High-priority SPI interrupt
8	RXINT			0006h	Y	SCI	SCI receiver interrupt
9	TXINT			0007h	Y	SCI	SCI transmitter interrupt
10	RTINT	(System)	701E h	0010h	Y	WDT	Real time interrupt
11	PDPINT		7432h	0020h	Y	External	Power-drive protection Interrupt
12	CMP1INT			0021h	Y	EV.CMP1	Full Compare 1 interrupt
13	CMP2INT			0022h	Y	EV.CMP2	Full Compare 2 interrupt
14	CMP3INT			0023h	Y	EV.CMP3	Full Compare 3 interrupt
15	SCMP1INT	$\overline{\text{INT2}}$ 0004h		0024h	Y	EV.CMP4	Simple compare 1 interrupt
16	SCMP2INT			0025h	Y	EV.CMP5	Simple compare 2 interrupt
17	SCMP3INT			0026h	Y	EV.CMP6	Simple compare 3 interrupt
18	TPINT1			0027h	Y	EV.GPT1	Timer1-period interrupt
19	TCINT1	(Event Manager Group A)		0028h	Y	EV.GPT1	Timer1-compare interrupt
20	TUFINT1			0029h	Y	EV.GPT1	Timer1-underflow interrupt
21	TOFINT1			002Ah	Y	EV.GPT1	Timer1-overflow interrupt

Table 11–9. TMS320x240 Interrupt Locations and Priorities (Continued)

Overall Priority	Interrupt Name	DSP-Core Interrupt and Address	Peripheral Vector Address	Peripheral Vector Address Offset	Maskable	’x240 Module	Function Interrupt
22	TPINT2	INT3 0006h (EV INTB)	7433h	002Bh	Y	EV.GPT2	Timer2-period interrupt
23	TCINT2			002Ch	Y	EV.GPT2	Timer2-compare interrupt
24	TUFINT2			002Dh	Y	EV.GPT2	Timer2-underflow interrupt
25	TOFINT2			002Eh	Y	EV.GPT2	Timer2-overflow interrupt
26	TPINT3			002Fh	Y	EV.GPT3	Timer3-period interrupt
27	TCINT3			0030h	Y	EV.GPT3	Timer3-compare interrupt
28	TUFINT3	(Event Manager Group B)		0031h	Y	EV.GPT3	Timer3-underflow interrupt
29	TOFINT3			0032h	Y	EV.GPT3	Timer3-overflow interrupt
30	CAPINT1	INT4 0008h	7434h	0033h	Y	EV.CAP1	Capture 1 interrupt
31	CAPINT2			0034h	Y	EV.CAP2	Capture 2 interrupt
32	CAPINT3	(Event Manager Group C)		0035h	Y	EV.CAP3	Capture 3 interrupt
33	CAPINT4			0036h	Y	EV.CAP4	Capture 4 interrupt
34	SPIINT	INT5 000Ah	SYSIVR 701Eh	0005h	Y	SPI	Low-priority SPI interrupt
35	RXINT			0006h	Y	SCI	SCI receiver interrupt
36	TXINT	(System)		0007h	Y	SCI	SCI transmitter interrupt
37	ADCINT	INT6 00Ch	SYSIVR	0004h	Y	ADC	Analog-to-digital interrupt
38 39 40	XINT1 XINT2 XINT3	(System)	701Eh	0001h 0011h 001Fh	Y Y Y	External pins	Low-priority external user interrupts
41	Reserved	000Eh	N/A		Y	DSP Core	Used for analysis
N/A	TRAP	0022h	N/A		N/A		TRAP instruction vector

11.5.3 External Interrupts

The 'x240 has five external interrupts. These interrupts include:

- **XINT1.** Type A interrupt. The XINT1 control register (at 7070h) provides control and status for this interrupt. XINT1 can be used as a high-priority (Level 1) or low-priority (Level 6) maskable interrupt or as a general-purpose input pin.
- **NMI.** Type A interrupt. The NMI control register (at 7072h) provides control and status for this interrupt. NMI is a nonmaskable external interrupt or a general-purpose input pin.
- **XINT2.** Type C interrupt. The XINT2 control register (at 7078h) provides control and status for this interrupt. XINT2 can be used as a high-priority (Level 1) or low-priority (Level 6) maskable interrupt or a general-purpose I/O pin.
- **XINT3.** Type C interrupt. The XINT3 control register (at 707Ah) provides control and status for this interrupt. XINT3 can be used as a high-priority (Level 1) or low priority (Level 6) maskable interrupt or as a general-purpose I/O pin.
- **PDPINT.** This interrupt is provided for safe operation of the power converter and motor drive. This maskable interrupt can put the timers and PWM output pins in high-impedance states and inform the CPU in case of motor drive abnormalities such as overvoltage, overcurrent, and excessive temperature rise. PDPINT is a Level 2 interrupt. Figure 11–16 shows the wake up sequence from a power down.

Table 11–10 is a summary of the external interrupt capability of the 'x240.

Figure 11–16. Waking Up the Device from Power–Down

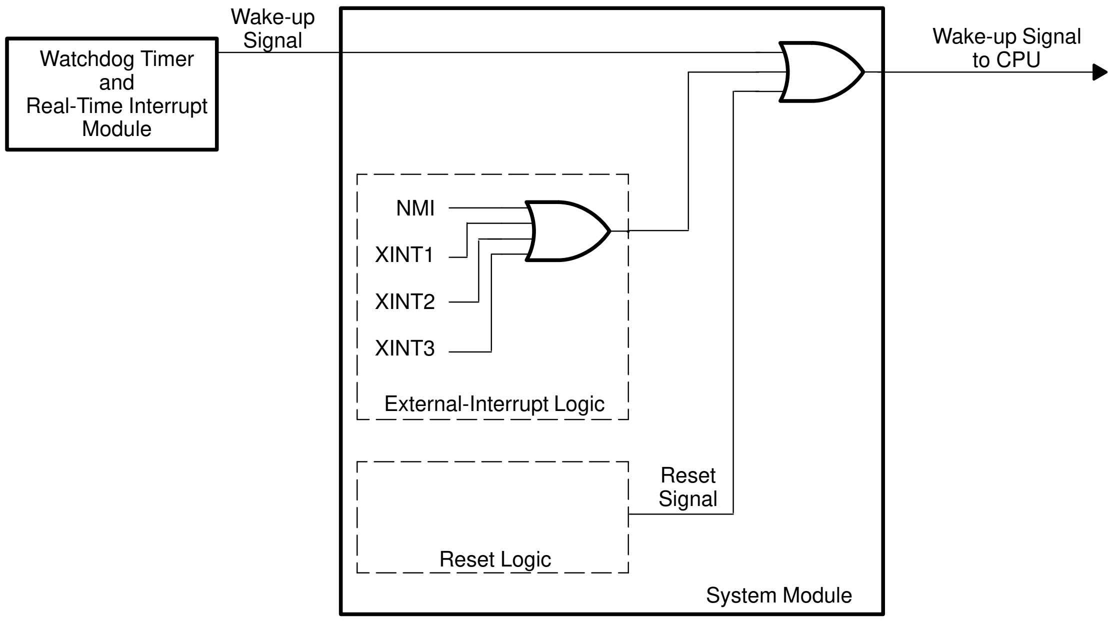


Table 11–10. External Interrupt Types and Functions

External Interrupt	Control Register Address	Interrupt Type	Can Do NMI?	Digital I/O Pin	Maskable?
XINT1	7070h	A	No	Input only	Yes (Level 1 or 6)
NMI	7072h	A	Yes	Input only	No
N/C	7074h	B		Reserved	
N/C	7076h	B		Reserved	
XINT2	7078h	C	No	I/O	Yes (Level 1 or 6)
XINT3	707Ah	C	No	I/O	Yes (Level 1 or 6)
N/C	707Ch	PM		Reserved	
N/C	707Eh	PM		Reserved	
PDPINT	742Ch	N/A	N/A	N/A	Yes (Level 2)

11.6 Clock Generation

The TMS320x240 has an on-chip, PLL-based clock module. This module provides all necessary clocking signals for the device as well as control for low-power mode entry. The only external component necessary for this module is an external fundamental reference crystal.

The ’x240 has two basic clocking domains: the CPU clock domain (CPUCLK), and the system clock domain (SYSCLK). The CPU, memories, external memory interface, and event manager are located in the CPU clock domain. All other peripherals are in the system clock domain. The CPUCLK runs at 2 × or 4 × the frequency of the SYSCLK; that is, CPUCLK = 20 MHz, SYSCLK = 10 MHz, or CPUCLK = 20 MHz, SYSCLK = 5 MHz.

The clock module includes three external pins:

XTAL1/CLKIN	Crystal input/clock source
XTAL2	Output to crystal
$\overline{\text{OSCBYP}}$	Oscillator bypass

For the external pins, if $\overline{\text{OSCBYP}}$ = 5 V, then the oscillator is enabled and if $\overline{\text{OSCBYP}}$ = 0 V, then the oscillator is bypassed. In oscillator-bypass mode, an external TTL clock must be applied to the OSCIN pin.

11.7 Low-Power Mode

The TMS320x240 has four low-power modes (idle 1, idle 2, standby, and halt). The low-power modes reduce the operating power by reducing or stopping the activity of various modules (by stopping their clocks). The two PLLPM bits of the clock-module-control register (CKCR0) select which of the low-power modes the device enters when executing an IDLE instruction. Reset or an unmasked interrupt from any source causes the device to exit from idle 1 low-power mode. A real-time interrupt (from WD/RTI) causes the device to exit all except the halt low-power mode (idle 1, idle 2, and standby). This is a wake-up interrupt.

Reset or any of the four external interrupts (NMI, XINT1, XINT2, or XINT3, if enabled) causes the device to exit from any low-power mode (idle 1, idle 2, standby and halt). The external interrupts are all wake-up interrupts. Any interrupt designed to allow an exit from a low-power mode must be enabled individually and globally to bring the device out of a low-power mode properly. It is important to ensure that the desired low-power mode exit path is enabled before entering a low-power mode.

Table 11–11 lists the low-power modes.

Table 11–11. Low-Power Modes

Low Power Mode	PLLPM(1:0) Bits in CKCR0	CPU Clock Status	System Clock Status	Watchdog Clock Status	PLL Status	Oscillator Status	Exit Condition	Power	Description
X + not IDLE	XX	On	On	On	On	On	—	$>$40 mA	Run
0 + IDLE	00	Off	On	On	On	On	Any Interrupt, Reset	15 mA	Idle1
1 + IDLE	01	Off	Off	On	On	On	Wake-Up Interrupt, Reset	4 mA	Idle2
2 + IDLE	10	Off	Off	Off	Off	On	Wake-Up Interrupt, Reset	1 mA	Standby
3 + IDLE	11	Off	Off	Off	Off	Off	Wake-Up Interrupt, Reset	$<$30 μA	Halt

Section II

11.8 Summary of Programmable Registers on the TMS320C240

Table 11–12 provides a summary of all the programmable registers on the ’C240.

Table 11–12. Addresses of TMS320C240 Registers

			Shown in	
Address	**Register**	**Name**	**Figure**	**Page**
Internal	ST0	Status register 0	Figure 11–17	11-34
Internal	ST1	Status register 1	Figure 11–18	11-34
0004h	IMR	Interrupt mask register	Figure 11–19	11-35
0006h	IFR	Interrupt flag register	Figure 11–20	11-35
7018h	SYSCR	System control register	Figure 11–21	11-35
701Ah	SYSSR	System status register	Figure 11–22	11-35
702Bh	CKCR0	Clock control register 0	Figure 11–23	11-36
702Dh	CKCR1	Clock control register 1	Figure 11–24	11-36
7032h	ADCTRL1	ADC control register 1	Figure 11–25	11-36
7034h	ADCTRL2	ADC control register 2	Figure 11–26	11-36
7040h	SPICCR	SPI configuration control register	Figure 11–27	11-37
7041h	SPICTL	SPI operation control register	Figure 11–28	11-37
7042h	SPISTS	SPI status register	Figure 11–29	11-37
7044h	SPIBRR	SPI baud rate register	Figure 11–30	11-37
7046h	SPIEMU	SPI emulation buffer register	Figure 11–31	11-37
7047h	SPIBUF	SPI serial input buffer register	Figure 11–32	11-38
7049h	SPIDAT	SPI serial data register	Figure 11–33	11-38
704Dh	SPIPC1	SPI port control register 1	Figure 11–34	11-38
704Eh	SPIPC2	SPI port control register 2	Figure 11–35	11-38
704Fh	SPIPRI	SPI priority control register	Figure 11–36	11-38
7050h	SCICCR	SCI communication control register	Figure 11–37	11-39
7051h	SCICTL1	SCI control register 1	Figure 11–38	11-39
7052h	SCIHBAUD	SCI baud select register, high bits	Figure 11–39	11-39

Table 11–12. Addresses of TMS320C240 Registers (Continued)

			Shown in	
Address	**Register**	**Name**	**Figure**	**Page**
7053h	SCILBAUD	SCI baud select register, low bits	Figure 11–40	11-39
7054h	SCICTL2	SCI control register 2	Figure 11–41	11-39
7055h	SCIRXST	SCI receiver status register	Figure 11–42	11-40
705Eh	SCIPC2	SCI port control register 2	Figure 11–43	11-40
705Fh	SCIPRI	SCI priority control register	Figure 11–44	11-40
7070h	XINT1	External interrupt control register	Figure 11–45	11-40
7072h	NMI	External interrupt control register	Figure 11–46	11-40
7078h	XINT2	External interrupt control register	Figure 11–47	11-41
707Ah	XINT3	External interrupt control register	Figure 11–48	11-41
7090h	OCRA	I/O mux control register A	Figure 11–49	11-41
7092h	OCRB	I/O mux control register B	Figure 11–50	11-41
7098h	PADATDIR	I/O port A data and direction register	Figure 11–51	11-42
709Ah	PBDATDIR	I/O port B data and direction register	Figure 11–52	11-42
709Ch	PCDATDIR	I/O port C data and direction register	Figure 11–53	11-42
7400h	GPTCON	General purpose timer control register.	Figure 11–54	11-43
7404h	T1CON	GP Timer 1 control register.	Figure 11–55	11-43
7408h	T2CON	GP Timer 2 control register.	Figure 11–56	11-43
740Ch	T3CON	GP Timer 3 control register.	Figure 11–57	11-44
7411h	COMCON	Compare control register.	Figure 11–58	11-44
7413h	ACTR	Full compare action control register.	Figure 11–59	11-44
7414h	SACTR	Simple compare action control register.	Figure 11–60	11-45
7415h	DBTCON	Dead-band timer control register.	Figure 11–61	11-45
7420h	CAPCON	Capture control register.	Figure 11–62	11-45
7422h	CAPFIFO	Capture FIFO status register.	Figure 11–63	11-46
742Ch	EVIMRA	EV interrupt mask register A.	Figure 11–64	11-46

Table 11–12. Addresses of TMS320C240 Registers (Continued)

			Shown in	
Address	Register	Name	Figure	Page
742Dh	EVIMRB	EV interrupt mask register B.	Figure 11–65	11-46
742Eh	EVIMRC	EV interrupt mask register C.	Figure 11–66	11-46
742Fh	EVIFRA	EV interrupt flag register A.	Figure 11–67	11-47
7430h	EVIFRB	EV interrupt flag register B.	Figure 11–68	11-47
7431h	EVIFRC	EV interrupt flag register C.	Figure 11–69	11-47
7432h	EVIVRA	EV interrupt vector register A.	Figure 11–70	11-47
7433h	EVIVRB	EV interrupt vector register B.	Figure 11–71	11-47
7434h	EVIVRC	EV interrupt vector register C.	Figure 11–72	11-47
0h†	SEG_CTR	Flash segment control register	Figure 11–73	11-48
FFFFh‡	WSGR	Wait-state generator control register	Figure 11–74	11-48

† Address in program memory space.
‡ Address in I/O space.

11.8.1 CPU Registers

CPU Status Registers (ST0 and ST1)

Figure 11–17. Status Register ST0 — Internal CPU register

15–13	12	11	10	9	8–0
ARP	OV	OVM	1	INTM	DP
R/W–x	R/W–0	R/W–x		R/W–1	R/W–x

Figure 11–18. Status Register ST1 — Internal CPU register

15–13	12	11	10	9	8	7	6	5	4	3	2	1–0
ARB	CNF	TC	SXM	C	1	1	1	1	XF	1	1	PM
R/W–x	R/W–0	R/W–x	R/W–1	R/W–1					R/W–1			R/W–00

CPU Interrupt Registers (IMR and IFR)

Figure 11–19. Interrupt Mask Register (IMR) — Address 0004h

15–6	5	4	3	2	1	0
Reserved	INT6	INT5	INT4	INT3	INT2	INT1
0	R/W–0	R/W–0	R/W–0	R/W–0	R/W–0	R/W–0

Figure 11–20. Interrupt Flag Register (IFR) — Address 0006h

15–6	5	4	3	2	1	0
Reserved	INT6	INT5	INT4	INT3	INT2	INT1
0	R/W1C–0	R/W1C–0	R/W1C–0	R/W1C–0	R/W1C–0	R/W1C–0

System Control Register (SYSCR)

Figure 11–21. System Control Register (SYSCR) — Address 7018h

15	14	13–8	7	6	5–0
RESET1	RESET0	Reserved	CLKSRC1	CLKSRC0	Reserved
R/W–0	R/W–1		R/W–0	R/W–0	

System Status Register (SYSSR)

Figure 11–22. System Status Register (SYSSR) — Address 701Ah

15	14–13	12	11	10	9	8–6	5	4	3	2–1	0
PORST	Res	ILLADR	Res	SWRST	WDRST	Res	HPO	Res	VCCAOR	Res	VECRD
R/C–x		R/C–x		R/C–x	R/C–x		R/C–i		R–1		R–0

11.8.2 PLL Clock Registers

Clock Control Register 0 (CKCR0)

Figure 11–23. Clock Control Register 0 (CKCR0) — Address 702Bh

7	6	5	4	3	2	1	0
CLKMD(1)	CLKMD(0)	PLLOCK(1)	PLLOCK(0)	PLLPM(1)	PLLPM(0)	ACLKENA	PLLPS
RW–x	RW–x	R–x	R–x	RW–0	RW–0	RW–x	RW–0

Clock Control Register 1 (CKCR1)

Figure 11–24. Clock Control Register 1 (CKCR1) — Address 702Dh

7	6	5	4	3	2	1	0
CKINF(3)	CKINF(2)	CKINF(1)	CKINF(0)	PLLDIV(2)	PLLFB(2)	PLLFB(1)	PLLFB(0)
RW–x	RW–x	RW–x	RW–x	RW–x	RW–x	RW–x	RW–x

11.8.3 Dual 10-Bit Analog to Digital Converter (ADC) Registers

ADC Control Register 1 (ADCTRL1)

Figure 11–25. ADC Control Register 1 (ADCTRL1) — Address 7032h

15	14	13	12	11	10	9	8
Suspend–soft	Suspend–free	ADCIMSTART	ADC1EN	ADC2EN	ADCCONRUN	ADCINTEN	ADCINTFLAG

7	6–4	3–1	0
ADCEOC	ADC2CHSEL	ADC1CHSEL	ADCSOC
R–0	SRW–0	SRW–0	SRW–0

ADC Control Register 2 (ADCTRL2)

Figure 11–26. ADC Control Register 2 (ADCTRL2) — Address 7034h

15–11	10	9	8
Reserved	ADCEVSOC	ADCEXTSOC	Reserved
	SRW–0	SRW–0	

7–6	5	4–3	2–0
ADCFIFO1	Reserved	ADCFIFO2	ADCPSCALE
R–0		R–0	SRW–0

11.8.4 Serial Peripheral Interface (SPI) Registers

SPI Configuration Control Register (SPICCR)

Figure 11–27. SPI Configuration Control Register (SPICCR) — Address 7040h

7	6	5 – 3	2	1	0
SPI SW RESET	CLOCK POLARITY	Reserved	SPI CHAR2	SPI CHAR1	SPI CHAR0
RW–0	RW–0		RW–0	RW–0	RW–0

SPI Operation Control Register (SPICTL)

Figure 11–28. SPI Operation Control Register (SPICTL) — Address 7041h

7 – 5	4	3	2	1	0
Reserved	OVERRUN INT ENA	CLOCK PHASE	MASTER/ SLAVE	TALK	SPI INT ENA
	RW–0	RW–0	RW–0	RW–0	RW–0

SPI Status Register (SPISTS)

Figure 11–29. SPI Status Register (SPISTS) — Address 7042h

7	6	5 – 0
RECEIVER OVERRUN	SPI INT FLAG	Reserved
RC–0	R–0	

SPI Baud Rate Register (SPIBRR)

Figure 11–30. SPI Baud Rate Register (SPIBRR) — Address 7044h

7	6	5	4	3	2	1	0
Reserved	SPI BIT RATE 6	SPI BIT RATE 5	SPI BIT RATE 4	SPI BIT RATE 3	SPI BIT RATE 2	SPI BIT RATE 1	SPI BIT RATE 0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

SPI Emulation Buffer Register (SPIEMU)

Figure 11–31. SPI Emulation Buffer Register (SPIEMU) — Address 7046h

7	6	5	4	3	2	1	0
ERCVD7	ERCVD6	ERCVD5	ERCVD4	ERCVD3	ERCVD2	ERCVD1	ERCVD0
R–x	R–x	R–x	R–x	R–x	R–x	R–x	R–x

SPI Serial Input Buffer Register (SPIBUF)

Figure 11–32. SPI Serial Input Buffer Register (SPIBUF) — Address 7047h

7	6	5	4	3	2	1	0
RCVD7	RCVD6	RCVD5	RCVD4	RCVD3	RCVD2	RCVD1	RCVD0
R–x	R–x	R–x	R–x	R–x	R–x	R–x	R–x

SPI Serial Data Register (SPIDAT)

Figure 11–33. SPI Serial Data Register (SPIDAT) — Address 7049h

7	6	5	4	3	2	1	0
SDAT7	SDAT6	SDAT5	SDAT4	SDAT3	SDAT2	SDAT1	SDAT0
RW–x	RW–x	RW–x	RW–x	RW–x	RW–x	RW–x	RW–x

SPI Port Control Register 1 (SPIPC1)

Figure 11–34. SPI Port Control Register 1 (SPIPC1) — Address 704Dh

7	6	5	4	3	2	1	0
SPISTE DATA IN	SPISTE DATA OUT	SPISTE FUNCTION	SPISTE DATA DIR	SPICLK DATA IN	SPICLK DATA OUT	SPICLK FUNCTION	SPICLK DATA DIR
R–x	RW–0	RW–0	RW–0	R–x	RW–0	RW–0	RW–0

SPI Port Control Register 2 (SPIPC2)

Figure 11–35. SPI Port Control Register 2 (SPIPC2) — Address 704Eh

7	6	5	4	3	2	1	0
SPISIMO DATA IN	SPISIMO DATA OUT	SPISiMO FUNCTION	SPISiMO DATA DIR	SPISOMI DATA IN	SPISOMI DATA OUT	SPISOMI FUNCTION	SPISOMI DATA DIR
R–x	RW–0	RW–0	RW–0	R–x	RW–0	RW–0	RW–0

SPI Priority Control Register (SPIPRI)

Figure 11–36. SPI Priority Control Register (SPIPRI) — Address 704Fh

7	6	5	4 – 0
Reserved	SPI PRIORITY	SPI ESPEN	Reserved
	RW–0	RW–0	

11.8.5 Serial Communications Interface (SCI) Registers

SCI Communication Control Register (SCICCR)

Figure 11–37. SCI Communication Control Register (SCICCR) — Address 7050h

7	6	5	4	3	2	1	0
STOP BITS	EVEN/ODD PARITY	PARITY ENABLE	SCI ENA	ADDR/IDLE MODE	SCI CHAR2	SCI CHAR1	SCI CHAR0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

SCI Control Register 1 (SCICTL1)

Figure 11–38. SCI Control Register 1 (SCICTL1) — Address 7051h

7	6	5	4	3	2	1	0
Reserved	RX ERR INT ENA	SW RESET	CLOCK ENA	TXWAKE	SLEEP	TXENA	RXENA
	RW–0	RW–0	RW–0	RS–0	RW–0	RW–0	RW–0

SCI Baud-Select Registers (SCIHBAUD and SCILBAUD)

Figure 11–39. SCI Baud-Select Register (SCIHBAUD) — Address 7052h

15	14	13	12	11	10	9	8
BAUD15 (MSB)	BAUD14	BAUD13	BAUD12	BAUD11	BAUD10	BAUD9	BAUD8
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

Figure 11–40. SCI Baud-Select Register (SCILBAUD) — Address 7053h

7	6	5	4	3	2	1	0
BAUD7	BAUD6	BAUD5	BAUD4	BAUD3	BAUD2	BAUD1	BAUD0 (LSB)
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

SCI Control Register 2 (SCICTL2)

Figure 11–41. SCI Control Register 2 (SCICTL2) — Address 7054h

7	6	5 – 2	1	0
TXRDY	TX EMPTY	Reserved	RX/BK INT ENA	TX INT ENA
R–1	R–1		RW–0	RW–0

SCI Receiver Status Register (SCIRXST)

Figure 11–42. SCI Receiver Status Register (SCIRXST) — Address 7055h

7	6	5	4	3	2	1	0
RX ERROR	RXRDY	BRKDT	FE	OE	PE	RXWAKE	Reserved
R–0	R–0	R–0	R–0	R–0	R–0	R–0	

SCI Port Control Register 2 (SCIPC2)

Figure 11–43. SCI Port Control Register 2 (SCIPC2) — Address 705Eh

7	6	5	4	3	2	1	0
SCITXD DATA IN	SCITXD DATA OUT	SCITXD FUNCTION	SCITXD DATA DIR	SCIRXD DATA IN	SCIRXD DATA OUT	SCIRXD FUNCTION	SCIRXD DATA DIR
R–x	RW–0	RW–0	RW–0	R–x	RW–0	RW–0	RW–0

SCI Priority Control Register (SCIPRI)

Figure 11–44. SCI Priority Control Register (SCIPRI) — Address 705Fh

7	6	5	4	3 – 0
Reserved	SCITX PRIORITY	SCIRX PRIORITY	SCI ESPEN	Reserved
	RW–0	RW–0	RW–0	

11.8.6 External Interrupt Control Registers (XINT1, NMI, XINT2, and XINT3)

Figure 11–45. External Interrupt Control Register (XINT1) — Address 7070h

15	14–7	6	5	4–3	2	1	0
XINTA1 Flag	Reserved	XINTA1 Pin data	XINTA1 NMI	Reserved	XINTA1 Polarity	XINTA1 Priority	XINTA1 Enable

Figure 11–46. External Interrupt Control Register (NMI) — Address 7072h

15	14–7	6	5	4–3	2	1–0
XINTA–NMI Flag	Reserved	XINTA–NMI Pin data	XINTA–NMI NMI	Reserved	XINTA–NMI Polarity	Reserved

Figure 11–47. External Interrupt Control Register (XINT2) — Address 7078h

15	14–7	6	5	4	3	2	1	0
XINTC1 Flag	Reserved	XINTC1 Pin data	Reserved	XINTC1 Data dir	XINTC1 Data out	XINTC1 Polarity	XINTC1 Priority	XINTC1 Enable

Figure 11–48. External Interrupt Control Register (XINT3) — Address 707Ah

15	14–7	6	5	4	3	2	1	0
XINTC2 Flag	Reserved	XINTC2 Pin data	Reserved	XINTC2 Data dir	XINTC2 Data out	XINTC2 Polarity	XINTC2 Priority	XINTC2 Enable

11.8.7 Digital I/O Control Registers

I/O Mux Control Registers (OCRA and OCRB)

Figure 11–49. I/O Mux Control Register A (OCRA) — Address 7090h

15	14	13	12	11	10	9	8
CRA.15	CRA.14	CRA.13	CRA.12	CRA.11	CRA.10	CRA.9	CRA.8
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
Reserved				CRA.3	CRA.2	CRA.1	CRA.0
RW–0				RW–0	RW–0	RW–0	RW–0

Figure 11–50. I/O Mux Control Register B (OCRB) — Address 7092h

15	14	13	12	11	10	9	8
Reserved							
RW–0							

7	6	5	4	3	2	1	0
CRB.7	CRB.6	CRB.5	CRB.4	CRB.3	CRB.2	CRB.1	CRB.0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

I/O Port Data and Direction Registers (PADATDIR, PBDATDIR, and PCDATDIR)

Figure 11–51. I/O Port A Data and Direction Register (PADATDIR) — Address 7098h

15	14	13	12	11	10	9	8
Reserved				A3DIR	A2DIR	A1DIR	A0DIR
				RW–0	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
Reserved				IOPA3	IOPA2	IOPA1	IOPA0
				RW–0	RW–0	RW–0	RW–0

Figure 11–52. I/O Port B Data and Direction Register (PBDATDIR) — Address 709Ah

15	14	13	12	11	10	9	8
B7DIR	B6DIR	B5DIR	B4DIR	B3DIR	B2DIR	B1DIR	B0DIR
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
IOPB7	IOPB6	IOPB5	IOPB4	IOPB3	IOPB2	IOPB1	IOPB0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

Figure 11–53. I/O Port C Data and Direction Register (PCDATDIR) — Address 709Ch

15	14	13	12	11	10	9	8
C7DIR	C6DIR	C5DIR	C4DIR	C3DIR	C2DIR	C1DIR	C0DIR
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
IOPC7	IOPC6	IOPC5	IOPC4	IOPC3	IOPC2	IOPC1	IOPC0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

11.8.8 General Purpose (GP) Timer Registers

GP Timer Control Register (GPTCON)

Figure 11–54. GP Timer Control Register (GPTCON) — Address 7400h

15	14	13	12–11	10–9	8–7
T3STAT	T2STAT	T1STAT	T3TOADC	T2TOADC	T1TOADC
R–1	R–1	R–1	RW–0	RW–0	RW–0

6	5–4	3–2	1–0
TCOMPOE	T3PIN	T2PIN	T1PIN
RW–0	RW–0	RW–0	RW–0

GP Timer Control Registers (T1CON, T2CON, and T3CON)

Figure 11–55. GP Timer 1 Control Register (T1CON) — Address 7404h

15	14	13	12	11	10	9	8
Free	Soft	TMODE2	TMODE1	TMODE0	TPS2	TPS1	TPS0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
TSWT1	TENABLE	TCLKS1	TCLKS0	TCLD1	TCLD0	TECMPR	SELT1PR
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

Figure 11–56. GP Timer 2 Control Register (T2CON) — Address 7408h

15	14	13	12	11	10	9	8
Free	Soft	TMODE2	TMODE1	TMODE0	TPS2	TPS1	TPS0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
TSWT1	TENABLE	TCLKS1	TCLKS0	TCLD1	TCLD0	TECMPR	SELT1PR
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

Figure 11–57. GP Timer 3 Control Register (T3CON) — Address 740Ch

15	14	13	12	11	10	9	8
Free	Soft	TMODE2	TMODE1	TMODE0	TPS2	TPS1	TPS0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0
7	6	5	4	3	2	1	0
TSWT1	TENABLE	TCLKS1	TCLKS0	TCLD1	TCLD0	TECMPR	SELT1PR
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

11.8.9 Compare Units Registers

Compare Control Register (COMCON)

Figure 11–58. Compare Control Register (COMCON) — Address 7411h

15	14	13	12	11	10	9	8
CENABLE	CLD1	CLD0	SVENABLE	ACTRLD1	ACTRLD0	FCOMPOE	SCOMPOE
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0
7	6	5	4	3	2	1	0
SELTMR	SCLD1	SCLD0	SACTRLD1	SACTRLD0	SELCMP3	SELCMP2	SELCMP1
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

Full Compare Action Control Register (ACTR)

Figure 11–59. Full Compare Action Control Register (ACTR) — Address 7413h

15	14	13	12	11	10	9	8
SVRDIR	D2	D1	D0	CMP6ACT1	CMP6ACT0	CMP5ACT1	CMP5ACT0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0
7	6	5	4	3	2	1	0
CMP4ACT1	CMP4ACT0	CMP3ACT1	CMP3ACT0	CMP2ACT1	CMP2ACT0	CMP1ACT1	CMP1ACT0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

Simple Compare Action Control Register (SACTR)

Figure 11–60. Simple Compare Action Control Register (SACTR) — Address 7414h

15–8
Reserved

7–6	5	4	3	2	1	0
Reserved	SCMP3ACT1	SCMP3ACT0	SCMP2ACT1	SCMP2ACT0	SCMP1ACT1	SCMP1ACT0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

11.8.10 Dead-Band Timer Control Register (DBTCON)

Figure 11–61. Dead-Band Timer Control Register (DBTCON) — Address 7415h

15	14	13	12	11	10	9	8
DBT7	DBT6	DBT5	DBT4	DBT3	DBT2	DBT1	DBT0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

7	6	5	4	3	2–0
EDBT3	EDBT2	EDBT1	DBTPS1	DBTPS0	Reserved
RW–0	RW–0	RW–0	RW–0	RW–0	

11.8.11 Capture Unit Registers

Capture Control Register (CAPCON)

Figure 11–62. Capture Control Register (CAPCON) — Address 7420h

15	14–13	12	11	10	9	8
CAPRES	CAPQEPN	CAP3EN	CAP4EN	CAP34TSEL	CAP12TSEL	CAP4TOADC
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

7–6	5–4	3–2	1–0
CAP1EDGE	CAP2EDGE	CAP3EDGE	CAP4EDGE
RW–0	RW–0	RW–0	RW–0

Capture FIFO Status Register (CAPFIFO)

Figure 11–63. Capture FIFO Status Register (CAPFIFO) — Address 7422h

15–14	13–12	11–10	9–8
CAP4FIFO	CAP3FIFO	CAP2FIFO	CAP1FIFO
R–0	R–0	R–0	R–0

7	6	5	4	3	2	1	0
CAPFIFO15	CAPFIFO14	CAPFIFO13	CAPFIFO12	CAPFIFO11	CAPFIFO10	CAPFIFO9	CAPFIFO8
W–0	W–0	W–0	W–0	W–0	W–0	W–0	W–0

11.8.12 Event Manager (EV) Interrupt Registers

EV Interrupt Mask Registers (EVIMRA, EVIMRB, and EVIMRC)

Figure 11–64. EV Interrupt Mask Register A (EVIMRA) — Address 742Ch

15–11	10	9	8
Reserved	T1OFINT ENABLE	T1UFINT ENABLE	T1CINT ENABLE
	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
T1PINT ENABLE	SCMP3INT ENABLE	SCMP2INT ENABLE	SCMP1INT ENABLE	CMP3INT ENABLE	CMP2INT ENABLE	CMP1INT ENABLE	PDPINT ENABLE
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

Figure 11–65. EV Interrupt Mask Register B (EVIMRB) — Address 742Dh

15–8	7	6	5	4	3	2	1	0
Reserved	T3OFINT ENABLE	T3UFINT ENABLE	T3CINT ENABLE	T3PINT ENABLE	T2OFINT ENABLE	T2UFINT ENABLE	T2CINT ENABLE	T2PINT ENABLE
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

Figure 11–66. EV Interrupt Mask Register C (EVIMRC) — Address 742Eh

15–4	3	2	1	0
Reserved	CAP4INT ENABLE	CAP3INT ENABLE	CAP2INT ENABLE	CAP1INT ENABLE
	RW–0	RW–0	RW–0	RW–0

EV Interrupt Flag Registers (EVIFRA, EVIFRB, and EVIFRC)

Figure 11–67. EV Interrupt Flag Register A (EVIFRA) — Address 742Fh

15–11	10	9	8
Reserved	T1OFINT	T1UFINT	T1CINT
	RW–0	RW–0	RW–0

7	6	5	4	3	2	1	0
T1PINT	SCMP3INT	SCMP2INT	SCMP1INT	CMP3INT	CMP2INT	CMP1INT	PDPINT
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

Figure 11–68. EV Interrupt Flag Register B (EVIFRB) — Address 7430h

15–8	7	6	5	4	3	2	1	0
Reserved	T3OFINT	T3UFINT	T3CINT	T3PINT	T2OFINT	T2UFINT	T2CINT	T2PINT
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

Figure 11–69. EV Interrupt Flag Register C (EVIFRC) — Address 7431h

15–4	3	2	1	0
Reserved	CAP4INT	CAP3INT	CAP2INT	CAP1INT
	RW–0	RW–0	RW–0	RW–0

EV Interrupt Vector Registers (EVIVRA, EVIVRB, and EVIVRC)

Figure 11–70. EV Interrupt Vector Register A (EVIVRA) — Address 7432h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0
R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0

Figure 11–71. EV Interrupt Vector Register B (EVIVRB) — Address 7433h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0
R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0

Figure 11–72. EV Interrupt Vector Register C (EVIVRC) — Address 7434h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0
R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0

11.8.13 Flash Segment Control Register (SEG_CTR)

Figure 11–73. Segment Control Register (SEG_CTR) — Program Space Address 0h

15–8	7	6	5	4	3	2–1	0
SEG7–0	Reserved	KEY1	KEY0	VER1	VER0	WRITE/ERASE	EXE
RW–0		RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

11.8.14 Wait-State Generator Control Register (WSGR)

Figure 11–74. Wait-State Generator Control Register (WSGR) — I/O Space Address FFFFh

15–4	3	2	1	0
Reserved	AVIS	ISWS	DSWS	PSWS
0	W–1	W–1	W–1	W–1

Index