TMS370 Family Data Manual

IMPORTANT NOTICE

Texas Instruments Incorporated (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to current specifications in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Please be aware that TI products are not intended for use in life-support appliances, devices, or systems. Use of TI product in such applications requires the written approval of the appropriate TI officer. Certain applications using semiconductor devices may involve potential risks of personal injury, property damage, or loss of life. In order to minimize these risks, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards. Inclusion of TI products in such applications is understood to be fully at the risk of the customer using TI devices or systems.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Read This First

About This Manual

This manual describes the TMS370 family of microcontroller devices. These devices have been revised to include more robust features that enhance performance and enable new application technologies. The specifications and descriptions included in this manual apply to the TMS370CxxxA devices; all differences for the TMS370Cxxx devices are described in Appendix A.

The objective of the manual is to provide the information you need to implement a microcontroller design using a TMS370 device. The manual contains the following chapters:

Chapter 1 **Introduction to the TMS370 Family Devices.** Discusses the key features and the major components of the TMS370 family devices. Also includes block diagrams for each device category.

Chapter 2 **TMS370 Family Pinouts and Pin Descriptions.** Provides pinouts and pin descriptions for the TMS370 family device categories.

Chapters 3–12 Describe the operation and programming of each major function in the TMS370 architecture.

Chapter 13 **Assembly Language Instruction Set.** Describes the TMS370 addressing modes and each of the 73 instructions, including samples and examples.

Chapter 14 **Design Aids.** Gives sample interface circuits and programming examples.

Chapter 15 **Development Support.** Describes the hardware and software development tools available for the TMS370 devices.

Chapter 16 **Electrical Specifications and Timings.** Gives timing diagrams and electrical specifications for each of the device categories.

Chapter 17 **Customer Information.** Describes mask-ROM prototyping, TMS370 physical characteristics, and parts ordering.

Appendix A **Differences Between a TMS370CxxxA Device and a TMS370Cxxx Device.** Points out the differences between the TMS370CxxxA devices that are described in this manual and the TMS370Cxxx devices.

Appendix B **Peripheral File Memory Map.** Gives reference tables for the TMS370 control bits and registers.

Appendix C **Block Diagrams.** Gives reference block diagrams of the major circuits.

Appendix D **ASCII Character Set.** Lists the ASCII character set that the TMS370 assembler recognizes.

Appendix E **Opcode/Instruction Cross-Reference.** Gives an opcode-to-instruction cross-reference of all 73 mnemonics and 246 opcodes of the TMS370 instruction set.

Appendix F **Instruction/Opcode Cross-Reference and Bus Activity Table.** Gives an instruction-to-opcode cross-reference of all 73 mnemonics and 246 opcodes of the TMS370 instruction set and provides a cycle-by-cycle bus activity table.

Appendix G **Device Pinouts.** Provides pinouts for the individual device categories.

Appendix H **PLCC-to-PGA Pinouts.** Shows the pinouts for the standard PLCC-to-PGA sockets that are commonly used in prototype and production applications. You can use these pinouts when you wirewrap your breadboard with a socket.

Appendix I **PACT.H.** Gives PACT.H macros used with PACT example programs.

Appendix J **Glossary.** Defines acronyms and key terms used in this book.

Style and Symbol Conventions

This document uses the following conventions.

Symbol or Term	Example	Description
(xxxxxx.n)	SPICTL.4	Bit location convention used in text, where 'xxxxxx' is the name of the register containing the bit and 'n' is the bit number (7 = MSB, 0 = LSB).
(xx.n)	4A.0	Bit location convention used in figures, where 'xx' is the hexadecimal address of the peripheral register containing the bit and 'n' is the bit number (7 = MSB, 0 = LSB).
h	1000h	Designates a number in the hexadecimal number system.
P0n	P012	Hexadecimal Peripheral File (PF) address used in instructions accessing the PF. (i.e., P012 = P18)
Pn	P18	Decimal Peripheral File (PF) address used in instructions accessing the PF. (i.e., P18 = P012).
R0n	R010	Hexadecimal Register File (RF) address used in instructions accessing the RF. (i.e., R010 = R16)
Rn	R16	Decimal Register File (RF) address used in instructions accessing the RF. (i.e., R16 = R010)
set		When used in reference to bits, means to write a logic 1 to the bit.
clear		When used in reference to bits, means to write a logic 0 to the bit.
MSbyte		Most significant byte
MSB		Most significant bit
LSbyte		Least significant byte
LSB		Least significant bit

- Program listings, program examples, interactive displays, filenames, and symbol names are shown in a special typeface similar to a typewriter's.

  Here is a sample program listing:

```
LABEL  SUB  R19,B        ;(B) minus (R19) is
                         ;stored in B

       SUB  076h,A       ;(A) minus 076h is stored
                         ;in A

       SUB  R4,R9        ;(R9) minus (R4) is stored
                         ;in R9
```

- In syntax descriptions, the instruction, command, or directive is in a **bold typeface** font, and parameters are in an *italic typeface*. Portions of a syntax that are in **bold** should be entered as shown; portions of a syntax that are in *italics* describe the type of information that should be entered. Here is an example of a directive syntax:

  MOV *s,d*

  MOV is the instruction. This instruction has two parameters, indicated by *s* and *d*.

- Braces ({ and }) indicate a list. The symbol | (read as *or*) separates items within the list. Here's an example of a command that has a list:

  TST {A|B}

  This provides two choices: **TST A** or **TST B**.

  Unless the list is enclosed in square brackets, you must choose one item from the list.

Information About Cautions and Warnings

This book may contain cautions and warnings.

> **This is an example of a caution statement.**
>
> **A caution statement describes a situation that could potentially damage your software or equipment.**

> **This is an example of a warning statement.**
>
> **A warning statement describes a situation that could potentially cause harm to you.**

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

Related Documentation From Texas Instruments

The following books describe the TMS370 family devices 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.

TMS370 Family Assembly Language Tools User's Guide (literature number SPNU010) describes the assembly language tools (assembler, linker, and other tools used to develop assembly code), assembler directives, macros, common object file format, and symbolic debugging directives for the '370 family of devices.

TMS370 Family Optimizing C Compiler User's Guide (literature number SPNU022) describes the '370 C compiler. This C compiler accepts ANSI standard C source code and produces assembly language source code for the '370 family of devices.

TMS370 Family C Source Debugger User's Guide (literature number SPNU028) tells you how to invoke the TMS370 family application board and XDS 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, and includes a tutorial that introduces basic debugger functionality.

TMS370 Family Application Board Technical Reference (literature number SPNU029) describes the components of the application board. Also included is a description of the command monitor and the commands you can use with it.

TMS370 Family Microcontroller Programmers User's Guide (literature number SPNU023) describes how to use and operate the TMS370 family microcontroller programmer and gang programmer.

TMS370Cx10 8-Bit Microcontrollers Data Sheet (literature number SPNS012B) describes the features of the TMS370Cx10 devices and provides pinouts, electrical specifications, and timings for these microprocessors.

TMS370Cx2x 8-Bit Microcontrollers Data Sheet (literature number SPNS018) describes the features of the TMS370Cx2x devices and provides pinouts, electrical specifications, and timings for these microprocessors.

TMS370Cx32 8-Bit Microcontrollers Data Sheet (literature number SPNS015) describes the features of the TMS370Cx32 devices and provides pinouts, electrical specifications, and timings for these microprocessors.

TMS370Cx4x 8-Bit Microcontrollers Data Sheet (literature number SPNS016) describes the features of the TMS370Cx4x devices and provides pinouts, electrical specifications, and timings for these microprocessors.

TMS370Cx5x 8-Bit Microcontrollers Data Sheet (literature number SPNS010B) describes the features of the TMS370Cx5x devices and provides pinouts, electrical specifications, and timings for these microprocessors.

Using the TMS370 A/D Converter Module Application Report (literature number SPNA005) provides examples of hardware interfaces and software routines for the TMS370 analog-to-digital converter module.

Using the TMS370 SPI and SCI Modules Application Report (literature number SPNA006) provides examples of hardware interfaces and software routines for the TMS370 SPI and SCI modules.

Using the TMS370 Timer Modules Application Report (literature number SPNA008) provides examples of hardware interfaces and software routines for the TMS370 timer 1 and timer 2 modules.

Trademarks

CROSSTALK is a trademark of Microstuf, Inc.

Kermit is a registered trademark of Columbia University.

MS-DOS is a registered trademark of Microsoft Corp.

PC-DOS is a trademark of International Business Machines Corp.

PROCOMM is a registered trademark of Datastorm Technologies Inc.

VAX and VMS are trademarks of Digital Equipment Corp.

XDS is a trademark of Texas Instruments Incorporated.

If You Need Assistance. . .

If you want to. . .	Do this. . .
Request more information about Texas Instruments microcontroller products	Write to: Texas Instruments Incorporated Market Communications Manager, MS 736 P.O. Box 1443 Houston, Texas 77251–1443
Order Texas Instruments documentation	Call the TI Literature Response Center: **(800) 477–8924**
Ask questions about product operation or report suspected problems	Call the Microcontroller Hotline: **(713) 274–2370** **FAX: (713) 274–4203**
Report mistakes in this document or any other TI documentation	Fill out and return the reader response card at the end of this book, or send your comments to: Texas Instruments Incorporated Technical Publications Manager, MS 702 P.O. Box 1443 Houston, Texas 77251–1443

Contents

Figures

Tables

Examples

Chapter 1

Introduction to the TMS370 Family Devices

This chapter discusses the key features and major components of the TMS370 family of microcontrollers. The chapter concludes with block diagrams for each device category.

This chapter covers the following topics:

Topic **Page**

1.1 Overview

The TMS370 family consists of VLSI, 8-bit, CMOS microcontrollers with on-chip EEPROM storage and peripheral support functions. These devices offer superior performance in complex, real-time control applications in demanding environments and are available in mask-programmable ROM and EPROM. Moreover, you have a wide range of options to choose from in deciding the most economical and efficient manner for getting a product to market faster.

In an effort to continually improve its products, Texas Instruments has added new, more robust features to the TMS370 family of devices. These features are designed to enhance performance and enable new application technologies. The improved features include new watchdog modes and low-power modes for mask-ROM devices. And, all family members are software compatible, so you can run many existing applications on the improved devices without having to modify your software. (Refer to Appendix A for more information about compatibility.)

In expanding its powerful TMS370 family of microcontrollers, TI offers many new configurable devices for specific applications. As microcontrollers have evolved, TI has added multiple peripheral functions to chips that originally had only a CPU, memory, and I/O blocks. Now, with the high-performance, software-compatible TMS370 microcontrollers, you can choose from over 40 standard products. Or, you can use up to 16 function modules to configure your new device quickly, easily, and cost effectively for your application.

The TMS370 family is fully supported by TI development tools that facilitate simplified software development for quicker market introduction of new products. These tools include an assembler, an optimizing C compiler, a linker, a C source debugger, a design kit, and an EEPROM/EPROM programmer. All of these tools work together using a DOS-based personal computer (PC) as the host and central control element. This allows you to select the host computer and text management as well as editing tools according to your system requirements.

Additionally, the TMS370 in-circuit emulator (XDS—eXtended Development Support) allows you to immediately begin designing, testing, and debugging your system upon specification. The reason for this is straightforward: the emulator itself is modular and configurable, eliminating the need to produce a complete, new emulator for each TMS370 configuration.

Typical Applications

The TMS370 family of devices is the ideal choice for applications like these:

Application Area	Applications	
Automotive	Climate control systems Cruise control Entertainment systems Instrumentation	Navigational systems Engine control Antilock braking
Computer	Keyboards Peripheral interface control	Disk controllers Terminals
Industrial	Motor control Temperature controllers Process control	Meter control Medical instrumentation Security systems
Telecommunications	Modems Intelligent phones Intelligent line card control	Telecopiers Debit cards

Device Categories

The TMS370 family of devices is divided into five categories (see Table 1–1). Each category is supported by development tools that include the in-circuit emulators, a C compiler, an assembler, a linker, and a C source debugger.

Table 1–1. TMS370 Family Categories and Their Corresponding Devices

Category	Devices Included		
TMS370Cx1x	TMS370C010 TMS370C310	TMS370C311 TMS370C610	TMS370C710 SE370C710†
TMS370Cx2x	TMS370C020 TMS370C022 TMS370C320	TMS370C322 TMS370C622	TMS370C722 SE370C722†
TMS370Cx3x	TMS370C032 TMS370C332	TMS370C732 SE370C732†	
TMS370Cx4x	TMS370C040 TMS370C042 TMS370C340	TMS370C342 TMS370C642	TMS370C742 SE370C742†
TMS370Cx5x	TMS370C050 TMS370C150 TMS370C250 TMS370C350 TMS370C052 TMS370C352	TMS370C056 TMS370C156 TMS370C256 TMS370C356 TMS370C756	TMS370C058 TMS370C358 TMS370C758 SE370C756† SE370C758†

† These system evaluators and development tools are used only in a prototype environment. Their reliability has not been characterized.

1.2 Key Features

The TMS370 family of devices has the following key features:

- **Series of compatible devices** that supports software migration
- **CMOS EPROM technology** that provides EPROM for reprogrammable and OTP (one-time programmable) program memory to be used as prototypes and for small-volume or quick-turn production
- **CMOS EEPROM technology** that provides EEPROM programming with a single 5-volt supply
- **A/D technology** that converts analog signals to digital values
- **Static RAM/registers** that offer numerous memory options
- **Flexible operating features:**
  - Power reduction standby and halt modes
  - -40°C to 85°C operating temperature
  - 2-MHz to 20-MHz input clock frequency
  - 5V ±10%
- **Flexible interrupt handling** for design flexibility:
  - Two programmable interrupt levels
  - Programmable rising or falling edge detect
- **System integrity features** that increase flexibility during the software development phase with:
  - Oscillator fault detection
  - Privileged mode lockout
  - Watchdog timer
- **Memory-mapped ports** for easy addressing
- An **optimizing C compiler** that translates ANSI C programs into '370 assembly language source
- A **high-level language debugger** that lets you refine and correct code
- A **modular library** for quick turns to different configurations
- **Fourteen addressing modes** that use eight formats, including:
  - Register-to-register arithmetic
  - Indirect addressing
  - Indexed and indirect branches and calls
- **250-mA typical latch-up immunity at 25°C**
- **ESD protection** that exceeds 2,000 V per MIL-STD-883C method 3015

1.3 Major Components of the TMS370 Architecture

In addition to the features listed in Section 1.2, the TMS370 family members have the following architectural features, according to the device category.

CPU

The TMS370 8-bit CPU has a status register, program counter register, and stack pointer. The CPU uses the register file as working registers, accessed on the internal bus in one bus cycle. The 8-bit internal bus also allows access to memory and the peripheral interfaces. TMS370Cx5x devices allow external memory expansion through ports A, B, C, and D.

Refer to Chapter 3, *CPU and Memory Organization*, for more information about the CPU.

Register File

The register file is located at the beginning of the TMS370 memory map. Register-access instructions in the TMS370 instruction set allow access to any of the first 256 registers (if available) in one bus cycle. This segment of the memory map is used as general-purpose RAM and the stack.

Chapter 3, *CPU and Memory Organization*, provides additional information about the register file.

RAM

RAM modules other than those contained in the register file are mapped after the register file. The TMS370 accesses this RAM in two cycles.

Memory is discussed in full in Chapter 3, *CPU and Memory Organization.*

Data EEPROM

The data EEPROM modules provide in-circuit programmability and data retention in power-off mode. The modules contain 256 or 512 bytes of EEPROM. This memory is useful for constants and infrequently changed variables required by the application program. The EEPROM can be programmed and erased by using available EEPROM programmers or by the TMS370 itself under program control.

The data EEPROM modules are discussed in Chapter 6, *EPROM and EEPROM Modules.*

Program Memory

The program memory provides alternatives to meet the needs of your application. The program memory modules presently contain 2K, 4K, 8K, 16K, or 32K bytes of memory. The program memory in TMS370C7xx, TMS370C6xx, and SE370C7xx devices is EPROM. EPROM can be programmed, erased, and reprogrammed for prototyping (in a ceramic package). EPROM devices that do not have a window (in a plastic package) are one-time programmable (OTP) devices, used for small production runs. In TMS370C0xx and TMS370C3xx devices, the program memory is mask ROM programmed at the factory.

Program memory is discussed in Chapter 6, *EPROM and EEPROM Modules.*

Input/Output Ports

- **TMS370Cx1x** devices have two ports: ports A and D. Port A is an 8-bit wide port, while port D is a 5-bit wide port. Both of these ports can be programmed, bit by bit, to function as either a digital input or a digital output.
- **TMS370Cx2x** devices have four ports: ports A, B, C, and D. Ports A and B are 8-bit wide ports, port C is a 1-bit wide port, and port D is a 5-bit wide port. Each of these ports can be programmed, bit by bit, to function as either a digital input or a digital output.
- **TMS370Cx3x** devices have two ports: ports A and D. Port A is an 8-bit wide port, while port D is a 4-bit wide port. Both of these ports can be programmed, bit by bit, to function as either a digital input or a digital output.
- **TMS370Cx4x** devices have three ports: ports A, B, and D. Port A is an 8-bit wide port, port B is a 3-bit wide port, and port D is a 5-bit wide port. Each of these ports can be programmed, bit by bit, to function as either a digital input or a digital output.
- **TMS370Cx5x** devices have four 8-bit ports: ports A, B, C, and D (port D for the 64-pin devices has only 6 pins). These ports can be configured by the software as the data, control, and address lines for external memory. Any bits not needed for external memory can be programmed to be either a digital input or a digital output.

I/O ports are discussed in greater detail in Chapter 4, *System and Digital I/O Configuration.*

Timer 1 and Timer 2

Timers 1 and 2 are 16-bit timers that can be configured in the following ways:

- Programmable 8-bit prescaler that determines the independent clock sources for the general-purpose timer and the watchdog timer
- 16-bit event timer, to keep a cumulative total of the transitions

- 16-bit pulse accumulator, to measure the pulse input width
- 16-bit input-capture function that latches the counter value on the occurrence of an external input
- Two 16-bit compare registers that trigger when the counter matches the contents of a compare register
- Self-contained PWM (pulse-width modulated) output control function

The results of these operations can generate an interrupt to the CPU, set flag bits, reset the timer counter, toggle an I/O line, or generate PWM outputs. These timers provide up to 200 ns of resolution at 20 MHz.

Timers 1 and 2 are discussed fully in Chapters 7 and 8, respectively.

Watchdog Timer

The watchdog timer helps ensure system integrity. It can be programmed to generate a hardware reset when it times out. This function provides a hardware monitor over the software to avoid losing a program. If not needed as a watchdog, this timer can be used as a general-purpose timer.

For more information about the watchdog timer, refer to Section 7.7, page 7-17.

PACT (Programmable Acquisition and Control Timer)

The PACT module is a programmable timing module that uses some of the on-chip RAM to store its commands as well as the timer values. The module offers the following:

- Input capture on up to six pins, four of which may have a programmable prescaler
- One input capture pin that can drive an 8-bit event counter
- Up to 8 timer-driven outputs
- Timer capability of up to 20 bits
- Interaction between event counter and timer activity
- Eighteen independent interrupt vectors to allow better servicing of events
- Watchdog with selectable time-out period
- Mini-SCI (serial communications interface), which works as a full duplex UART (universal asynchronous receiver transmitter)

Once set up, the PACT requires no CPU overhead, except to service interrupts. The PACT module is discussed in full in Chapter 12, *Programmable Acquisition and Control Timer (PACT).*

SCI (Serial Communications Interface)

The SCI module is a built-in serial interface that offers the following features:

- Programmable to be asynchronous (up to 156K bits/s) or isosynchronous (up to 2.5 Mbits/s)
- Full duplex, double-buffered Rx and Tx
- Programmable format with error-checking capabilities

The SCI module programs and controls all timing, data format, and protocol factors. The CPU takes no part in the serial communications except to write data transmitted to registers in the SCI and to read received data from registers in the SCI when interrupted.

The SCI module is described fully in Chapter 9, *Serial Communications Interface (SCI) Module.*

SPI (Serial Peripheral Interface)

The SPI module is a built-in serial interface that facilitates communication between the network master, slave CPUs, and external peripheral devices. It provides synchronous data transmission up to 2.5 Mbits/s. Like the SCI, the SPI is set up by software. After that, the CPU takes no part in timing, data format, or protocol. Also, like the SCI, the CPU reads and writes to memory-mapped registers to receive and transmit data. An SPI interrupt alerts the CPU when received data is ready.

The SPI module is described fully in Chapter 10, *Serial Peripheral Interface (SPI) Module.*

A/D (Analog-to-Digital) Converter

The 8-bit analog-to-digital converter module performs successive approximation and offers four channels in 40-pin packages and eight channels in 44-pin, 64-pin, and 68-pin packages. The reference source and input channel are selectable. You can program the conversion result to be the ratio of the input voltage to the reference voltage or the ratio of one analog input to another. Input lines that are not required for A/D conversion can be programmed to be digital input lines.

The A/D converter module is described in greater detail in Chapter 11, *Analog-to-Digital Converter Module.*

1.4 Summary of Components by Device

The major components of the TMS370 family devices (discussed in Section 1.3) are summarized in Table 1–2 by device.

Table 1–2. TMS370 Family Architecture Summary

Type	Device	Program Memory (bytes)		Data Memory (bytes)		Off-Chip Memory Expansion (bytes)	Serial Interface Modules†	Timer Modules‡	A/D Channels
		ROM	EPROM	EEPROM	RAM				
ROM	TMS370C010	4K	—	256	128	—	SPI	T1	—
	TMS370C020	4K	—	256	256	—	SPI/SCI	T1	—
	TMS370C040	4K	—	256	256	—	SCI	T1/T2	4/8§
	TMS370C050	4K	—	256	256	112K	SPI/SCI	T1/T2	8
	TMS370C022	8K	—	256	256	—	SPI/SCI	T1	—
	TMS370C032	8K	—	256	256	—	PACT/SCI	PACT	8
	TMS370C042	8K	—	256	256	—	SCI	T1/T2	4/8§
	TMS370C052	8K	—	256	256	112K	SPI/SCI	T1/T2	8
	TMS370C056	16K	—	512	512	112K	SPI/SCI	T1/T2	8
	TMS370C058	32K	—	256	1K	—	SPI/SCI	T1/T2	8
	TMS370C311	2K	—	—	128	—	SPI	T1	—
	TMS370C310	4K	—	—	128	—	SPI	T1	—
	TMS370C320	4K	—	—	256	—	SPI/SCI	T1	—
	TMS370C340	4K	—	—	256	—	SCI	T1/T2	4/8§
	TMS370C350	4K	—	—	256	112K	SPI/SCI	T1/T2	8
	TMS370C322	8K	—	—	256	—	SPI/SCI	T1	—
	TMS370C332	8K	—	—	256	—	PACT/SCI	PACT	8
	TMS370C342	8K	—	—	256	—	SCI	T1/T2	4/8§
	TMS370C352	8K	—	—	256	112K	SPI/SCI	T1/T2	8
	TMS370C356	16K	—	—	512	112K	SPI/SCI	T1/T2	8
	TMS370C358	32K	—	—	1K	—	SPI/SCI	T1/T2	8
ROM-less¶	TMS370C150	—	—	—	256	56K	SPI/SCI	T1/T2	8
	TMS370C250	—	—	256	256	56K	SPI/SCI	T1/T2	8
	TMS370C156	—	—	—	512	56K	SPI/SCI	T1/T2	8
	TMS370C256	—	—	512	512	56K	SPI/SCI	T1/T2	8
OTP#	TMS370C610	—	4K	—	128	—	SPI	T1	—
	TMS370C622	—	8K	—	256	—	SPI/SCI	T1	—
	TMS370C642	—	8K	—	256	—	SCI	T1/T2	4/8§
	TMS370C710	—	4K	256	128	—	SPI	T1	—
	TMS370C722	—	8K	256	256	—	SPI/SCI	T1	—
	TMS370C732	—	8K	256	256	—	PACT/SCI	PACT	8
	TMS370C742	—	8K	256	256	—	SCI	T1/T2	4/8§
	TMS370C756	—	16K	512	512	112K	SPI/SCI	T1/T2	8
	TMS370C758	—	32K	256	1K	—	SPI/SCI	T1/T2	8
SE*	SE370C710	—	4K	256	128	—	SPI	T1	—
	SE370C722	—	8K	256	256	—	SPI/SCI	T1	—
	SE370C732	—	8K	256	256	—	PACT/SCI	PACT	8
	SE370C742	—	8K	256	256	—	SCI	T1/T2	4/8§
	SE370C756	—	16K	512	512	112K	SPI/SCI	T1/T2	8
	SE370C758	—	32K	256	1K	—	SPI/SCI	T1/T2	8

† PACT module has a mini-SCI port.

‡ Timer module includes a watchdog timer that you can program to serve as a general-purpose 16-bit timer. The PACT module includes a watchdog timer.

§ Eight channels for 44-pin device or four channels for 40-pin device.

¶ In ROMless (microprocessor) mode, all address, data, and control lines are fixed as their function.

For OTP (PLCC) availability information, contact your local TI sales office or distributor.

* System evaluator (reprogrammable EPROM)—for use in prototype environment only.

Table 1–2. TMS370 Family Architecture Summary (Continued)

Type	Device	Interrupts/Reset			I/O Pins	No. of Pins/Package◊□
		External	Vectors Total	Sources Total		
	TMS370C010	4	6	13	22	28 DIP/PLCC
	TMS370C020	4	8	16	34	40 DIP/SDIP I 44 PLCC
	TMS370C040	4	9	22	32/36	40 DIP/SDIP I 44 PLCC
	TMS370C050	4	10	23	55/53	68 PLCC I 64 SDIP
	TMS370C022	4	8	16	34	40 DIP/SDIP I 44 PLCC
	TMS370C032	4	23	25	36	44 PLCC
	TMS370C042	4	9	22	32/36	40 DIP/SDIP I 44 PLCC
	TMS370C052	4	10	23	55/53	68 PLCC I 64 SDIP
	TMS370C056	4	10	23	55/53	68 PLCC I 64 SDIP
	TMS370C058	4	10	23	55/53	68 PLCC I 64 SDIP
ROM	TMS370C311	4	6	13	22	28 DIP/PLCC
	TMS370C310	4	6	13	22	28 DIP/PLCC
	TMS370C320	4	8	16	34	40 DIP/SDIP I 44 PLCC
	TMS370C340	4	9	16	32/36	40 DIP/SDIP I 44 PLCC
	TMS370C350	4	10	23	55/53	68 PLCC I 64 SDIP
	TMS370C322	4	8	16	34	40 DIP/SDIP I 44 PLCC
	TMS370C332	4	23	25	36	44 PLCC
	TMS370C342	4	9	22	32/36	40 DIP/SDIP I 44 PLCC
	TMS370C352	4	10	23	55/53	68 PLCC I 64 SDIP
	TMS370C356	4	10	23	55/53	68 PLCC I 64 SDIP
	TMS370C358	4	10	23	55/53	68 PLCC I 64 SDIP
ROM-less¶	TMS370C150	4	10	23	55	68 PLCC
	TMS370C250	4	10	23	55	68 PLCC
	TMS370C156	4	10	23	55	68 PLCC
	TMS370C256	4	10	23	55	68 PLCC
OTP#	TMS370C610	4	6	13	22	28 DIP/PLCC
	TMS370C622	4	8	16	34	40 DIP/SDIP I 44 PLCC
	TMS370C642	4	9	22	32/36	40 DIP/SDIP I 44 PLCC
	TMS370C710	4	6	13	22	28 DIP/PLCC
	TMS370C722	4	8	16	34	40 DIP/SDIP I 44 PLCC
	TMS370C732	4	23	25	36	44 PLCC
	TMS370C742	4	9	22	32/36	40 DIP/SDIP I 44 PLCC
	TMS370C756	4	10	23	55/53	68 PLCC I 64 SDIP
	TMS370C758	4	10	23	55/53	68 PLCC I 64 SDIP
SE*	SE370C710	4	6	13	22	28 CDIP/CLCC
	SE370C722	4	8	16	34	40 CSDIP/CDIP I 44 CLCC
	SE370C732	4	23	25	36	44 CLCC
	SE370C742	4	9	22	32/36	40 CSDIP/CDIP I 44 CLCC
	SE370C756	4	10	23	55/53	68 CLCC I 64 CSDIP
	SE370C758	4	10	23	55/53	68 CLCC I 64 CSDIP

¶ In ROM-less (microprocessor) mode, all address, data, and control lines are fixed as their function.
# For OTP (PLCC) availability information, contact your local TI sales office or distributor.
* System evaluator (reprogrammable EPROM)—for use in prototype environment only.
□ Refer to Table 17–1, page 17-6 for package type acronyms.
◊ 64-pin devices do not support off-chip memory expansion.

1.5 Device Block Diagrams

The TMS370 family is based on a register-to-register architecture, which allows access to a register file (up to 256 bytes) in a single bus cycle. On-chip memory includes program memory (mask ROM or EPROM), static RAM, and data EEPROM.

The versatile on-chip peripheral functions include (depending on the specific member of the series) an analog-to-digital converter (A/D), a serial communications interface (SCI), a serial peripheral interface (SPI), three different timer modules, and up to 55 digital input/output pins.

Figure 1–1 is a block diagram of the TMS370Cx1x devices, showing the major functional blocks.

Figure 1–1. TMS370Cx1x Block Diagram

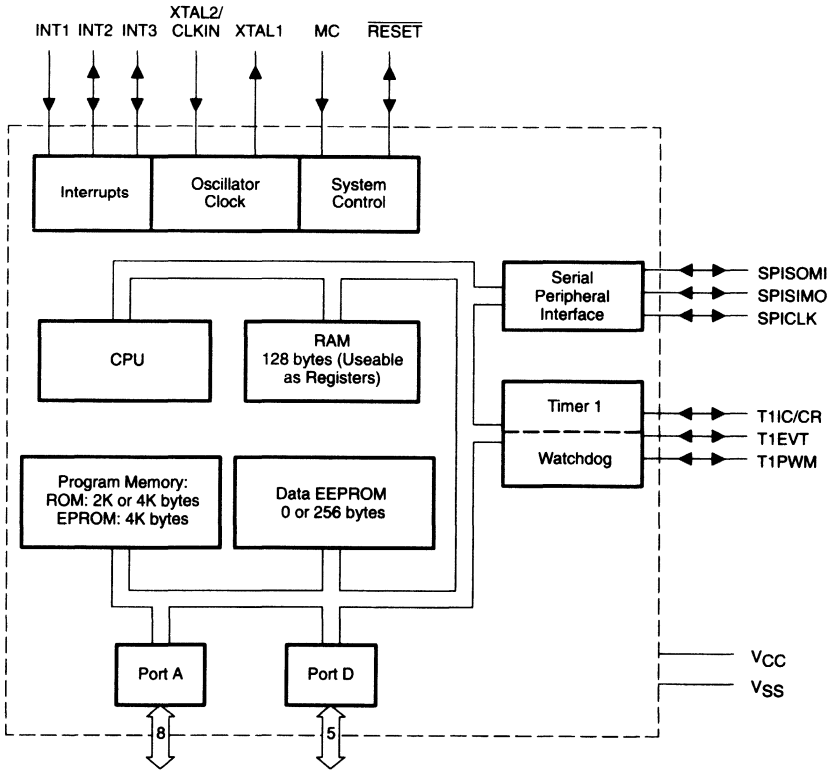


Figure 1–2 is a block diagram of the TMS370Cx2x devices, showing the major functional blocks.

Figure 1–2. TMS370Cx2x Block Diagram

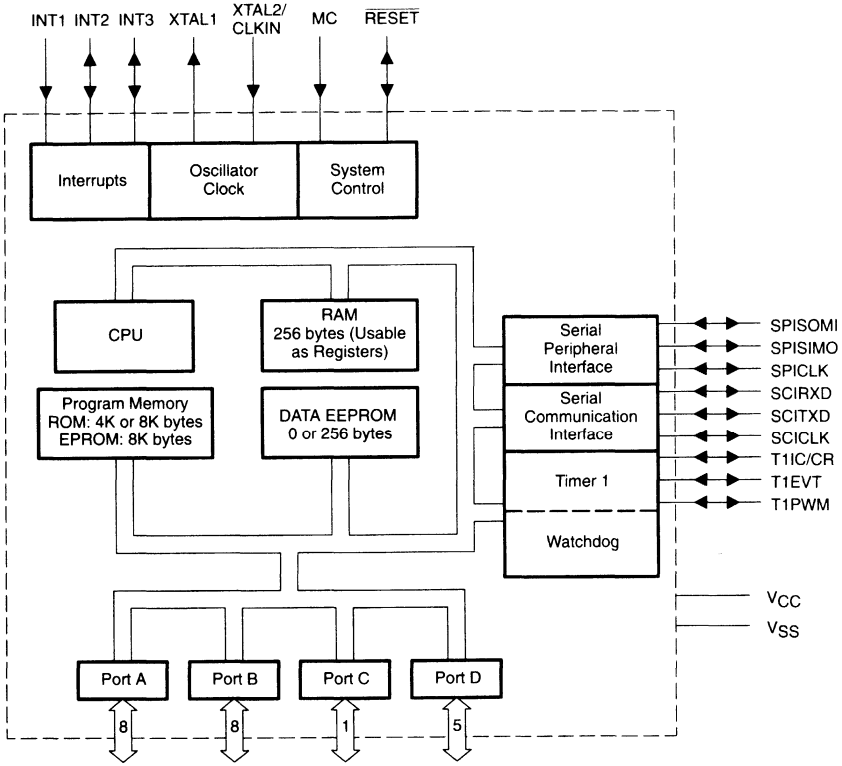
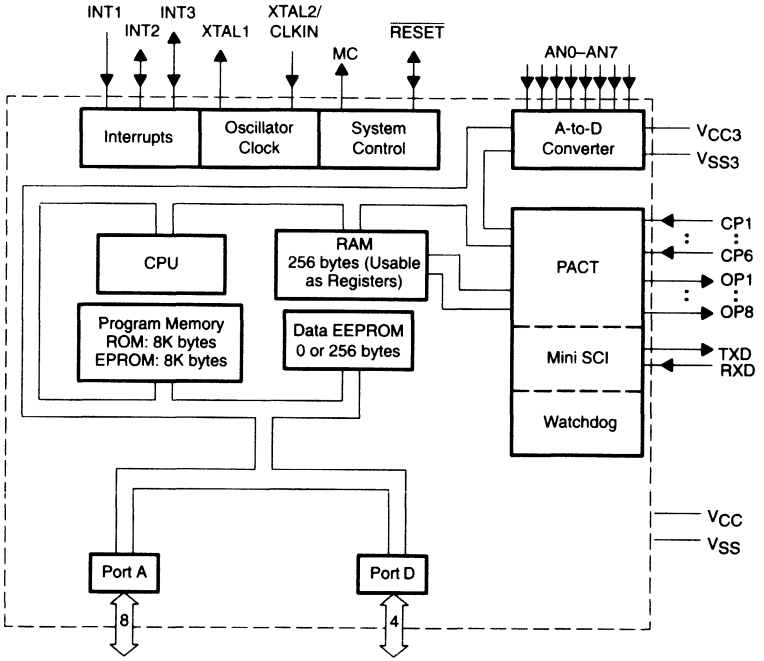


Figure 1–3 is a block diagram of the TMS370Cx3x devices, showing the major functional blocks.

Figure 1–3. TMS370Cx3x Block Diagram



Note: Three of port D's four I/O buffers (D4, D6, and D7) are internally connected to three of the PACT module's inputs (CP3, CP4, and CP5). This gives the actual pins of D4/CP3, D6/CP4, and D7/CP5.

Figure 1–4 is a block diagram of the TMS370Cx4x devices, showing the major functional blocks.

Figure 1–4. TMS370Cx4x Block Diagram

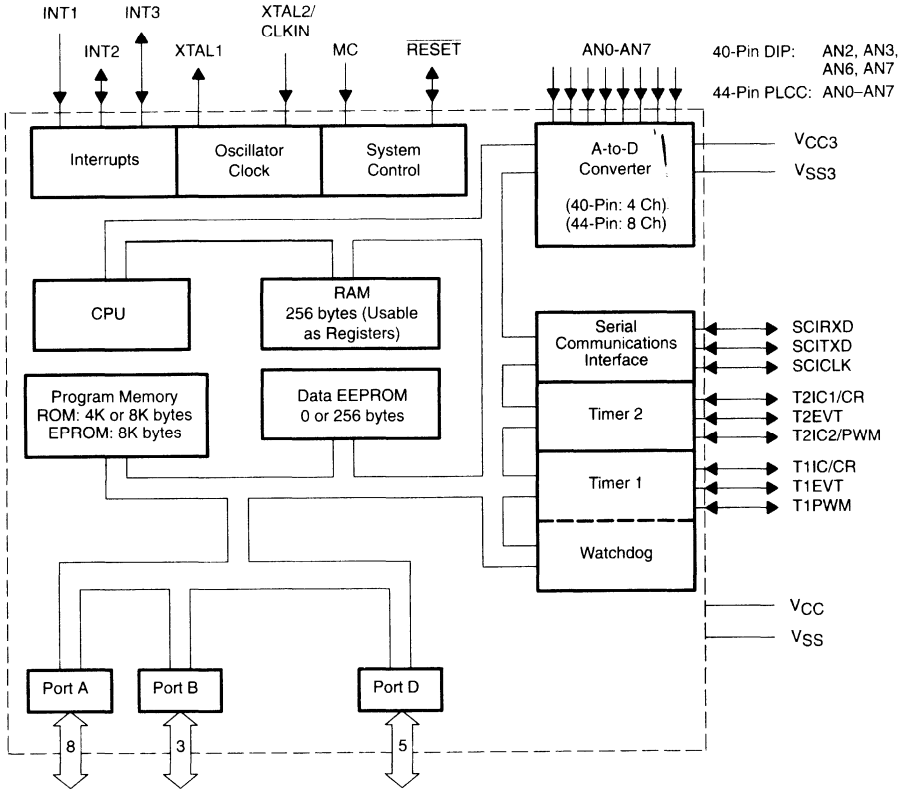
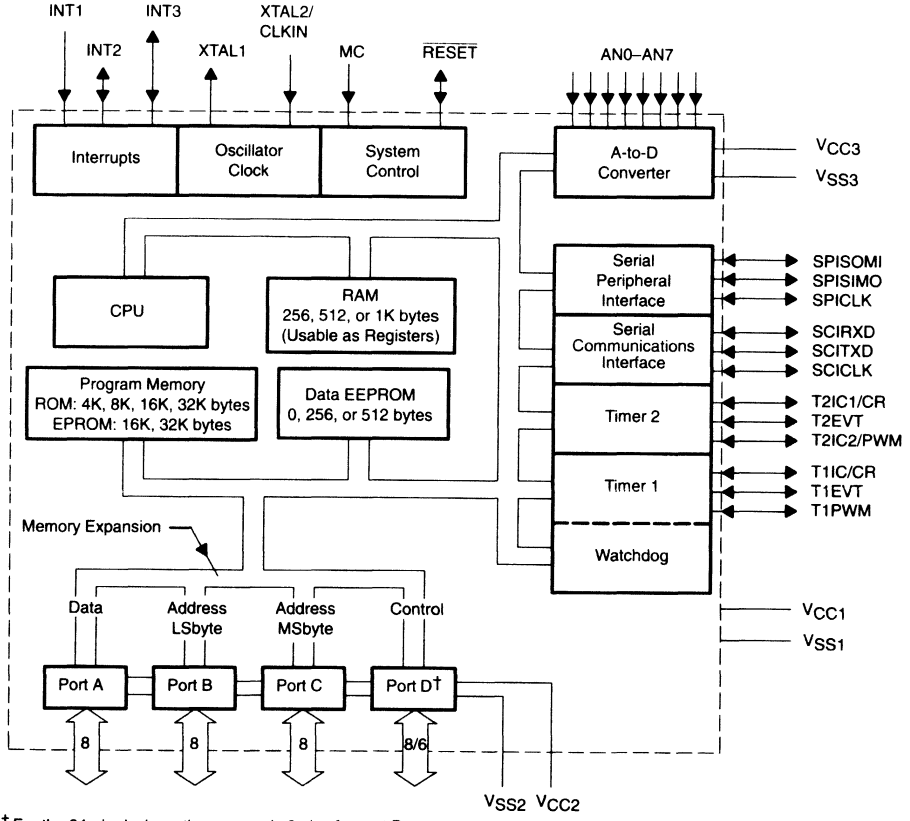


Figure 1–5 is a block diagram of the TMS370Cx5x devices, showing the major functional blocks.

Figure 1–5. TMS370Cx5x Block Diagram



† For the 64-pin devices, there are only 6 pins for port D.

Chapter 2

TMS370 Family Pinouts and Pin Descriptions

This chapter provides pinouts and pin descriptions for the individual device categories.

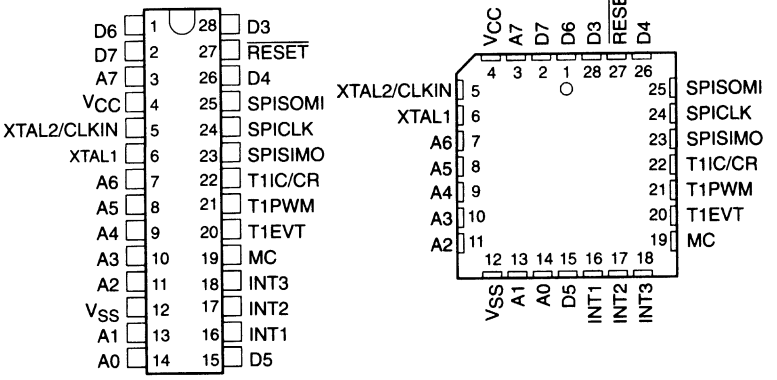
Topic **Page**

2.1 TMS370Cx1x Pinouts and Pin Descriptions

2

The pinouts and pin descriptions for the TMS370Cx1x devices are shown in Figure 2–1 and in Table 2–1, respectively.

Figure 2–1. Pinouts for TMS370Cx1x



A. 28-Pin PDIP (N)
B. 28-Pin CDIP (JD)

C. 28-Pin PLCC (FN)
D. 28-Pin CLCC (FZ)

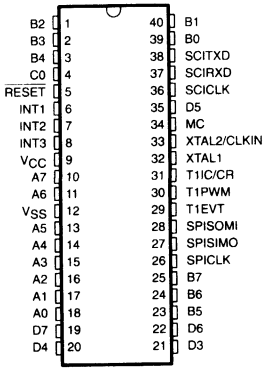
Table 2–1. TMS370Cx1x Pin Descriptions

Pin		I/O	Description
Name	**No.**		
A0	14	I/O	Port A is a general-purpose bidirectional I/O port
A1	13	I/O	
A2	11	I/O	
A3	10	I/O	
A4	9	I/O	
A5	8	I/O	
A6	7	I/O	
A7	3	I/O	
D3	28	I/O	Port D is a general-purpose bidirectional I/O port;
D4	26	I/O	D3 is also configurable as CLKOUT
D5	15	I/O	
D6	1	I/O	
D7	2	I/O	
INT1	16	I	External (nonmaskable or maskable) interrupt/general-purpose input pin
INT2	17	I/O	External maskable interrupt input/general-purpose bidirectional pin
INT3	18	I/O	External maskable interrupt input/general-purpose bidirectional pin
T1IC/CR	22	I/O	Timer 1 input capture/counter reset input pin/general-purpose bidirectional pin
T1PWM	21	I/O	Timer 1 PWM output pin/general-purpose bidirectional pin
T1EVT	20	I/O	Timer 1 external event input pin/general-purpose bidirectional pin
SPISOMI	25	I/O	SPI slave output pin, master input pin/general-purpose bidirectional pin
SPISIMO	23	I/O	SPI slave input pin, master output pin/general-purpose bidirectional pin
SPICLK	24	I/O	SPI bidirectional serial clock pin/general-purpose bidirectional pin
$\overline{\text{RESET}}$	27	I/O	System reset bidirectional pin; as an input, it initializes the microcontroller; as an open-drain output, it indicates that an internal failure was detected by the watchdog or oscillator fault circuit
MC	19	I	Mode control input pin; enables EEPROM write protection override (WPO) mode
XTAL2/CLKIN	5	I	Internal oscillator crystal input/external clock source input
XTAL1	6	O	Internal oscillator output for crystal
V_{CC}	4		Positive supply voltage
V_{SS}	12		Ground reference

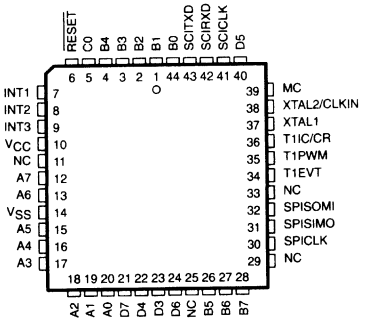
2.2 TMS370Cx2x Pinouts and Pin Descriptions

The pinouts and pin descriptions for the TMS370Cx2x devices are shown in Figure 2–2 and Table 2–2, respectively.

Figure 2–2. Pinouts for TMS370Cx2x



A. 40-Pin PDIP (N)
B. 40-Pin CDIP (JD)
C. 40-Pin SDIP (N2)
D. 40-Pin CSDIP (JC)



E. 44-Pin PLCC (FN)
F. 44-Pin CLCC (FZ)

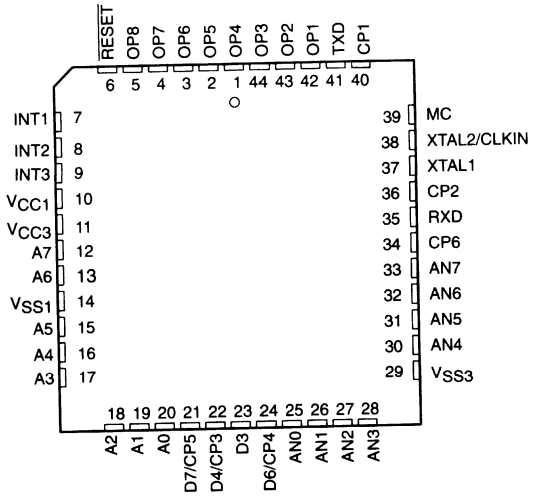
Table 2–2. TMS370Cx2x Pin Descriptions

Pin				
	No. DIP / PLCC			
Name	40	44	I/O	Description
A0	18	20	I/O	Port A is a general-purpose bidirectional I/O port
A1	17	19	I/O	
A2	16	18	I/O	
A3	15	17	I/O	
A4	14	16	I/O	
A5	13	15	I/O	
A6	11	13	I/O	
A7	10	12	I/O	
B0	39	44	I/O	Port B is a general-purpose bidirectional I/O port
B1	40	1	I/O	
B2	1	2	I/O	
B3	2	3	I/O	
B4	3	4	I/O	
B5	23	26	I/O	
B6	24	27	I/O	
B7	25	28	I/O	
C0	4	5	I/O	Port C is a general-purpose bidirectional I/O port
D3	21	23	I/O	Port D is a general-purpose bidirectional I/O port;
D4	20	22	I/O	D3 is also configurable as a CLKOUT
D5	35	40	I/O	
D6	22	24	I/O	
D7	19	21	I/O	
SCITXD	38	43	I/O	SCI transmit data output pin/general-purpose bidirectional pin
SCIRXD	37	42	I/O	SCI receive data input pin/general-purpose bidirectional pin
SCICLK	36	41	I/O	SCI bidirectional serial clock pin/general-purpose bidirectional pin
INT1	6	7	I	External (nonmaskable or maskable) interrupt/general-purpose input pin
INT2	7	8	I/O	External maskable interrupt input/general-purpose bidirectional pin
INT3	8	9	I/O	External maskable interrupt input/general-purpose bidirectional pin
T1IC/CR	31	36	I/O	Timer 1 input capture/counter reset input pin/general-purpose bidirectional pin
T1PWM	30	35	I/O	Timer 1 PWM output pin/general-purpose bidirectional pin
T1EVT	29	34	I/O	Timer 1 external event input pin/general-purpose bidirectional pin
SPISOMI	28	32	I/O	SPI slave output pin, master input pin/general-purpose bidirectional pin
SPISIMO	27	31	I/O	SPI slave input pin, master output pin/general-purpose bidirectional pin
SPICLK	26	30	I/O	SPI bidirectional serial clock pin/general-purpose bidirectional pin
$\overline{\text{RESET}}$	5	6	I/O	System reset bidirectional pin; as an input, it initializes the microcontroller; as an open-drain output, it indicates an internal failure was detected by the watchdog or oscillator fault circuit
MC	34	39	I	Mode control input pin; enables the EEPROM write-protection override (WPO) mode
XTAL1	32	37	O	Internal oscillator output for crystal
XTAL2/CLKIN	33	38	I	Internal oscillator crystal input/external clock source input
NC	–	11		No connections
	–	25		
	–	29		
	–	33		
V_{CC}	9	10		Positive supply voltage
V_{SS}	12	14		Ground reference

2.3 TMS370Cx3x Pinout and Pin Description

The pinout and pin descriptions for the TMS370Cx3x devices are shown in Figure 2–3 and Table 2–3, respectively.

Figure 2–3. Pinout for TMS370Cx3x



A. 44-Pin PLCC (FN)
B. 44-Pin CLCC (FZ)

Table 2–3. TMS370Cx3x Pin Descriptions

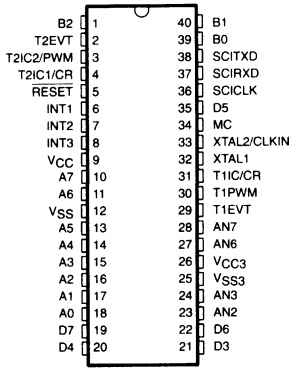
Pin			
Name	**No.**	**I/O**	**Description**
A0	20	I/O	Port A is a general-purpose bidirectional port
A1	19	I/O	
A2	18	I/O	
A3	17	I/O	
A4	16	I/O	
A5	15	I/O	
A6	13	I/O	
A7	12	I/O	
D3	23	I/O	Port D is a general-purpose bidirectional port
D4/CP3	22	I/O	I/O pin: Also configurable as CLKOUT
D6/CP4	24	I/O	I/O pin: PACT input capture 3.†
D7/CP5	21	I/O	I/O pin: PACT input capture 4.† I/O pin: PACT input capture 5.†
INT1	7	I	External (nonmaskable or maskable) interrupt/general-purpose input pin
INT2	8	I/O	External maskable interrupt input/general-purpose bidirectional pin
INT3	9	I/O	External maskable interrupt input/general-purpose bidirectional pin
CP1	40	I	PACT input capture pin 1
CP2	36	I	PACT input capture pin 2
CP6	34	I	PACT input capture pin 6; external event input pin (for event counter)
TXD	41	O	PACT SCI transmit output pin
RXD	35	I	PACT SCI receive input pin
OP1	42	O	PACT output pin 1
OP2	43	O	PACT output pin 2
OP3	44	O	PACT output pin 3
OP4	1	O	PACT output pin 4
OP5	2	O	PACT output pin 5
OP6	3	O	PACT output pin 6
OP7	4	O	PACT output pin 7
OP8	5	O	PACT output pin 8
AN0	25	I	A/D analog input (AN0–AN7) or positive reference pins (AN1–AN7)
AN1	26	I	
AN2	27	I	
AN3	28	I	
AN4	30	I	The analog port can be individually programmed as general-purpose input pins if it is not used as A/D converter analog input or positive reference input
AN5	31	I	
AN6	32	I	
AN7	33	I	
$\overline{\text{RESET}}$	6	I/O	System reset bidirectional pin; as input, it initializes the microcontroller; as an open-drain output, it indicates that an internal failure was detected by the watchdog or oscillator fault circuit
MC	39	I	Mode control input pin; enables EEPROM write-protection override (WPO) mode
XTAL2/CLKIN	38	I	Internal oscillator crystal input/external clock source input
XTAL1	37	O	Internal oscillator output for crystal
V_{CC1}	10		Positive supply voltage for digital logic and digital I/O pins
V_{SS1}	14		Ground reference for digital logic and digital I/O pins
V_{CC3}	11		A/D converter positive supply voltage and optional positive reference input
V_{SS3}	29		A/D converter ground supply and low reference input pin

† Some of port D's digital I/O buffers are internally connected to some of the PACT module's input capture pins. This allows the microcontroller to read the level on the input capture pin or, if the port D pin is configured as an output, to generate a capture. Be careful to leave the port D pin configured as an input if the corresponding input capture pin is being driven by external circuitry.

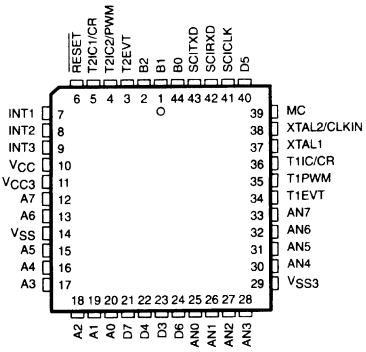
2.4 TMS370Cx4x Pinouts and Pin Descriptions

The pinouts and pin descriptions for the TMS370Cx4x devices are shown in Figure 2–4 and Table 2–4, respectively.

Figure 2–4. Pinouts for TMS370Cx4x



A. 40-Pin PDIP (N)
B. 40-Pin CDIP (JD)
C. 40-Pin SDIP (N2)
D. 40-Pin CSDIP (JC)



E. 44-Pin PLCC (FN)
F. 44-Pin CLCC (FZ)

Table 2–4. TMS370Cx4x Pin Descriptions

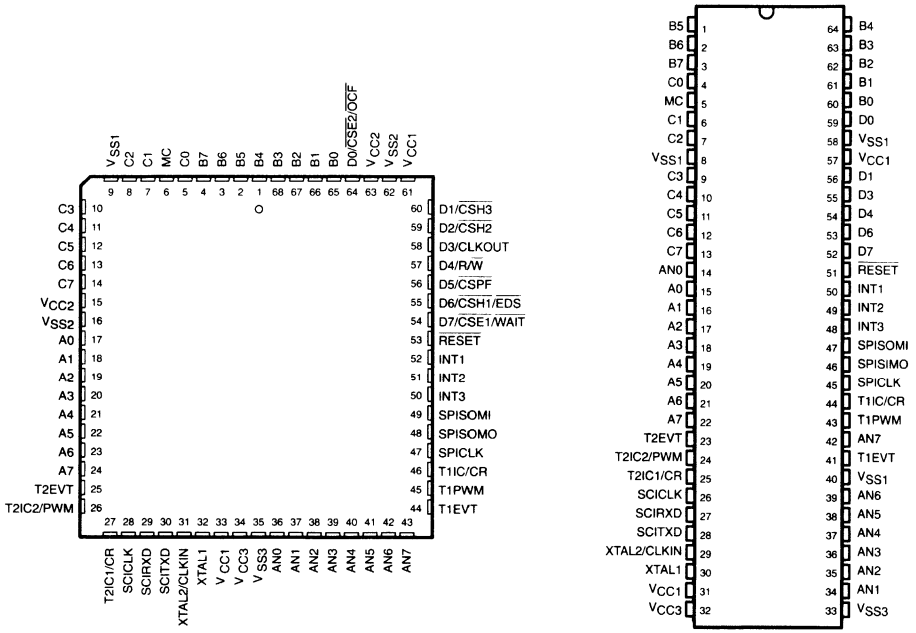
Pin				
Name	DIP (40)	PLCC (44)	I/O	Description
A0	18	20	I/O	Port A is a general-purpose bidirectional I/O port
A1	17	19	I/O	
A2	16	18	I/O	
A3	15	17	I/O	
A4	14	16	I/O	
A5	13	15	I/O	
A6	11	13	I/O	
A7	10	12	I/O	
B0	39	44	I/O	Port B is a general-purpose bidirectional I/O port
B1	40	1	I/O	
B2	1	2	I/O	
D3	21	23	I/O	Port D is a general-purpose bidirectional I/O port; D3 is also configurable as a CLKOUT
D4	20	22	I/O	
D5	35	40	I/O	
D6	22	24	I/O	
D7	19	21	I/O	
AN0/E0	—	25	I	A/D analog input channels or positive reference pins; any A/D channel can be programmed as general-purpose input pins (E port) if not used as an analog input or reference channel
AN1/E1	—	26	I	
AN2/E2	23	27	I	
AN3/E3	24	28	I	
AN4/E4	—	30	I	
AN5/E5	—	31	I	
AN6/E6	27	32	I	
AN7/E7	28	33	I	
V_{CC3}	26	11		A/D converter positive supply voltage and optional positive reference input pin
V_{SS3}	25	29		A/D converter ground supply and low reference input pin
INT1	6	7	I	External (nonmaskable or maskable) interrupt/general-purpose input pin
INT2	7	8	I/O	External maskable interrupt input/general-purpose bidirectional pin
INT3	8	9	I/O	External maskable interrupt input/general-purpose bidirectional pin
T1IC/CR	31	36	I/O	Timer 1 input capture/counter reset input pin/general-purpose bidirectional pin
T1PWM	30	35	I/O	Timer 1 PWM output pin/general-purpose bidirectional pin
T1EVT	29	34	I/O	Timer 1 external event input pin/general-purpose bidirectional pin
T2IC1/CR	4	5	I/O	Timer 2 input capture/counter reset input pin/general-purpose bidirectional pin
T2IC2/PWM	3	4	I/O	Timer 2 input capture 2/PWM output pin/general-purpose bidirectional pin
T2EVT	2	3	I/O	Timer 2 external event Input pin/general-purpose bidirectional pin
SCITXD	38	43	I/O	SCI transmit data output pin/general-purpose bidirectional pin
SCIRXD	37	42	I/O	SCI receive data input pin/general-purpose bidirectional pin
SCICLK	36	41	I/O	SCI bidirectional serial clock pin/general-purpose bidirectional pin
$\overline{\text{RESET}}$	5	6	I/O	System reset bidirectional pin; as input, it initializes the microcontroller; as an open-drain output, it indicates that an internal failure was detected by the watchdog or oscillator fault circuit
MC	34	39	I	Mode control input pin; enables the EEPROM write protection override (WPO) mode
XTAL1	32	37	I	Internal oscillator output for crystal
XTAL2/CLKIN	33	38	O	Internal oscillator crystal input/external clock source input
V_{CC}	9	10		Positive supply voltage
V_{SS}	12	14		Ground reference

2.5 TMS370Cx5x Pinouts and Pin Descriptions

2

The pinout and pin descriptions for the TMS370Cx5x devices are shown in Figure 2–5 and Table 2–5, respectively.

Figure 2–5. Pinouts for TMS370Cx5x



A. 68-Pin PLCC (FN)
B. 68-Pin CLCC (FZ)

C. 64-Pin SDIP (NM)
D. 64-Pin CSDIP (JN)

Table 2–5. TMS370Cx5x Pin Descriptions

Pin					
Name	Alternate Function†	DIP (64)	PLCC (68)	I/O	Description
A0	DATA0 (LSB)	15	17	I/O	Single-chip mode: port A is a general-purpose bidirectional port Expansion mode: port A can be individually programmed as the external bidirectional data bus (DATA0–DATA7)
A1	DATA1	16	18	I/O	
A2	DATA2	17	19	I/O	
A3	DATA3	18	20	I/O	
A4	DATA4	19	21	I/O	
A5	DATA5	20	22	I/O	
A6	DATA6	21	23	I/O	
A7	DATA7 (MSB)	22	24	I/O	
B0	ADD0	60	65	I/O	Single-chip mode: port B is a general-purpose bidirectional I/O port Expansion modes: port B can be individually programmed as the low-order address output bus (ADD0–ADD7)
B1	ADD1	61	66	I/O	
B2	ADD2	62	67	I/O	
B3	ADD3	63	68	I/O	
B4	ADD4	64	1	I/O	
B5	ADD5	1	2	I/O	
B6	ADD6	2	3	I/O	
B7	ADD7	3	4	I/O	
C0	ADD8	4	5	I/O	Single-chip mode: port C is a general-purpose bidirectional I/O port Expansion mode: port C can be individually programmed as the high-order address output bus (ADD8–ADD15)
C1	ADD9	6	7	I/O	
C2	ADD10	7	8	I/O	
C3	ADD11	9	10	I/O	
C4	ADD12	10	11	I/O	
C5	ADD13	11	12	I/O	
C6	ADD14	12	13	I/O	
C7	ADD15	13	14	I/O	
INT1	INTIN	50	52	I	External (nonmaskable or maskable) interrupt/general-purpose input pin
INT2	INTIO1	49	51	I/O	External maskable interrupt input/general-purpose bidirectional pin
INT3	INTIO2	48	50	I/O	External maskable interrupt input/general-purpose bidirectional pin

† For '370Cx58 devices, there is no memory bus expansion; ports A, B, C, and D can be configured only as general-purpose I/O pins.

2

Table 2–5. TMS370Cx5x Pin Descriptions (Continued)

Pin					I/O	Description
Name	Alternate Function†		DIP (64)	PLCC (68)		
	Function A	Function B				Single-chip mode: port D is a general-purpose bidirectional I/O port. Each of the port D pins can be individually configured as a general-purpose I/O pin, primary memory control signal (function A), or secondary memory control signal (function B). All chip selects are independent and can be used for memory bank switching.
D0	$\overline{CSE2}$	$\overline{OCF}$	59	64	I/O	I/O pin/A: chip select eighth output 2 goes low during memory accesses to 2000h–3FFFh /B: Opcode fetch goes low during the opcode fetch memory cycle
D1	$\overline{CSH3}$		56	60	I/O	I/O pin/A: chip select half output 3 goes low during memory accesses to 8000h–FFFFh
D2	$\overline{CSH2}$		—	59	I/O	I/O pin/A: chip select half output 2 goes low during memory accesses to 8000h–FFFFh
D3	CLKOUT	CLKOUT	55	58	I/O	I/O pin/A, B: internal clock signal is 1/4 XTAL2/CLKIN frequency
D4	R/$\overline{W}$	R/$\overline{W}$	54	57	I/O	I/O pin/A, B: read/write output pin
D5	$\overline{CSPF}$		—	56	I/O	I/O pin/A: chip select peripheral output for peripheral file goes low during memory accesses to 10C0h–10FFh
D6	$\overline{CSH1}$	$\overline{EDS}$	53	55	I/O	I/O pin/A: chip select half output 1 goes low during memory accesses to 8000h–FFFFh. I/O pin/B: external data strobe output goes low during memory accesses from external memory and has the same timings as the five chip selects
D7	$\overline{CSE1}$	$\overline{WAIT}$	52	54	I/O	I/O pin/A: chip select eighth output goes low during memory accesses to 2000h–3FFFh. I/O pin/B: wait input pin extends bus signals
T1IC/CR	T1IO1		44	46	I/O	Timer 1 input capture/counter reset input pin/general-purpose bidirectional pin
T1PWM	T1IO2		43	45	I/O	Timer 1 PWM output pin/general-purpose bidirectional pin
T1EVT	T1IO3		41	44	I/O	Timer 1 external event input pin/general-purpose bidirectional pin
T2IC1/CR	T2IO1		25	27	I/O	Timer 2 input capture 1/counter reset input pin/general-purpose bidirectional pin
T2IC2/PWM	T2IO2		24	26	I/O	Timer 2 input capture 2/PWM output pin/general-purpose bidirectional pin
T2EVT	T2IO3		23	25	I/O	Timer 2 external event input pin/general-purpose bidirectional pin
SPISOMI	SPIIO1		47	49	I/O	SPI slave output pin, master input pin/general-purpose bidirectional pin
SPISIMO	SPIIO2		46	48	I/O	SPI slave input pin, master output pin/general-purpose bidirectional pin
SPICLK	SPIIO3		45	47	I/O	SPI bidirectional serial clock pin/general-purpose bidirectional pin

† For '370Cx58 devices, there is no memory bus expansion; ports A, B, C, and D can be configured only as general-purpose I/O pins.

Table 2–5. TMS370Cx5x Pin Descriptions (Concluded)

Pin					
Name	**Alternate Function**	**DIP (64)**	**PLCC (68)**	**I/O**	**Description**
SCITXD	SCIIO1	28	30	I/O	SCI transmit data output pin/general-purpose bidirectional pin
SCIRXD	SCIIO2	27	29	I/O	SCI receive data input pin/general-purpose bidirectional pin
SCICLK	SCIIO3	26	28	I/O	SCI bidirectional serial clock pin/general-purpose bidirectional pin
AN0	E0	14	36	I	A/D analog input (AN0–AN7) or positive reference pins (AN1–AN7)
AN1	E1	34	37	I	
AN2	E2	35	38	I	
AN3	E3	36	39	I	Port E can be individually programmed as general-purpose input pins if not used as A/D converter analog input or positive reference input
AN4	E4	37	40	I	
AN5	E5	38	41	I	
AN6	E6	39	42	I	
AN7	E7	42	43	I	
V_{CC3}		32	34		A/D converter positive supply voltage and optional positive reference input pin
V_{SS3}		33	35		A/D converter ground supply and low reference input pin
$\overline{\text{RESET}}$		51	53	I/O	System reset bidirectional pin; as an input, it initializes microcontroller; as open-drain output, it indicates an internal failure was detected by the watchdog or oscillator fault circuit
MC		5	6	I	Mode control pin; enables EEPROM write protection override (WPO) mode
XTAL2/CLKIN		29	31	I	Internal oscillator crystal input/external clock source input
XTAL1		30	32	O	Internal oscillator output for crystal
V_{CC1}		31, 57	33, 61		Positive supply voltage
V_{CC2}		—	15, 63		Positive supply voltage
V_{SS1}		8, 58	9		Ground reference for digital logic
V_{SS2}		—	16, 62		Ground reference for digital I/O pins

Chapter 3

CPU and Memory Organization

This chapter describes the CPU registers and memory organization. In the TMS370 register-to-register architecture, the CPU and up to the first 256 bytes of RAM act as a single unit along with the program counter, stack pointer, and status register.

This chapter covers the following topics:

Topic **Page**

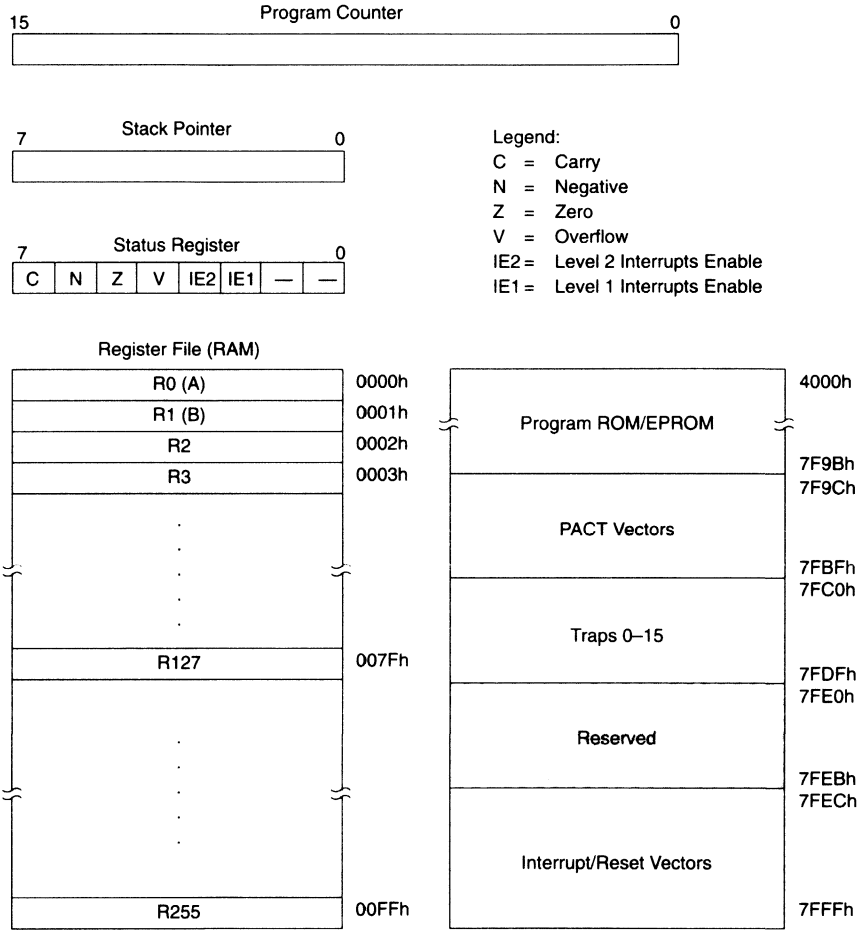
3.1 CPU/Register File Interaction

The TMS370 architecture provides the following components:

- CPU registers:
  - A **stack pointer**, which points to the last entry in the memory stack,
  - A **status register**, which monitors the operation of the instructions and contains the global interrupt bits, and
  - A **program counter**, which points to the memory location of the next instruction to be executed.
- A memory map that includes:
  - A **register file** that can be accessed as general-purpose registers, data memory storage, program instructions, or part of the stack,
  - A **peripheral file** that provides access to all internal peripheral modules, system-wide control functions, and EEPROM/EPROM programming control,
  - **Data EEPROM modules**, which provide in-circuit programmability and data retention in power-off mode, and
  - **Program memory** that provides alternatives to meet the needs of your application.

Figure 3–1 illustrates the CPU registers and memory blocks.

Figure 3–1. Programmer's Model



3.2 CPU Registers

The CPU contains three registers to control the status and direction of the program. These are the stack pointer, status register, and program counter. These registers and their use are described in the following subsections.

3.2.1 Stack Pointer (SP)

The stack operates as a last-in, first-out, read/write memory. The stack is typically used to store the return address on subroutine calls and the status register contents during interrupts.

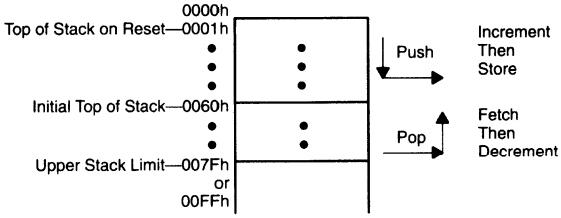
The stack pointer (SP) is an 8-bit CPU register that points to the last entry or top of the stack. The SP is automatically incremented *before* data is pushed onto the stack and decremented *after* data is popped from the stack.

The stack can be placed anywhere in the register file. During reset, the SP is loaded with 01h. To control the area occupied by the stack, the application program must set the stack pointer and include code to monitor the stack size.

The SP is loaded from register B (R1) by the assembly language instruction LDSP (load stack pointer). The LDSP instruction allows the stack to be located anywhere in the register file space. The SP can be read into register B by the STSP (store stack pointer) command. Figure 3–2 illustrates an example SP initialization and stack operation.

```
INIT   MOV #60h,B          ;Load register B with the value
                           ;60h.
       LDSP                ;Load the stack pointer with the
                           ;contents of register B.
```

Figure 3–2. Stack Example



For devices with 256 (or more) bytes of RAM, if the stack is pushed beyond its limit of 00FFh, the SP register wraps around from 00FFh to 0000h without an error indication. For devices with only 128 bytes of RAM, the stack is not implemented beyond 007Fh; data pushed beyond this limit is lost. Your application program must guard against stack overflow.

3.2.2 Status Register (ST)

The status register (ST) monitors the operation of the instructions and contains the global interrupt enable bits. The ST register includes four status bits (condition flags) and two interrupt enable bits:

- The four status bits indicate the outcome of the previous instruction; conditional instructions (for example, the conditional jump instructions) use these status bits to determine program flow.
- The two interrupt bits control the two interrupt levels.

The ST register, status bit notation, and status bit definitions are as follows:

Status Register (ST)

Bit #	7	6	5	4	3	2	1	0
P010	C	N	Z	V	IE2	IE1	—	—
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0		

R = Read, W = Write, –*n* = Value of the bit after the register is reset

Bits 0–1 **Reserved.** Read data is indeterminate.

Bit 2 **IE1**. Level 1 Interrupt Enable.

This bit controls interrupt level 1 (highest priority).

0 = Disables interrupt requests from priority level 1.
1 = Enables interrupt requests from priority level 1.

Bit 3 **IE2**. Interrupt Enable, Chain 2.

This bit controls interrupt level 2 (lowest priority).

0 = Disables interrupt requests from priority level 2.
1 = Enables interrupt requests from priority level 2.

Bit 4 **V**. Overflow.

This bit is set by the CPU if a signed arithmetic overflow condition was detected during the previous instruction. The value of this flag is significant at the completion of the following instructions: ADC, ADD, INC, INCW, CMP, DEC, SUB, SBB, and DIV.

Instruction	V = 1 If
ADC, ADD, INC, INCW	(C XOR N) AND (Bit 7{s} XNOR Bit 7{d})
CMP, DEC, SUB, SBB	(C XOR N) AND (Bit 7{s} XOR Bit 7{d})
DIV (Rs, A)	Rs ≤ A, which means quotient > 255

Bit 5 **Z**. Zero.

This bit is set by the CPU if the result of the previous operation was 0; cleared otherwise.

Bit 6 **N**. Negative.

The CPU sets this bit to the value of the most significant bit (sign bit) of the result of the previous operation.

Bit 7 **C**. Carry.

This status bit is set by arithmetic instructions as a carry bit or as a no-borrow bit. It is also affected by the rotate instructions. See each instruction in Chapter 13 for a detailed description of how the carry bit is used.

When the CPU acknowledges an interrupt, the contents of the status register are automatically pushed onto the stack; then the status register is cleared (for more information about interrupt effects on the status register, see subsection 5.1.1, page 5-2). The RTI instruction implements a normal exit from an interrupt service routine. When the CPU executes the RTI instruction, it automatically restores the contents of the status register with a stack-pop operation.

The four condition flags (C, N, Z, and V) are updated every time an instruction is executed that manipulates or moves data. As a result, conditional branches should be performed immediately after a data manipulation operation. The instructions that *do not* affect the contents of these flags are:

TRAP 0 through TRAP 15	IDLE
CALL	NOP
CALLR	PUSH ST
BR	RTS
DJNZ	STSP
JMP	JMPL
Conditional jump instructions	LDSP

The LDST instruction allows a program to change all bits in the status register. The byte following this instruction is loaded directly into the status register. The assembly language instructions DINT, EINT, EINTH, and EINTL enable specific interrupts. These instructions are converted to an LDST #iop8 opcode by the assembler so that #iop8 is the appropriate value to set or clear the specific interrupt (see Chapter 13 for more information on the LDST instruction).

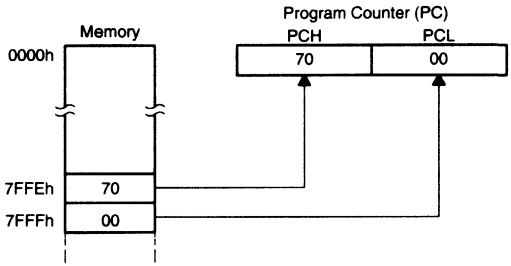
The carry (C) bit can be set with the SETC opcode and cleared with the CLRC opcode.

3.2.3 Program Counter (PC)

The contents of the program counter (PC) point to the memory location of the next instruction to be executed. The PC consists of two 8-bit registers in the CPU: the program counter high (PCH) and program counter low (PCL). These registers contain the MSbyte and LSbyte of a 16-bit address.

During reset, the PCH (MSbyte of the PC) is loaded with the contents of memory location 7FFEh, and the PCL (LSbyte of the PC) is loaded with the contents of memory location 7FFFh. Figure 3–3 illustrates this operation using an example value of 7000h as the contents of memory locations 7FFEh and 7FFFh (reset vector).

Figure 3–3. Program Counter After Reset

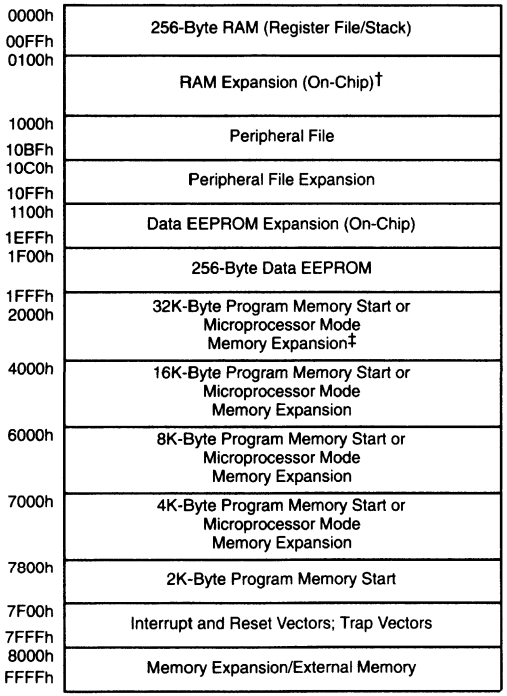


3.3 Memory Map

Figure 3–4 shows the memory map of the TMS370 family members. The partitioning of memory and physical location of memory (that is, on- or off-chip) depends on the device used and the memory mode of operation. The memory modes of operation are discussed in Section 3.4.

Each device that has memory expansion can be programmed to use up to sixteen address bits. This allows access of up to 56K bytes of memory. In addition, memory expansion features allow up to 112K bytes of external memory. (The expansion features are described further in subsection 3.4.2.) †

Figure 3–4. TMS370 Memory Map



† In devices with more than 256 bytes of RAM, only the first 256-byte block can be used as registers/stack.

‡ In devices that have 32K bytes of program memory, it begins at 2000h and ends at 9FFFh.

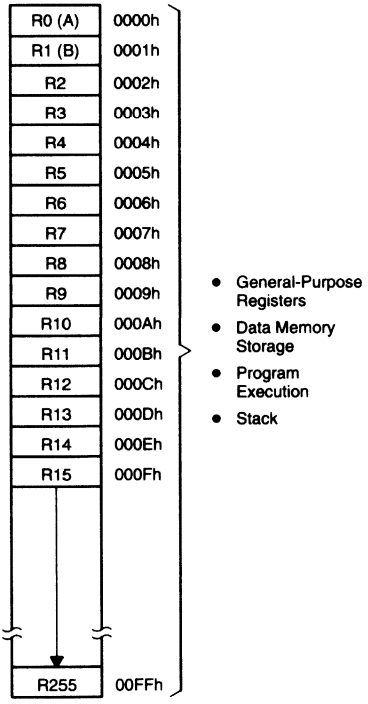
3.3.1 Register File

The beginning addresses of the memory map (0000h–00FFh) are on-chip RAM, called the *register file* (RF).

- In devices with 128 bytes of RAM, the RF has 128 memory locations treated as registers R0 through R127.
- In devices with 256 bytes of RAM, the RF has 256 memory locations treated as registers R0 through R255.
- If the device incorporates the PACT module with 128 bytes of dual-port RAM, then the dual-port RAM is mapped into memory locations 0080h–00FFh. Any of this RAM not used by the PACT module can be used as registers or stack.

The memory addresses of the registers in the RF are shown in Figure 3–5.

Figure 3–5. Register File Addresses



The first two registers, R0 and R1, are also called registers A and B, respectively. Some instructions imply registers A or B; for example, the instruction LDSP assumes that the value to be loaded into the stack pointer is contained in register B.

Locations within the RF address space can serve as either the CPU register file or general-purpose read/write memory. Program instructions can reside in and be executed from any location in the address space without restriction. The stack also occupies a portion of the RF.

The multiple use of the RF gives you the flexibility to use the register file however you wish. The partitioning of the RF is determined by the value loaded into the stack pointer and by the program's use of the RF.

Any location in the RF can be accessed in one of three ways:

- Register access using the register number. For example,

```
MOV     A,R6        ;Move the contents of Register A to
                    ;Register R6.
MOV     R12,R200    ;Move the contents of Register 12 to
                    ;Register R200.
```

- Stack access using the stack pointer. For example,

```
MOV     #5,B        ;Move the value 5 into Register B.
        LDSP        ;Move the contents of Register B to
                    ;the stack pointer.
        PUSH A      ;Increment stack pointer to 6.
                    ;Move contents of Register A to 0006h.
```

- Normal memory access using 16-bit addresses. For example,

```
MOV     A,0006      ;Move the contents of Register A to
                    ;memory location 0006h.
```

When working with the RF, you must keep the following in mind:

- **Access time.** When the RF is used as a general-purpose register, the access time is a single system clock cycle. Access to the RF for any other purpose takes two clock cycles.

- **Reset operations.** A reset operation has no effect on the contents of any memory location within the RF except for locations 0000h (register A) and 0001h (register B). Registers A and B are cleared in the beginning of the reset process.

- **Halt, idle, and standby states.** The halt, idle, and standby states have no effect on the contents of the RF or RAM.

- **RAM outside of the RF.** RAM that is not within the first 256 bytes (0000h–00FFh) is general-purpose RAM and is not considered part of the RF. Access to this RAM will take two clock cycles.

3.3.2 Peripheral File

The peripheral file (PF) is a set of memory-mapped registers that provide access to all internal peripheral modules, system-wide control functions, and EEPROM/EPROM programming control.

The PF includes 256 addresses in the memory map from 1000h–10FFh. The PF is divided into sixteen frames of 16 bytes each. Each peripheral frame is allocated its own set of control registers. In addition, some frames are dedicated to specific functions.

The instruction set includes some instructions which access the peripheral file directly. These instructions designate the register by the number of the file register relative to 1000h, preceded by P0 for a hexadecimal designator or P for a decimal designator. For example, the system configuration control register 0 is located at address 1010h; its peripheral file hexadecimal designator is P010, and its decimal designator is P16.

Table 3–1 shows the address map for the peripheral file.

Table 3–1. Peripheral File Address Map

Frame No.	Address	Description	TMS370C				
			x1x	x2x	x3x	x4x	x5x
0	1000h	Reserved for Factory Test	—	—	—	—	—
1	1010h	System and EEPROM / EPROM Control Registers	Yes	Yes	Yes	Yes	Yes
2	1020h	Digital I/O Port Control Registers	Yes	Yes	Yes	Yes	Yes
3	1030h	SPI Registers	Yes	Yes	NA	NA	Yes
4	1040h	Timer 1 Registers	Yes	Yes	NA	Yes	Yes
		PACT Registers	NA	NA	Yes	NA	NA
5	1050h	SCI Registers	NA	Yes	NA	Yes	Yes
6	1060h	Timer 2 Registers	NA	NA	NA	Yes	Yes
7	1070h	A-to-D Registers	NA	NA	Yes	Yes	Yes
8	1080h	Reserved	NA	NA	NA	NA	NA
9	1090h	Reserved	NA	NA	NA	NA	NA
10	10A0h	Reserved	NA	NA	NA	NA	NA
11	10B0h	Reserved	NA	NA	NA	NA	NA
12	10C0h	External Peripheral Control	NA	NA	NA	NA	Yes†
13	10D0h	External Peripheral Control	NA	NA	NA	NA	Yes†
14	10E0h	External Peripheral Control	NA	NA	NA	NA	Yes†
15	10F0h	External Peripheral Control	NA	NA	NA	NA	Yes†

NA = Not Available
† Does not apply to TMS370Cx58 devices or 64-pin devices.

Here is additional information on the peripheral file frames shown in Table 3–1:

- Frame 0 of the peripheral file (memory addresses 1000h–100Fh) is reserved for factory testing. The results of access to this frame are unpredictable.

- Frame 1 (1010h–101Fh) contains system configuration and control functions. It also contains registers for controlling EEPROM/EPROM programming. EEPROM/EPROM module control registers are described in Chapter 6, *EPROM and EEPROM Modules*.

- Frame 2 (1020h–102Fh) contains the digital I/O pin configuration/control registers. The individual functions controlled by these registers are described in Section 4.4, page 4-16.

- Frames 3 through 7 are used by the internal peripherals. These peripherals and their control registers are described in the following chapters:
  - Timer 1 registers — Chapter 7
  - Timer 2 registers — Chapter 8
  - SCI registers — Chapter 9
  - SPI registers — Chapter 10
  - A-to-D registers — Chapter 11
  - PACT registers — Chapter 12

- Frames 8 through 11 are reserved.

- Frames 12 through 15 are available for external expansion of the peripheral file on devices that have memory expansion capability. These frames are located in external memory and accessed by the external address and data buses.

3.3.3 Data EEPROM Modules

The data EEPROM modules are 256- and 512-byte arrays. The 256-byte array is located at memory addresses 1F00h through 1FFFh, with the WPR (write protection register) at 1F00h. The 512-byte array is located at memory addresses 1E00h through 1FFFh, with WPRs at 1E00h and 1F00h. Larger arrays will continue to grow toward the smaller memory addresses with WPRs located in the first byte of every 256-byte boundary.

Each set of 256 bytes is configured into eight blocks of 32 bytes and has an associated WPR. Each block can be individually write protected by setting the appropriate bit in the WPR. This module can be programmed on an entire array, byte-wide, or single-bit basis. The read-access time for the EEPROM module is two system clock cycles.

Programming of the data EEPROM array is controlled by the data EEPROM control register (DEECTL) at memory address 101Ah (P01A) and the corresponding WPRs. See subsections 6.2.1 and 6.2.2 for more details on the WPR and DEECTL registers.

3.3.4 Program Memory

The program memory options available in the TMS370 family allow a wide selection of memory types: ROM or EPROM, ranging in size from 2K to 32K bytes. The program memory is arranged as individually addressable bytes in the memory map. Data can be read or code can be executed directly from these locations.

Memory addresses 7F9Ch through 7FBFh and 7FECh through 7FFFh are reserved for interrupt and reset vectors. Trap vectors, used with TRAP0 through TRAP15 instructions, are at addresses 7FC0h through 7FDFh. Table 3–2 gives the memory map for the reserved vector locations and describes the differences among TMS370 family members.

Table 3–2. Vector Address Map

Address	Description	TMS370C					No. of K bytes
		x1x	x2x	x3x	x4x	x5x	
7F9Ch	PACT INT 1–18	NA	NA	Yes	NA	NA	36
7FC0h	Trap 0–15	Yes	Yes	Yes	Yes	Yes	32
7FE0h	Reserved	Yes	Yes	Yes	Yes	Yes	12
7FECh	A/D Converter	NA	NA	Yes	Yes	Yes	2
7FEEh	Timer 2	NA	NA	NA	Yes	Yes	2
7FF0h	Serial Communications Interface TX	NA	Yes	NA	Yes	Yes	2
7FF2h	Serial Communications Interface RX	NA	Yes	NA	Yes	Yes	2
7FF4h	Timer 1	Yes	Yes	NA	Yes	Yes	2
7FF6h	Serial Peripheral Interface	Yes	Yes	NA	NA	Yes	2
7FF8h	Interrupt 3	Yes	Yes	Yes	Yes	Yes	2
7FFAh	Interrupt 2	Yes	Yes	Yes	Yes	Yes	2
7FFCh	Interrupt 1	Yes	Yes	Yes	Yes	Yes	2
7FFEh	Reset	Yes	Yes	Yes	Yes	Yes	2

NA = Not Available

3.3.4.1 Program ROM Module (TMS370C0xx and TMS370C3xx Devices Only)

The program ROM module consists of read-only memory, which is programmed at the time of device fabrication. The present ROM module sizes are 2K, 4K, 8K, 16K, and 32K. All accesses to the ROM module require two system clock cycles.

Note:

All TMS370 family devices contain mask-ROM space reserved for TI use only. This space includes locations 7FE0h through 7FEBh. This reserved area should not be used in your software algorithm, nor should it be used during mask-ROM/firmware development.

The contents of the reserved locations are changed by TI only.

3.3.4.2 ROM-less Devices (TMS370C1xx and TMS370C2xx Devices Only)

The program memory for ROM-less devices must be off-chip. The TMS370 must be in the microprocessor mode to operate.

3.3.4.3 Program EPROM Modules (TMS370C6xx and TMS370C7xx Devices Only)

The program EPROM modules replace the program ROM for systems in prototype or small production runs. The modules presently consist of 4K, 8K, 16K, or 32K bytes of EPROM and the necessary programming control logic.

Read access to the program EPROM is performed as normal memory read cycles. Write cycles require a special sequence of events. See subsection 6.4.2, page 6-12, for a detailed discussion of programming the EPROM modules.

The EPROM can be written to only when V_{PP} is applied to the MC pin and the VPPS bit (EPCTL.6) is set. When V_{PP} is applied to the MC pin, all on-chip EEPROM is in write protect override (WPO) mode, regardless of the state of the VPPS bit. This allows the EPROM to be protected while the EEPROM is in WPO.

3.4 Memory Operating Modes

Devices that have the memory expansion can operate in one of two major memory modes.

- ❑ Microcomputer modes (μC)
  - ■ Microcomputer single-chip mode
  - ■ Microcomputer with external expansion
- ❑ Microprocessor modes (μP)
  - ■ Microprocessor without internal program memory
  - ■ Microprocessor with internal program memory

Devices that do not have the memory expansion can operate only in the microcomputer single-chip mode. Table 3–3 shows the presently available devices and the modes that they can operate in.

Table 3–3. Memory Modes Available

Mode	Device TMS370C					
	x1x	x2x	x3x	x4x	15x, 25x	05x, 35x, 75x
μC Single Chip	Yes	Yes	Yes	Yes	No	Yes
μC External Expansion	No	No	No	No	No	Yes†
μP Without Internal Program Memory	No	No	No	No	Yes	Yes†
μP With Internal Program Memory	No	No	No	No	No	Yes†

† Does not apply to TMS370Cx58 devices.

For devices that have the memory expansion, the basic microcomputer and microprocessor operating modes are selected by the voltage level applied to the dedicated MC pin when the $\overline{\text{RESET}}$ pin goes inactive (high).

- ❑ If the MC pin is low when the $\overline{\text{RESET}}$ signal goes high, then the processor enters the microcomputer mode.
- ❑ If the MC pin is high when the $\overline{\text{RESET}}$ signal goes high, then it enters the microprocessor mode.

Changing the MC pin alone will not change the memory mode. To change memory operating mode, change the MC pin and then reset the device.

Applying 12 volts to the MC pin *after* reset forces the device to enter the WPO mode.

Note:

If 12 volts are applied to the MC pin when the $\overline{\text{RESET}}$ pin goes from low to high, the results are unpredictable.

If the processor resets into a microcomputer mode, the software can change the internal system configuration registers to select the desired memory expansion configuration. Part of this configuration set-up involves digital I/O port D. Each pin of port D can be programmed to serve one of three purposes: digital I/O, function A signal, or function B signal. Function A includes chip-select signals, which can be used in the microcomputer mode with external memory expansion. Function B includes signals used in either the microcomputer or the microprocessor modes to access external memory chips.

Each of the memory operating modes is described in the following subsections.

3.4.1 Microcomputer Single-Chip Mode

In the microcomputer single-chip mode, a TMS370 device functions as a self-contained microcomputer with all memory and peripherals on the chip. This mode has no external address or data memory and allows more pins (used for the external buses in other modes) to be programmed as input/output pins. The single-chip mode maximizes the general-purpose I/O capability for real-time control applications. Figure 3–6 on the following page shows a memory map for the microcomputer single-chip mode.

During reset, the MC pin must remain at a low level in order to successfully enter the microcomputer mode. While the device is operating in the single-chip mode, external circuitry can place 12 volts on the MC pin to enter the WPO mode to alter protected EEPROM.

To put a TMS370 device into the microcomputer single-chip mode:

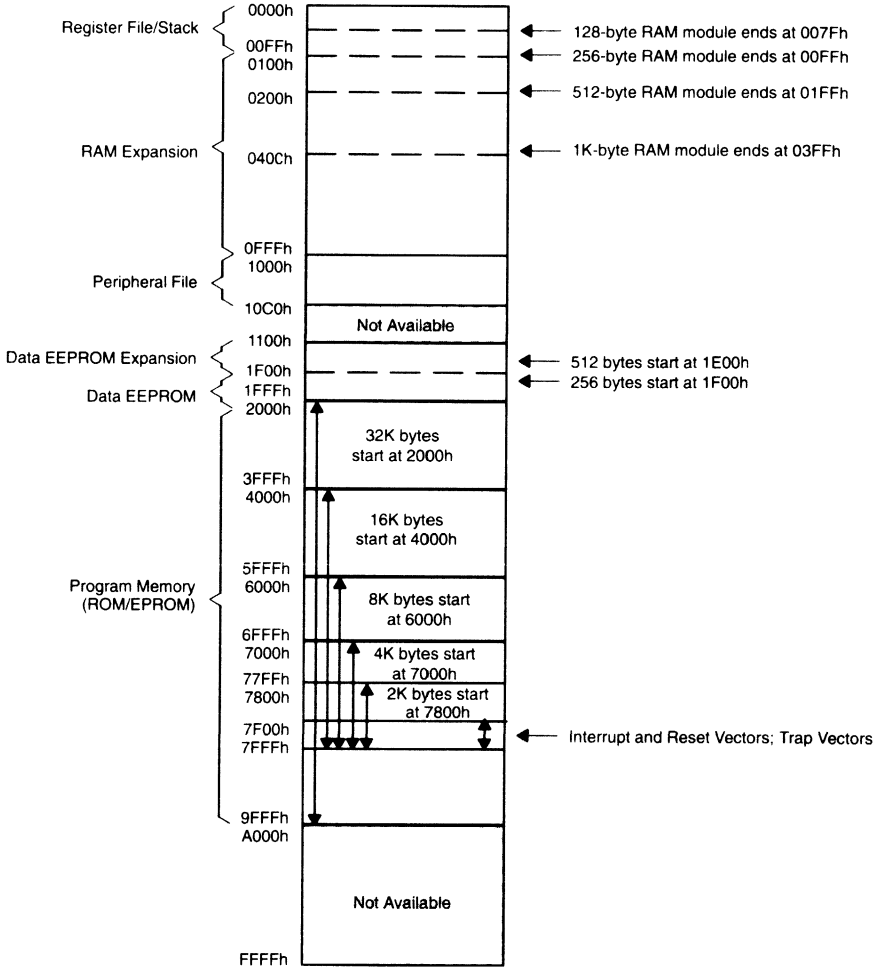
1) Place a low logic level on the MC pin.

2) Take the $\overline{\text{RESET}}$ pin active low, then return $\overline{\text{RESET}}$ to its inactive high state.

Note:

The preceding procedure must be followed for devices that do not have the memory expansion, even though they operate only in the microcomputer single-chip mode.

Figure 3–6. Microcomputer Single-Chip Mode



3.4.2 Microcomputer Mode With External Expansion (All Devices With Memory Expansion and Internal Program Memory)

The microcomputer mode also supports memory expansion to external memory or peripherals, while all on-chip memory (register file, ROM, and EPROM) remains active. Digital I/O ports, under the control of their associated port control registers, become external memory as follows:

- Port A: 8-bit data memory
- Port B and C: 16-bit address memory
- Port D: 8-bit control memory

Each pin that is not used for address, data, or control memory can be individually programmed as a general-purpose input/output pin. These bits are programmed by setting the digital I/O control registers in the peripheral file (see Section 4.4, page 4-16, for further information on programming I/O pins).

The address memory and data memory are nonmultiplexed, eliminating the requirement for an external address/data latch and thereby lowering system cost. External interface decode logic can be reduced further by using the precoded chip-select outputs.

The port D outputs can be programmed on a pin-by-pin basis to provide direct memory/peripheral chip-select or chip-enable functions. Each port D pin can be individually set to function A, function B or general-purpose I/O.

Function A

When port D is set up to drive the chip-selection signals (function A), memory accesses to certain ranges of memory activate pins (refer to Figure 3–7 on the following page):

- A memory access to any location between 2000h and 3FFFh activates pins $\overline{\text{CSE1}}$ and $\overline{\text{CSE2}}$. Typically, an application that uses both $\overline{\text{CSE1}}$ and $\overline{\text{CSE2}}$ sets one as the active chip-select function and sets the other as a general-purpose high-level output. Up to 16K bytes of external memory can be mapped into this address space.

- A memory access to any location between 8000h and FFFFh activates the $\overline{\text{CSH1}}$, $\overline{\text{CSH2}}$, and $\overline{\text{CSH3}}$ pins if they are enabled by the appropriate port control registers. The $\overline{\text{CSH1}}$, $\overline{\text{CSH2}}$, and $\overline{\text{CSH3}}$ signals can be used as memory bank select signals under software control. As a result, up to 96K bytes of external memory can be mapped into the 32K-byte logical address space of 8000h–FFFFh.

- A memory access to the upper four frames (memory addresses 10C0h–10FFh) of the peripheral file activates the $\overline{\text{CSPF}}$ pin if the pin is enabled. This signal can be used as a chip select for external expansion of the peripheral file. These chip-select control lines allow access to more than 112K bytes of external memory.

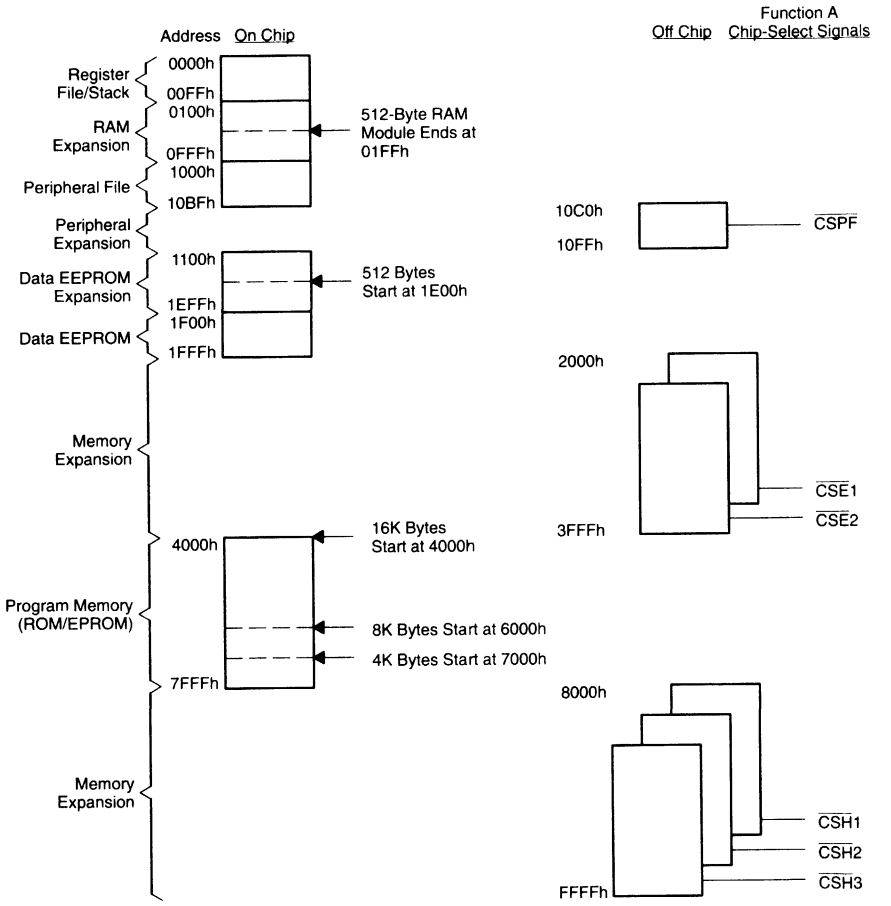
See Section 4.4 for a description of the digital I/O port control registers and how the chip-select signals are enabled.

Note that the RAM expansion area, data EEPROM expansion area, and internal memory expansion area are not available for external accesses.

Note:

Applications that use more than one chip-select signal for the same address should set the unused chip selects (i.e., chip selects not currently used to select memory banks) to general-purpose high-level outputs.

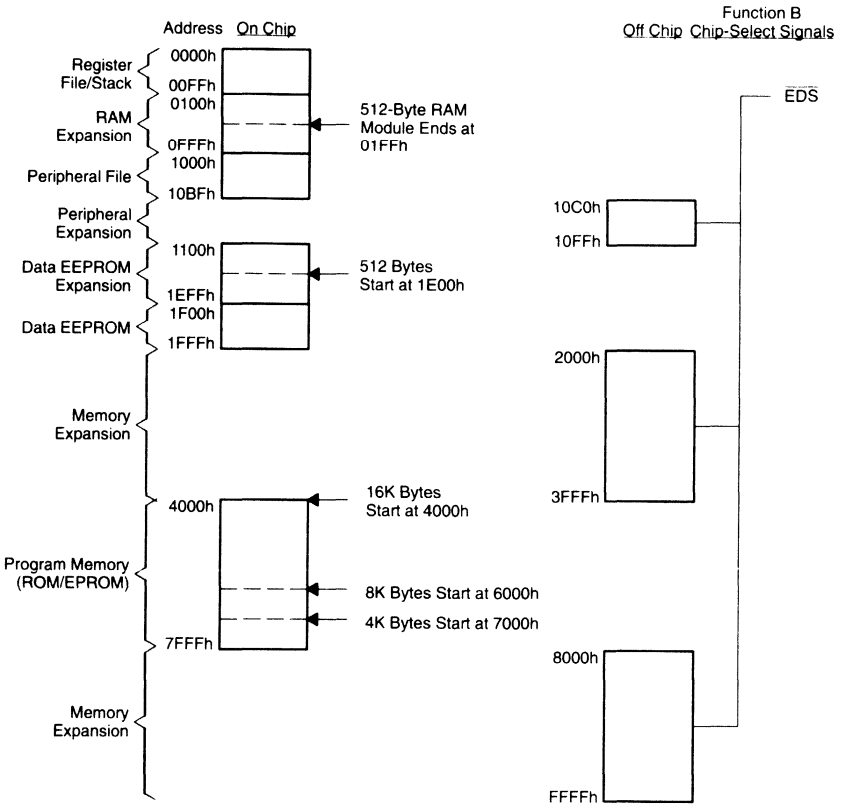
Figure 3–7. Microcomputer Mode With Function A Expansion



Function B

All predecoded chip selects have the same timing as the external data strobe ($\overline{EDS}$) signal (see Chapter 16, *Electrical Specifications*). $\overline{EDS}$ is a function B (microprocessor mode) signal that goes low whenever an access to external memory is made. Figure 3–8 shows a memory map for the microcomputer mode with function B expansion.

Figure 3–8. Microcomputer Mode With Function B Expansion



See Section 4.4 for a description of the digital I/O port control registers and how the chip-select signals are enabled.

To put a device into the microcomputer mode with external expansion (the device *must* have bus expansion), execute the following steps:

1) Place a low logic level on the MC pin.
2) Take the $\overline{\text{RESET}}$ pin active low, then return $\overline{\text{RESET}}$ to its inactive high state.
3) Program the digital I/O registers to select the chip-select or control signals needed (function A or function B).

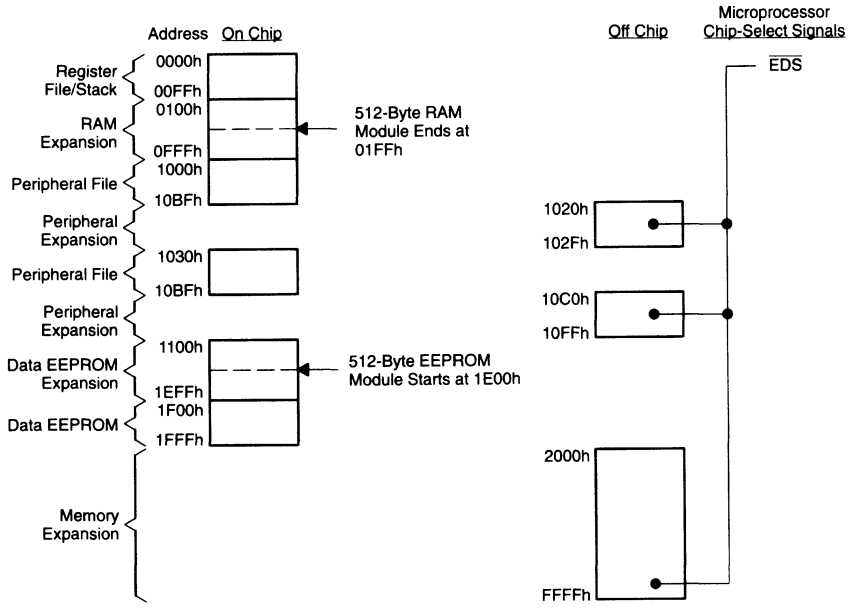
3.4.3 Microprocessor Mode Without Internal Memory (Memory Expansion Devices Only)

When a device is activated in the microprocessor mode, the register file and data EEPROM remain active, but the on-chip program ROM or EPROM is disabled. The $\overline{EDS}$ signal goes low when a memory access is made to addresses 1020–102F, 10C0h–10FFh, and 2000h–FFFFh. The program area, the reset vector, interrupt vectors, and trap vectors must be located in off-chip memory locations.

When a device is reset into the microprocessor mode, the digital I/O port D registers are set to function B expansion memory control signals. The chip-select signals are not available in function B. Ports B and C are set up as the external address bus, and port A is set up to be the external data bus. Software cannot change the digital I/O configuration.

Figure 3–9 shows a memory map for the microprocessor mode.

Figure 3–9. Microprocessor Mode Without Internal Memory



To put a device into the microprocessor mode without internal memory:

1) Place a high logic level on the MC pin.

2) Take the $\overline{\text{RESET}}$ pin active low, then return $\overline{\text{RESET}}$ to its inactive high state.

3.4.4 Microprocessor Mode With Internal Program Memory (Memory Expansion Devices Only)

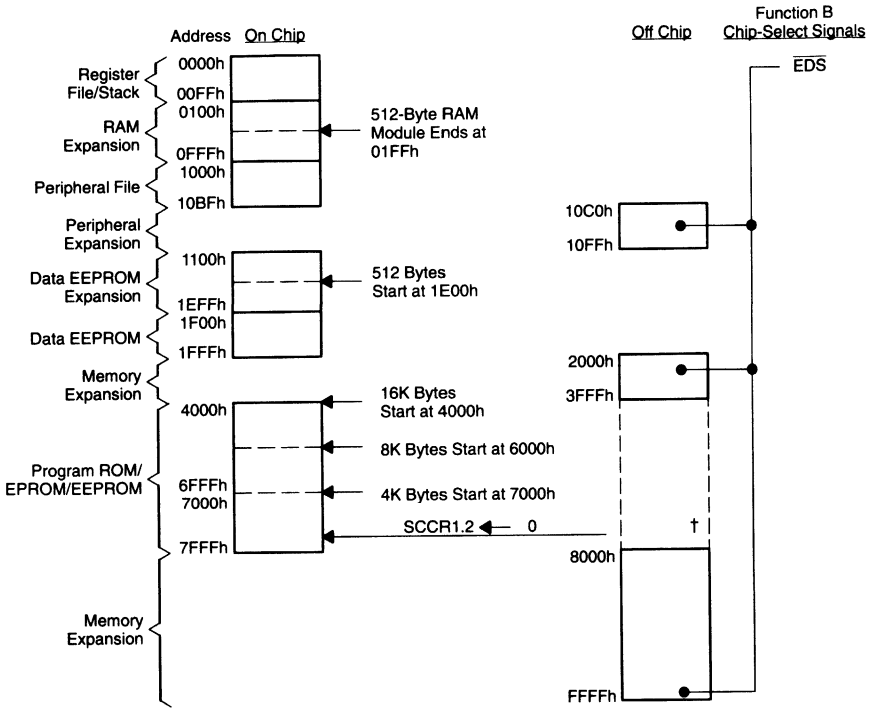
Once the device is in microprocessor mode, the internal program memory can be re-enabled by clearing the MEMORY DISABLE bit (SCCR1.2). This mode is exactly the same as the microprocessor mode without internal memory except that when the internal memory is enabled, the $\overline{\text{EDS}}$ signal is no longer active on memory access to 1020h–102Fh and 4000h–7FFFh. Memory accesses from 4000h to 7FFFh will access internal program memory. The actual amount of program memory available depends on the device.

In this mode, accesses to 1020h–102Fh are not valid for external memory or for the internal port control registers. This peripheral frame should not be used in this mode.

To use this mode, external memory must be implemented at 7FFEh–7FFFh to contain the reset vector. This memory can be switched in and out with the internal memory by clearing and setting the memory disable bit.

Figure 3–10 shows a memory map for the microprocessor mode with internal program memory.

Figure 3–10. Microprocessor Mode With Internal Program Memory



† After reset, until SCCR1.2 is cleared by the program.

To put a device into the microprocessor mode with internal program memory, follow these steps:

1) Place a high logic level on the MC pin.

2) Take the $\overline{\text{RESET}}$ pin active low, then return $\overline{\text{RESET}}$ to its inactive high state.

3) The CPU reads the reset vectors from external memory (7FFEh/7FFFh). The program pointed to by the vectors must include code to clear the MEMORY DISABLE bit (SCCR1.2) to enable the internal memory. The internal program memory is now available (The SCCR1 register is described in subsection 4.3.2, page 4-13).

Note:

Once the MEMORY DISABLE bit is cleared, the external memory at 1020h–102Fh and 4000h – 7FFFh is no longer available to the processor.

3.4.5 Memory Mode Summary

Table 3–4 summarizes the features of each memory mode and outlines the procedure for putting the TMS370 device into each mode. Figure 3–11 gives the memory maps of the four modes.

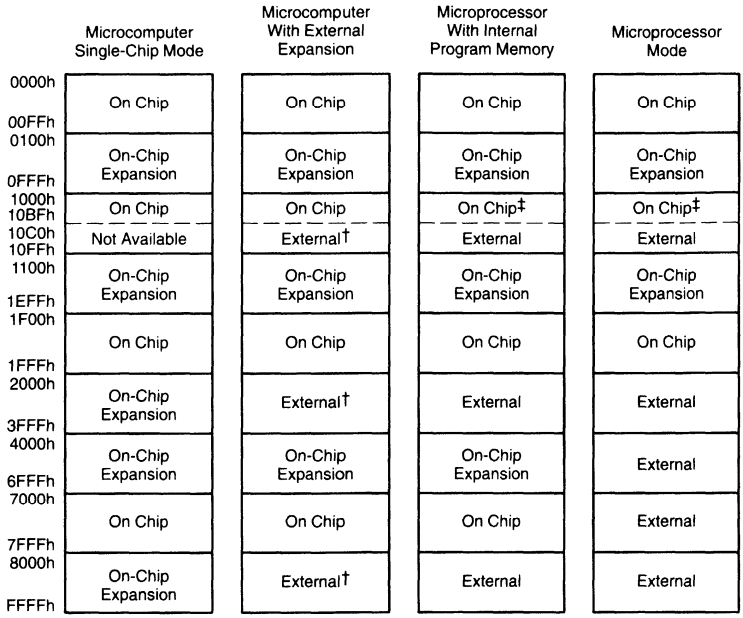
Table 3–4. Operating Mode Summary

Feature	μComputer Single-Chip	μComputer With Expanded Memory	μProcessor with Internal Memory	μProcessor
Devices	All TMS370s with internal program memory	Devices with memory expansion and internal program memory	Devices with memory expansion and internal program memory	Devices with memory expansion
Program Memory	Internal	Internal	Internal and external	External
Ports A, B,C,D	Digital I/O	Digital I/O Function A† Function B‡	Function B‡	Function B‡
Predecoded CS (chip selects)	No	Optional	No	No
Procedure to enter the mode	1) Place logic 0 on the MC pin. 2) Take the $\overline{\text{RESET}}$ pin active low, then release $\overline{\text{RESET}}$.	1) Place logic 0 on the MC pin. 2) Take the $\overline{\text{RESET}}$ pin active low, then release $\overline{\text{RESET}}$. 3) Set digital I/O registers to function A†/B‡.	1) Place logic 1 on the MC pin. 2) Take the $\overline{\text{RESET}}$ pin active low, then release $\overline{\text{RESET}}$. 3) Enable internal memory (clear SCCR1.2).	1) Place logic 1 on the MC pin. 2) Take the $\overline{\text{RESET}}$ pin active low, then release $\overline{\text{RESET}}$.

† Function A: Port D = Chip-select signals $\overline{\text{CSE1}}$, $\overline{\text{CSE2}}$, $\overline{\text{CSH1}}$, $\overline{\text{CSH2}}$, $\overline{\text{CSH3}}$, and $\overline{\text{CSPF}}$ (see Section 4.4, page 4-16).
‡ Function B: Port D = Expansion memory control signals $\overline{\text{OCF}}$, $\overline{\text{EDS}}$, and $\overline{\text{WAIT}}$ (see Section 4.4).

3

Figure 3–11. Memory Operating Modes



† Precoded chip-select outputs available on external expansion memory.
‡ 1020h–102Fh external.

Chapter 4

System and Digital I/O Configuration

This chapter discusses the system and I/O configuration. Features and options are described, as well as the registers that control the configuration. Examples of how to set specific configurations are also given. This chapter covers the following topics:

Topic **Page**

4.1 System Configuration

The system configuration is controlled and monitored by the first three registers of peripheral file frame 1. These registers' names, designations, addresses, and peripheral file register numbers (PF) are shown below and in Table 4–1. The PF numbers are used by peripheral file instructions, for example MOV #00h,P010.

Name	Designation	Address	PF
System control and configuration register 0	SCCR0	1010h	P010
System control and configuration register 1	SCCR1	1011h	P011
System control and configuration register 2	SCCR2	1012h	P012

Table 4–1. Peripheral File Frame 1: System Configuration and Control Registers

Designation	ADDR	PF	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SCCR0	1010h	P010	COLD START	OSC POWER	PF AUTO WAIT	OSC FLT FLAG	MC PIN WPO	MC PIN DATA	—	µP/µC MODE
SCCR1	1011h	P011	—	—	—	AUTO-WAIT DISABLE	—	MEMORY DISABLE	—	—
SCCR2	1012h	P012	HALT/ STANDBY	PWR-DWN/ IDLE	—	BUS STEST	CPU STEST	—	INT1 NMI	PRIVI-LEGE DISABLE

The shaded boxes in Table 4–1 denote privilege mode bits; that is, these bits can be written to only in the privilege mode.

4.1.1 Privilege Mode

The TMS370 architecture allows you to configure the system and peripherals by software to meet the requirements of a variety of applications. The privilege mode of operation ensures the integrity of the system configuration, once it is defined for an application.

Following a hardware reset, the processor operates in the privilege mode. In this mode, peripheral file registers have unrestricted read/write access. The application program can configure the system during the initialization sequence following reset. As the last step of a system initialization, set the PRIVILEGE DISABLE bit (SCCR2.0) to enter the nonprivilege mode and prevent changes to specific control bits within the peripheral file.

Table 4–2 shows the system configuration bits that are write-protected during the nonprivilege mode. These bits should be configured by software before exiting the privilege mode.

Table 4–2. Privilege-Mode Configuration Bits

Register	Bit	Bit Name
SCCR0	PF AUTOWAIT OSC POWER	Peripheral File Automatic Wait Cycle Oscillator Power
SCCR1	MEMORY DISABLE AUTOWAIT DISABLE	Memory Disable Automatic Wait-State Disable
SCCR2	PRIVILEGE DISABLE INT NMI PWRDWN/IDLE HALT/STANDBY	Privilege Mode Disable Interrupt 1, Nonmaskable Interrupt Powerdown/Idle Halt/Standby
SPIPRI	SPI PRIORITY	SPI Interrupt Priority Select
SCIPRI	SCI TX PRIORITY SCI RX PRIORITY	SCI Transmitter Interrupt Priority Select SCI Receiver Interrupt Priority Select
T1PRI	T1 PRIORITY	Timer 1 Interrupt Priority Select
T2PRI	T2 PRIORITY	Timer 2 Interrupt Priority Select
ADPRI	AD PRIORITY	A/D Interrupt Priority Select
PACTPRI	PACT WD PRESCALE SELECT 0 PACT WD PRESCALE SELECT 1 PACT MODE SELECT PACT GROUP 3 PRIORITY PACT GROUP 2 PRIORITY PACT GROUP 1 PRIORITY PACT STEST	PACT Watchdog Prescale Select 0 PACT Watchdog Prescale Select 1 PACT Mode Select PACT Group 3 Priority Select PACT Group 2 Priority Select PACT Group 1 Priority Select PACT Stest
PACTSCR	PACT PRESCALE SELECT 0 PACT PRESCALE SELECT 1 PACT PRESCALE SELECT 2 PACT PRESCALE SELECT 3 FAST MODE SELECT	PACT Prescale Select 0 PACT Prescale Select 1 PACT Prescale Select 2 PACT Prescale Select 3 Fast Mode Select

The only way to change the privilege bits after leaving the privilege mode is to reset the processor and then program the control registers. The write protection override (WPO) used for the EEPROM has no effect on the privilege bits.

Note that privilege mode has no effect on timer 1 watchdog bits. These bits are protected in a separate manner.

4.1.2 Oscillator Fault

The processor contains a system of circuits to monitor the oscillator operation and to detect and contain major oscillator problems. This enhances processor and system reliability and aids in system recovery from a crash that was caused by a temporary fault.

The circuit stops the processor whenever circuitry detects an out-of-range oscillator operation. The oscillator fault detection circuitry consists of:

- An amplitude detector—detects if the oscillator signal has a proper voltage level.
- A frequency detector—senses when the oscillator frequency goes too low. The oscillator fault detection circuit will always trigger below 20 kHz and never above 500 kHz.

The oscillator circuitry is designed to delay operation of the device until a stable clock signal is received. This protects against slow crystal startup times coming out of a halt mode or after an oscillator fault when the input clock cannot be operating at the correct voltage range. The circuitry holds device operation until the input clock signal is within the required voltage range.

Whenever the oscillator fault detection circuitry detects a major oscillator problem, the processor will generate a reset by pulling the $\overline{\text{RESET}}$ pin low for at least eight cycles; this causes external devices to reset along with the processor. After reset, the program can check the oscillator fault flag (OSC FLT FLAG, SCCR0.4), the cold start flag (COLD START, SCCR0.7), and the watchdog reset key (WDRST) to determine the source of the reset. A reset does not clear these flags.

4.1.3 Automatic Wait States

If an application system uses peripherals or expansion memory with access times slower than those of the TMS370 processor, wait states are required. Other microprocessors require complex additional circuitry, but the TMS370 series provides for the automatic addition of wait states that can slow the processor's access time to a compatible period.

In addition, the TMS370 series has a $\overline{\text{WAIT}}$ pin that can hold the processor in a wait state indefinitely. Two bits control the insertion of the automatic wait state:

- The PF AUTOWAIT bit (SCCR0.5) that controls the external frames of the peripheral file so that these frames can access off-chip peripherals.
- The AUTOWAIT DISABLE bit (SCCR1.4) that controls all other external memory.

When the AUTOWAIT DISABLE bit equals 1, any access to *external* memory (excluding the PF file) takes two system clock cycles to complete. When AUTOWAIT DISABLE bit equals 0, the access takes three cycles. The reset value of this bit selects the slower 3-cycle access.

When the PF AUTOWAIT bit equals 1, memory access to the *external* peripheral files takes four system clock cycles. This bit does not affect the accesses to the internal registers. When the PF AUTOWAIT bit equals 0, the memory is treated like any external memory, and the AUTOWAIT DISABLE bit selects the number of cycles per access as either two or three cycles.

Table 4–3 summarizes the effects of the wait-state control bits.

Table 4–3. Wait-State Control Bits

Wait State Control Bits		No. of Clock Cycles per Access	
PF AUTOWAIT Bit (SCCR0.5)	AUTOWAIT DISABLE Bit (SCCR1.4)	PF File	External Memory
0	0	3	3
0	1	2	2
1	0	4	3
1	1	4	2

An external device can pull the $\overline{\text{WAIT}}$ input pin low and cause the processor to wait an indefinite number of clock cycles for its data. When the wait line is released, the processor resynchronizes with the rising edge of the CLOCKOUT signal and continues with the program. The $\overline{\text{WAIT}}$ pin is sampled only during *external* memory cycles.

Note:

When constructing an application circuit with expansion memory, do not forget to connect an unneeded $\overline{\text{WAIT}}$ line to V_{CC}.

4.2 Low-Power and Idle Modes

The OTP, mask-ROM, and reprogrammable EPROM devices have two low-power (powerdown) modes and an idle mode. For mask-ROM devices, low-power modes can be disabled permanently through a programmable contact at the time when the mask is manufactured (refer to Chapter 17, *Customer Information*, for order information about mask-ROM devices).

Note:

Low-power modes operate differently for TMS370Cxxx devices. Refer to Section A.5, page A-4.

The low-power modes reduce the operating power by reducing or stopping the activity of various modules. The processor has two types of low-power modes: the halt mode and the standby mode (see Table 4–4). Bits 6 and 7 of SCCR2 select the halt, standby, or idle modes.

- The **standby** mode stops the internal clock in every module except the timer 1 module. The timer 1 module continues to run and can bring the processor out of the standby mode. In devices with the PACT module, only the default timer and the first command are active in standby mode.
- The **halt** mode stops the internal clock. This stops processing in all of the modules, resulting in the lowest power consumption.
- The **idle** mode (which is not a low-power mode) is a state that waits for the next interrupt.

Executing an IDLE instruction causes the processor to enter one of the two low-power modes or the simple idle mode, depending on SCCR2.6 and SCCR2.7. The low-power and idle mode selection bits are summarized in Table 4–4.

Table 4–4. Powerdown/Idle Control Bits

Powerdown Control Bits		
PWRDWN/IDLE (SCCR2.6)	**HALT/STANDBY (SCCR2.7)**	**Mode Selected**
1	0	Standby
1	1	Halt
0	X†	Idle

† don't care

When low-power modes are disabled through a programmable contact in the mask-ROM devices, writing to the SCCR2.6–7 bits is ignored. In addition, if you execute an IDLE instruction when low-power modes are disabled through a programmable contact, the device will *always* enter the idle mode.

To provide a method of exiting low-power modes for mask-ROM devices, INT1 is automatically enabled as a nonmaskable interrupt (NMI) during low-power modes when the hard watchdog mode is selected. This means that the NMI is always generated, regardless of the interrupt enable flags and the values of these status bits: INT1 PRIORITY bit (INT1.1), INT1 ENABLE bit (INT1.0), INT1 NMI bit (SCCR2.1), and global interrupt enable flags of the status register (IE1 and IE2). See subsection 7.7.2.1, page 7-21, for more information. The low-power modes and the methods of exiting these modes are discussed further in subsections 4.2.1 and 4.2.2.

In the standby and halt modes, the digital output ports remain active. In addition, the following information is retained:

- ❑ The CPU registers:
  - PC
  - Status
  - Stack pointer
- ❑ The contents of the RAM
- ❑ The digital output data registers
- ❑ Control and status registers of all the modules, including the timer contents and the watchdog counter

If the serial peripheral interface (SPI) or serial communications interface (SCI) is in the process of receiving or transmitting data when a low-power mode is entered, the data received or transmitted can be lost. The results of an A/D conversion or an EEPROM write in process will be invalid when a low-power mode is entered.

Use the watchdog mode with caution; the watchdog stops counting in both low-power modes. In the standard watchdog option, if the program executes an IDLE instruction without the interrupts enabled (described in subsections 4.2.1 and 4.2.2), then only a reset can start the processor running again.

For additional information regarding watchdog operation during low-power modes, see subsection 7.7.2.1, page 7-21, and Section 7.8, page 7-24.

4.2.1 Standby Mode

The standby mode uses less power than the normal operating mode but more than the halt mode. The standby mode stops the clocks to every module except the timer 1 module or the PACT module. These modules can bring the processor out of this low-power mode if the interrupts are enabled. To enter the standby mode:

- ❑ Set the PWRDWN/IDLE bit (SCCR2.6)
- ❑ Clear the HALT/STANDBY bit (SCCR2.7).
- ❑ The next execution of an IDLE instruction causes the processor to enter the standby mode.

You can cause the processor to exit the standby mode in one of the following four ways:

- ❑ Reset
- ❑ External interrupt 1, 2, or 3 (if enabled)
- ❑ Low level on the SCIRXD pin if the SCI RX interrupt and receiver are enabled (described in Chapter 9)
- ❑ Timer 1 or PACT's first command/definition entry interrupt, if enabled

For additional standby mode power savings, see subsection 4.2.3.

4.2.2 Halt Mode

The halt mode stops all internal operations and clocks (including timer 1 and the PACT counter) and uses the least power of the low-power modes. Timer 1 cannot bring the processor out of this low-power mode. To select the halt mode:

- ❑ Set the PWRDWN/IDLE bit (SCCR2.6),
- ❑ Set HALT/STANDBY bit (SCCR2.7), and
- ❑ Execute an IDLE instruction.

You can cause the processor to exit the halt mode in one of the following three ways:

- ❑ Reset
- ❑ External Interrupt 1, 2, or 3 if enabled
- ❑ Low level on the SCIRXD pin if the SCI RX interrupt and receiver are enabled (described in Chapter 9)

4.2.3 Using Interrupts to Exit From the Halt Mode

You must be aware of several items when using an interrupt to exit the halt mode.

- ❑ Interrupts enabled during halt mode are level sensitive and not edge sensitive.
- ❑ The interrupt must be at the inactive level when the device enters halt mode.
- ❑ The processor will exit the halt mode when the interrupt goes from the inactive level to the active level.

- ❑ Bit 2 in the interrupt control register determines the active and inactive levels.
  - ■ When the INT2 POLARITY bit (INT2.2) is selected to be triggered on the rising edge, the active level is high.
  - ■ When the INT2 POLARITY bit is selected to be triggered on the falling edge, the active level is low.
- ❑ Ensure that the interrupt is at an active level until the oscillator is stable enough to generate an interrupt.
- ❑ If the halt mode is entered with the interrupt at an active level, the processor will exit the halt mode and enter the idle mode.
- ❑ When the selected interrupt edge is detected, the program will continue.

Refer to Figure 4–1 and Figure 4–2 for the differences between the correct and incorrect methods for entering halt mode.

Figure 4–1. Correct Method to Enter Halt Mode

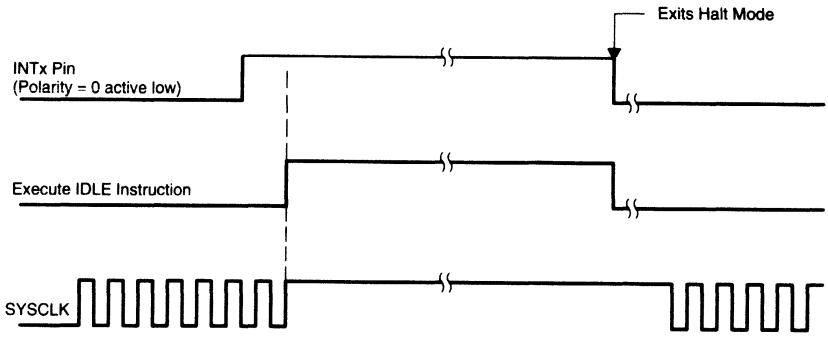
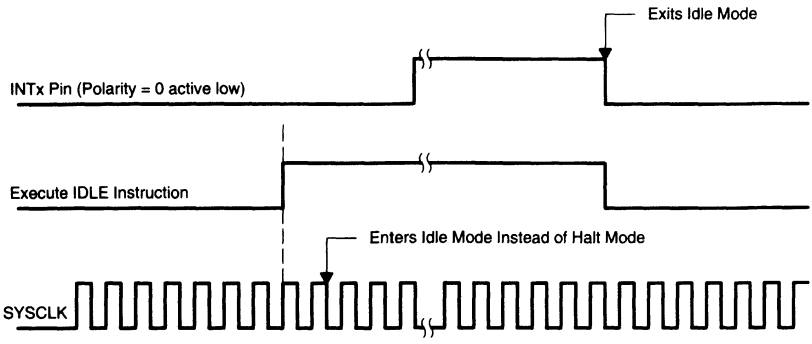


Figure 4–2. Improper Method to Enter Halt Mode



If this halt-mode condition exists and the device uses the watchdog, then the watchdog can reset while the program is waiting and unable to service the watchdog in the normal power idle mode.

The same considerations apply when you use the SCIRXD pin to exit the halt mode. The processor will exit halt mode anytime an enabled SCI receiver and pin detect a low level on SCIRXD.

4.2.4 Oscillator Power Bit

The OSC POWER bit (SCCR0.6) allows additional standby mode power savings. When in effect, this feature reduces the oscillator drive current and disables the oscillator fault detection circuitry. The OSC POWER bit can be used effectively between 2 MHz and 12 MHz. If the oscillator frequency is greater than 12 MHz, this bit must be cleared. For power reduction specifications, see Chapter 16, *Electrical Specifications.*

4.3 System Control Registers

Each system control register is summarized in the following charts with definitions of each control bit.

4.3.1 System Control and Configuration Register 0 (SCCR0)

System Control and Configuration Register 0 (SCCR0)
[Memory Address – 1010h]

Bit #	7	6	5	4	3	2	1	0
P010	COLD START	OSC POWER	PF AUTOWAIT	OSC FLT FLAG	MC PIN WPO	MC PIN DATA	—	µP/µC MODE
	RW–*	RP–0	RP–0	RW–0	R–0	R–*		R–*

R = Read, W = Write, P= Privilege write only, C = Clear only,
–n = Value of the bit after the register is reset, –* = See bit description

Bit 0 **µP/µC MODE.** Microprocessor/Microcomputer Mode.

This bit indicates the current operating mode (as described in subsection 3.4, page 3-15).

0 = Currently operating in microcomputer mode.
1 = Currently operating in microprocessor mode.

Bit 1 **Reserved.** Read data is indeterminate.

Bit 2 **MC PIN DATA.** Mode Control Pin Data.

This bit shows the current status of the MC pin.

0 = Voltage on the MC pin is a logic 0 level.
1 = Voltage on the MC pin is a logic 1 level.

Bit 3 **MC PIN WPO.** Mode Control Pin Write Protect Override Status.

This bit indicates whether or not the voltage on the MC pin is adequate for WPO functions. (If this bit is set, then bit 2 is also set.)

0 = Voltage on the MC pin is not enough to override write protection.
1 = Voltage on the MC pin is enough for write-protect operation override. Protected bits in data EEPROM and program EEPROM can now be written to. Override voltage is nominally 12 volts.

Bit 4 **OSC FLT FLAG.** Oscillator Fault Flag.

This flag is reset upon an initial power-up reset. A reset under power does not affect this flag. Therefore, this bit can be polled to determine the source of a reset.

0 = No oscillator fault found.
1 = Oscillator fault found. Oscillator period is now or was out of correct operating range. The oscillator fault detect circuit always triggers below 20 KHz and never above 500 kHz.

Bit 5 **PF AUTOWAIT.** Peripheral File Automatic Wait Cycle.

0 = Any access to the peripheral file will take two system clock cycles with no system auto wait (bit 4 of SCCR1=1), or three system clock cycles with the system auto-wait on (bit 4 of SCCR1=0). (See subsection 4.1.3, page 4-4.)

1 = Any access to the upper four frames of the peripheral file (address 10C0h to 10FFh) will take four system clock cycles to complete. This eases interface requirements for peripheral devices slower than the TMS370 processor. Normal full-speed operation consists of two system clock cycles per access.

Bit 6 **OSC POWER.** Oscillator Power.

This bit controls an oscillator power reduction feature. When this feature is in effect, the oscillator drive current is reduced, and the oscillator fault detection circuitry is powered down. Current reduction is most useful in the standby mode. However, when this bit is set during normal operation, the operating mode power consumption can be slightly reduced. When operating in the halt mode, this bit has no effect because the oscillator is not active. This feature is effective up to a 12-MHz maximum oscillator frequency. If the oscillator frequency is greater than 12 MHz, this bit must be cleared. For power reduction specifications, see Chapter 16, *Electrical Specifications.*

0 = No oscillator drive current reduction.

1 = Oscillator drive current reduction.

Bit 7 **COLD START.** Cold Start Flag.

This bit indicates whether or not the microcontroller is coming out of power-up reset. This bit does not change during a reset under normal power.

0 = No full-power cycle occurred since last writing a 0 to this bit. This setting is used to determine the source of a reset.

1 = A full-power cycle has occurred since last writing a 0 to this bit. If the application does not zero this bit, the bit has no meaning.

Only writing a 0 to this bit can clear the COLD START flag. This bit is set to 1 only after the V_{CC} is off for several hundred milliseconds. As a result, you cannot use this bit to detect short V_{CC} glitches or brown-out conditions.

4.3.2 System Control and Configuration Register 1 (SCCR1)

System Control and Configuration Register 1 (SCCR1)
[Memory Address – 1011h]

Bit #	7	6	5	4	3	2	1	0
P011	—	—	—	AUTO-WAIT DISABLE	—	MEMORY DISABLE	—	—
				RP–0		RP–*		

R = Read, P= Privilege write only, –n = Value of the bit after the register is reset, –* = See bit description

Bits 0,1,3,5,6,7 **Reserved.** Read data is indeterminate.

Bit 2 **MEMORY DISABLE.**

This bit enables or disables the internal program memory (memory addresses 4000h–7FFFh). This bit does not affect data EEPROM or internal RAM. A reset initializes this bit to the state of the MC pin. Changes to this bit can occur only in the privilege mode.

0 = Enables internal program memory and accesses internal memory at these locations. The $\overline{EDS}$ memory signal will not appear during access to locations 4000h–7FFFh and 1020h–102Fh. These ranges are accessed as on-chip memory.

1 = Disables internal program memory and accesses external memory at these locations. An operation on these locations generates an external memory bus cycle with the $\overline{EDS}$ memory signal validating the access. This bit disables the program EPROM control register, EPCTL (described in Section 6.4), if applicable, and disables 1020h–102Fh. These ranges are accessed as off-chip memory.

Bit 4 **AUTOWAIT DISABLE.** Automatic Wait State Disable.

This bit, which is cleared at reset, causes an extra cycle to be added to all external bus accesses in order to accommodate slower memory.

0 = Enables the autowait feature and makes the external bus access three system clock cycles long.

1 = Disables the autowait feature and makes the external bus access two system clock cycles long.

Changes to this bit can occur only in the privilege mode. If the PF AUTOWAIT bit in SCCR0 is set, external peripheral file access takes four system clock cycles, regardless of the AUTOWAIT DISABLE bit.

4.3.3 System Control and Configuration Register 2 (SCCR2)

System Control and Configuration Register 2 (SCCR2)
[Memory Address – 1012h]

Bit #	7	6	5	4	3	2	1	0
P012	HALT/ STANDBY	PWRDWN/ IDLE	—	BUS STEST	CPU STEST	—	INT1 NMI	PRIVILEGE DISABLE
	RP–0	RP–0		RP–0	RP–1		RP–0	RS–0

R = Read, P= Privilege write only, S = Set only, –n = Value of the bit after the register is reset

Bit 0 **PRIVILEGE DISABLE.** Privilege Mode Disable.

Many bits controlling the system configuration can be changed only while in the privilege mode. After setting the system configuration bits, write a 1 to the PRIVILEGE DISABLE bit to disable the privilege mode and lock out any changes to the privilege protected bits. Only a reset can clear this bit.

0 = System is operating in privilege mode.
1 = System is not operating in privilege mode.

Bit 1 **INT1 NMI.** Interrupt 1, Nonmaskable Interrupt.

This bit determines whether interrupt 1 is maskable or nonmaskable (NMI). When interrupt 1 is nonmaskable, it is the second highest priority interrupt (reset is highest) and is unaffected by the interrupt mask described in subsection 5.1.2, page 5-6. The NMI mode disables the enable and priority select bits of the interrupt 1 control register. The program can change this bit only in the privilege mode.

When programming an EEPROM, you must ensure that nonmaskable interrupt routines do not access the EEPROM between an EEPROM write instruction and the point when the EXE bit (DEECTL.0) is set to 1, or data will be corrupted.

0 = Interrupt 1 is maskable.
1 = Interrupt 1 is nonmaskable (NMI).

Bit 2 **Reserved.** Read data is indeterminate.

> **Note:**
>
> This bit operates differently for TMS370Cxxx devices. Refer to Section A.3, page A-4.

Bit 3 **CPU STEST.** CPU STEST bit.

This bit is used only during factory test and has no effect in normal operating modes.

Bit 4 **BUS STEST.** BUS STEST bit.

This bit must be cleared (0) to ensure proper operation.

Bit 5 **Reserved.** Read data is indeterminate.

> **Note:**
>
> This bit operates differently for TMS370Cxxx devices. Refer to Section A.3, page A-4.

Bit 6 **PWRDWN/IDLE.** Powerdown/Idle.

This bit determines the mode entered by the CPU when an IDLE instruction is executed. Changes to this bit can occur only in the privilege mode.

0 = The processor will enter an idle mode when the program executes an IDLE instruction. The processor waits at the IDLE instruction until any enabled interrupt occurs. The processor then enters the interrupt routine and returns to the instruction after the IDLE instruction. The idle is not a low-power mode.

1 = The processor will enter a low-power mode when the program executes an IDLE instruction. The HALT/STANDBY bit determines the type of low-power mode.

Bit 7 **HALT/STANDBY.**

The following descriptions apply only if the PWRDWN/IDLE bit is set; otherwise, the HALT/STANDBY bit has no effect. See subsection 4.2 for a description of the halt and standby modes. Changes to this bit can occur only in the privilege mode.

0 = When an IDLE instruction is executed, the processor will enter the standby mode, which stops program execution and disables the system clock to all nonessential peripherals. The system clock to the timer 1 continues to run, and the timer can generate an interrupt to bring the processor out of the standby mode.

1 = When an IDLE instruction is executed, the processor will enter the halt mode, which stops the internal oscillator and suspends the system and peripheral operations. This mode provides the lowest power consumption.

4.4 Digital I/O Configuration

On TMS370 devices, the power, reset, MC, and crystal pins are dedicated to one function. The remaining pins can be programmed to be a general-purpose input and/or output or a special function pin. Some of these pins are associated with the functions of the peripheral modules. The pins are briefly described below and are summarized in Table 4–5.

- On TMS370Cx1x devices, 13 of a possible 22 I/O pins are dedicated to ports A and D. Port A contains eight pins, and port D contains five pins.
- On TMS370Cx2x devices, 22 of a possible 34 I/O pins are dedicated to ports A, B, C, and D. Ports A and B each have eight pins. Port C contains one pin, and port D contains five pins.
- On TMS370Cx3x devices, 12 of a possible 36 I/O pins are dedicated to ports A and D. Port A contains eight pins. Port D contains four pins.
- On TMS370Cx4x devices, 16 of possible 40 or 44 I/O pins are dedicated to ports A, B, and D. Port A contains eight pins, port B contains three pins, and port D contains five pins.
- On 68-pin TMS370Cx5x devices, 32 of a possible 55 I/O pins are dedicated to ports A, B, C, and D; each port has eight pins. On 64-pin TMS370Cx5x devices, 30 of a possible 53 I/O pins are dedicated to ports A, B, C, and D; ports A, B, and C each have eight pins, and port D has six.

Frame 2 of the peripheral file (memory addresses 1020h–102Fh) contains the control registers for reading, writing, and configuring ports A, B, C, and D. These registers are shown in Figure 4–3 on the following page.

Table 4–5. Digital I/O Pins by Device

Device Category	Maximum Digital I/O		
	Bidirectional	Input Only	Output Only
TMS370Cx1x	21	1	0
TMS370Cx2x	33	1	0
TMS370Cx3x	14	13	9
TMS370Cx4x	27	5/9†	0
TMS370Cx5x	46/44‡	9	0

† 5 input pins for 40-pin devices; 9 input pins for 44-pin devices
‡ 46 bidirectional pins for 68-pin devices; 44 bidirectional pins for 64-pin devices

Figure 4–3. Peripheral File Frame 2: Digital Port Control Registers

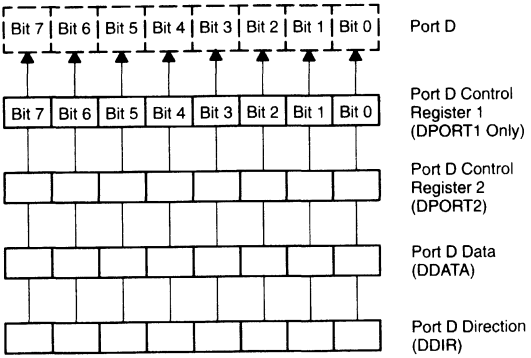
Designation	ADDR	PF	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
APORT1	1020h	P020	Reserved							
APORT2	1021h	P021	Port A Control Register 2							
ADATA	1022h	P022	Port A Data							
ADIR	1023h	P023	Port A Direction							
BPORT1	1024h	P024	Reserved							
BPORT2	1025h	P025	Port B Control Register 2							
BDATA	1026h	P026	Port B Data							
BDIR	1027h	P027	Port B Direction							
CPORT1	1028h	P028	Reserved							
CPORT2	1029h	P029	Port C Control Register 2							
CDATA	102Ah	P02A	Port C Data							
CDIR	102Bh	P02B	Port C Direction							
DPORT1	102Ch	P02C	Port D Control Register 1							
DPORT2	102Dh	P02D	Port D Control Register 2							
DDATA	102Eh	P02E	Port D Data							
DDIR	102Fh	P02F	Port D Direction							

Each port has up to four control registers associated with it. They are:

- Port X control register 1 (XPORT1)
- Port X control register 2 (XPORT2)
- Port X data (XDATA)
- Port X direction (XDIR)

The same bit position of each of these registers affects the corresponding bit in the port. For example, bit 0 of registers DPORT1, DPORT2, DDATA, and DDIR control port D, bit 0. This is illustrated in Figure 4–4.

Figure 4–4. Port Control Register Operation



Bits from the XPORT1 and XPORT2 registers determine the function of the corresponding port pin to be an I/O, data, address, or control signal, depending on the port. The same bit from the XDIR register determines the direction (input or output) if the pin has been defined as an I/O pin. The same bit from the XDATA register is the bit to write to or read from if the pin has been defined as an I/O pin.

Table 4–6 shows the function that each pin can serve, depending on which port contains the pin. Definitions of the memory expansion signals of function A and function B follow the table.

Table 4–6. Port Configuration Registers Set-Up

Port	Pin	Input	Output	Function A (µC Mode)	Function B (µC Mode)	µP Mode
		XPORT1 = 0† XPORT2 = 0 XDATA = y XDIR = 0	XPORT1 = 0† XPORT2 = 0 XDATA = q XDIR = 1	XPORT1 = 0† XPORT2 = 1 XDATA = x XDIR = x	XPORT1 = 1† XPORT2 = 1 XDATA = x XDIR = x	
A	0–7	Data IN y	Data OUT q	DATA BUS	‡	DATA BUS
B	0–7	Data IN y	Data OUT q	LOW ADDR	‡	LOW ADDR
C	0–7	Data IN y	Data OUT q	HI ADDR	‡	HI ADDR
D	0	Data IN y	Data OUT q	$\overline{\text{CSE2}}$	$\overline{\text{OCF}}$	$\overline{\text{OCF}}$
D	1	Data IN y	Data OUT q	$\overline{\text{CSH3}}$		‡
D	2	Data IN y	Data OUT q	$\overline{\text{CSH2}}$		‡
D	3	Data IN y	Data OUT q	CLKOUT	CLKOUT	CLKOUT
D	4	Data IN y	Data OUT q	R/$\overline{\text{W}}$	R/$\overline{\text{W}}$	R/$\overline{\text{W}}$
D	5	Data IN y	Data OUT q	$\overline{\text{CSPF}}$		‡
D	6	Data IN y	Data OUT q	$\overline{\text{CSH1}}$	$\overline{\text{EDS}}$	$\overline{\text{EDS}}$
D	7	Data IN y	Data OUT q	$\overline{\text{CSE1}}$	$\overline{\text{WAIT}}$	$\overline{\text{WAIT}}$

If XPORT1 = 1 and XPORT2 = 0, the function is not valid.
† XPORT1 exists for DPORT only.
‡ Reserved. Not available.

4.4.1 Function A and Function B Signal Definitions

The following paragraphs define the memory expansion signals of function A and B.

$\overline{\text{CSE1}}$ Chip-Select Eighth 1. This signal selects the first bank of memory. It has the same timing as $\overline{\text{EDS}}$, but it goes active only during accesses to an eighth of memory (locations 2000h–3FFFh). Setting this pin to a high-level general-purpose output disables the bank.

$\overline{\text{CSE2}}$ Chip-Select Eighth 2. This signal selects a second bank of memory. It has the same timing as $\overline{\text{EDS}}$, but it goes active only during accesses to an eighth of memory (locations 2000h–3FFFh). Setting this pin to a high-level general-purpose output disables the bank.

$\overline{\text{CSPF}}$ Chip-Select Peripheral File. This signal has the same timing as $\overline{\text{EDS}}$, but it goes active only during access to external frames of the peripheral file (locations 10C0h–10FFh).

$\overline{\text{CSH1}}$ Chip-Select Half 1. This signal selects the first bank of memory. It has the same timing as $\overline{\text{EDS}}$, but it goes active only during access to the upper half of memory (locations 8000h–FFFFh). Setting this pin to a high-level general-purpose output disables the bank.

$\overline{\text{CSH2}}$ Chip-Select Half 2. This signal selects a second bank of memory. It has the same timing as $\overline{\text{EDS}}$, but it goes active only during access to the upper half of memory (locations 8000h–FFFFh). Setting this pin to a high-level general-purpose output disables the bank.

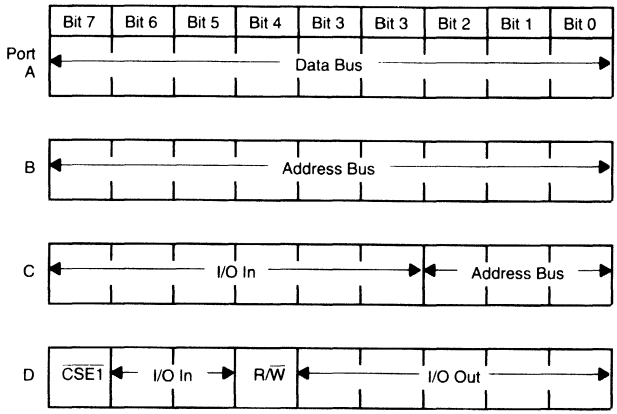
$\overline{CSH3}$	Chip-Select Half 3. This signal selects a third bank of memory. It has the same timing as $\overline{EDS}$, but it goes active only during access to the upper half of memory (locations 8000h–FFFFh). Setting this pin to a high-level general-purpose output disables the bank.
CLKOUT	Clock Output. This signal synchronizes external peripherals. It outputs one quarter of the crystal or external oscillator frequency.
DATA BUS	External data bus. Input and output.
$\overline{EDS}$	External Data Strobe. This signal goes low during external memory operations. The rising edge of $\overline{EDS}$ validates the read input data; the write data is available after the falling edge of $\overline{EDS}$.
LOW ADDR/HI ADDR	External memory address bus. Output only.
R/$\overline{W}$	Read or Write operation. Goes high at the beginning of read operations and low during write operations. This line is active during both internal and external accesses.
$\overline{OCF}$	Opcode Fetch. Goes low at the beginning of a memory read operation that fetches the first byte of an instruction. It then resumes its high level at the end of the opcode fetch(s).
$\overline{WAIT}$	Wait input. An external, low signal applied to this pin, when sampled, causes the processor to hold the information on the expansion bus for one or more extra clockout cycles. This pin is sampled during the rising edge of CLKOUT after $\overline{EDS}$ goes active.

The predecoded chip selects allow the TMS370 to access external addresses with a minimum of external logic. In many cases, no external logic is necessary between the TMS370 and the peripheral device, because of the predecoded chip selects, autowait features, and the nonmultiplexed bus. Chip selects also make it easy to do memory bank selection. Without bank selection, the $\overline{CSH1}$, $\overline{CSE1}$, and $\overline{CSPF}$ signals can easily access about 40K bytes of memory in the three different areas. With bank selection, the processor can access 112K bytes of memory.

To illustrate configuring the digital ports, assume that a TMS370C050 is to operate in the expanded microcomputer mode, and that 2K bytes of memory is needed at 2000h to 27FFh. The top half of Example 4–1 shows the desired port configuration.

- Port A is set as the external data bus.
- Port B is the low-order address bits of the eleven bits necessary to access 2K bytes of memory.
- Bits 4 through 7 of port C are set as I/O input.
- In port D, bit 7 is the chip-select signal to access 2000h to 3FFFh, and bit 4 is for external memory control signal R/$\overline{W}$. The remaining bits of port D are used as I/O output.

Example 4–1. Digital Ports Set-Up



Port A Control Registers

1021h	1	1	1	1	1	1	1	1	1	MOV #0FFH, P021
1022h	x	x	x	x	x	x	x	x	x	
1023h	x	x	x	x	x	x	x	x	x	

Port B Control Registers

1025h	1	1	1	1	1	1	1	1	1	MOV #0FFH, P025
1026h	x	x	x	x	x	x	x	x	x	
1027h	x	x	x	x	x	x	x	x	x	

Port C Control Registers

1029h	0	0	0	0	0	0	1	1	1	MOV #007H, P029
102Ah	x	x	x	x	x	x	x	x	x	
102Bh	0	0	0	0	0	0	x	x	x	MOV #0, P02B

Port D Control Registers

102Ch	0	0	0	0	0	0	0	0	0	MOV #0, P02C
102Dh	1	0	0	1	0	0	0	0	0	MOV #090h, P02D
102Eh	x	q	q	x	q	q	q	q	q	
102Fh	x	1	1	x	1	1	1	1	1	MOV #0FFH, P02F

The bottom half of Example 4–1 shows the port control registers set up to establish the configuration shown in the top half. To determine the bits needed to set the registers, use Table 4–6 on page 4-19. For example, to set port A as the data bus, find Port A in the left-hand column of Table 4–6. Look across the row to find data bus, then follow the column up to the bit settings for each part in the column heading:

1
x
x

The assembly language instructions in the right column of Example 4–1 show one method of setting up the registers to the left. The Pxxx operand indicates peripheral file access (see Chapter 13, *Assembly Language Instruction Set,* for more information on peripheral file instructions).

When the device operates with internal program memory disabled, any access to the port peripheral frame, 1020h–102Fh, is decoded as external address. Memory accesses to this frame can control external hardware that emulates the digital I/O functions.

4.4.2 Microprocessor Mode

Initializing a device with bus expansion to the microprocessor mode forces ports A, B, C, and D to function B as shown in Table 4–6, page 4-19. Port A is the data bus, port B is the low-order-address bus, and port C is the high-order-address bus in this mode. Devices that are not defined for operation in the memory expansion modes must be powered up in the microcomputer single-chip mode.

4.4.3 Microcomputer Mode

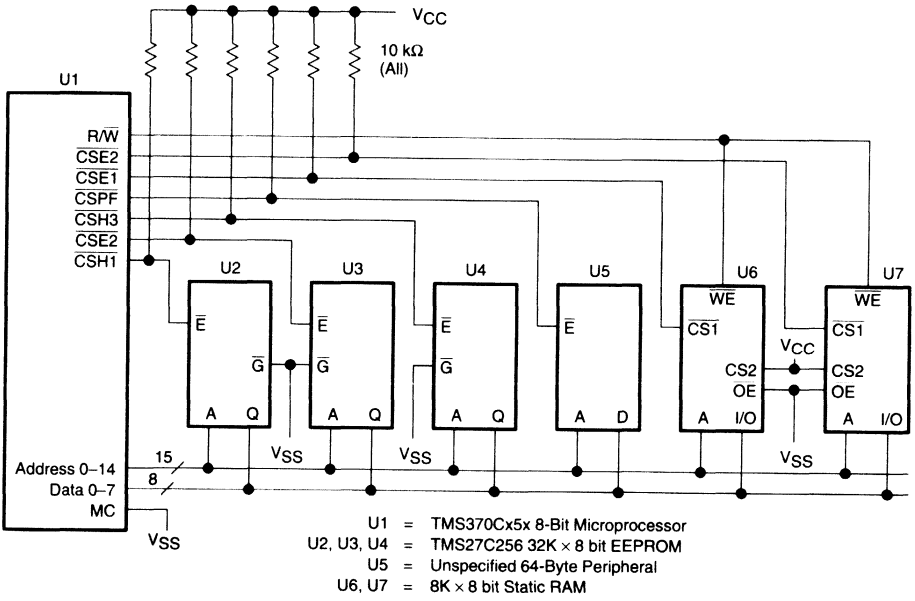
Initializing the device to the microcomputer mode forces ports A, B, C, and D to general-purpose high-impedance inputs. The program can set the control bits to change the function of the port pins to one of four functions: general-purpose output, general-purpose input, function A, or function B.

When you change a pin from a general-purpose input pin to an output pin, write to the data register first to set up the data; then, set the data direction register. This prevents unknown data on the pin from interfering with the external circuitry.

The TMS370 in the microcomputer mode can individually reconfigure any address, data, or control signal to use only the necessary signals and leave the other signals on the port for general-purpose I/O operations.

Figure 4–5 shows an example of the TMS370Cx5x interfaced to 112K bytes of external memory (see Chapter 14, *Design Aids*, for a more complete example). The function-A chip-select signals enable one of three banks of EPROM, an external peripheral device, and one of two banks of static RAM. In this example, all eight bits of port A are the data bus, all eight bits of port B the address LSbyte, and seven port C bits complete the 15-bit address bus.

Figure 4–5. System Interface Example



Chapter 5

Interrupts and System Reset

This chapter discusses the internal and external interrupts of the TMS370. The methods of device reset are also discussed. This chapter covers the following topics:

5

Topic **Page**

5.1 Interrupts

The TMS370 programmable interrupt structure allows flexible on-chip and external interrupt configurations to meet real-time interrupt-driven application requirements.

Whenever an internal or external circuit requests an enabled interrupt, the processor finishes the current instruction and then fetches, from the interrupt table, the address of the appropriate interrupt service routine. The processor then pushes the contents of the program counter and status register onto the stack and begins execution at the interrupt service routine address found in the interrupt table. When the interrupt service routine completes, the program executes an RTI (return from interrupt) instruction, which pops the previous status register and program counter contents from the stack. The processor resumes execution from the point of interruption.

Table 5–1 shows the interrupt and reset vectors by device category.

Table 5–1. Interrupts and Reset Vectors

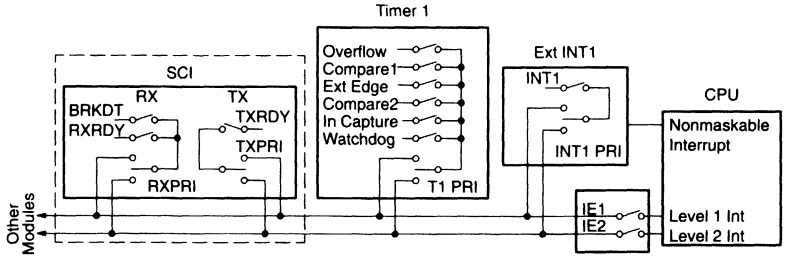
Device Category	Interrupts/Reset		
	External†	Vectors Total	Sources Total
TMS370Cx1x	4	6	13
TMS370Cx2x	4	8	16
TMS370Cx3x	4	23	25
TMS370Cx4x	4	9	22
TMS370Cx5x	4	10	23

† Three external interrupts and a reset

5.1.1 Interrupt Operation

The hardware interrupt structure includes two selectable priority levels as shown in Figure 5–1. Interrupt level 1 has a higher priority than interrupt level 2. The two priority levels can be independently masked by clearing the global interrupt enable bits (IE1 and IE2) of the status register (described in subsection 3.2.2, page 3-5).

Figure 5–1. Interrupt Control



During system initialization, the application program can assign system interrupts to either the high or low priority level. The program can reassign priority levels at any time, except for those priority levels that are protected by the privilege mode. Within each level, the hardware determines the interrupt priority.

The processor services the pending interrupts upon completion of the current instruction execution, depending on interrupt mask and priority conditions. The processor services all enabled level 1 interrupts before servicing any level 2 interrupts. Within each level, the processor services the highest priority interrupts first. Table 5–2 shows the hardware priorities of the devices at time of this printing.

Table 5–2. Module Interrupt Priority

Vector Start Address	Module		Device Module Priority					Module Vector Bytes
			TMS370Cx1x	TMS370Cx2x	TMS370Cx3x	TMS370Cx4x	TMS370Cx5x	
7F9Ch	PACT	Group 2	NA	NA	6 (see Note)	NA	NA	36
		Group 3	NA	NA	7 (see Note)	NA	NA	
		Group 1	NA	NA	5 (see Note)	NA	NA	
7FECh	A/D Converter		NA	NA	8	10	10	2
7FEEh	Timer 2		NA	NA	NA	9	9	2
7FF0h	SCI	Transmit	NA	8	NA	8	8	4
		Receive	NA	7	NA	7	7	
7FF4h	Timer 1		6	6	NA	6	6	2
7FF6h	SPI		5	5	NA	NA	5	2
7FF8h	External INT3		4	4	4	4	4	6
7FFAh	External INT2		3	3	3	3	3	
7FFCh	External INT1		2	2	2	2	2	
7FFEh	$\overline{\text{RESET}}$		1	1	1	1	1	2

Note: The in-circuit emulator with PACT has the PACT module as the lowest priority interrupts, while the device family (TMS370Cx3x) has the A/D interrupt at lowest priority.

The TMS370 architecture allows up to 128 independent interrupt vectors. These system interrupt vectors must be located within memory addresses 7F00h to 7FFFh. This memory space also contains the trap tables and 12 bytes reserved for Texas Instruments use. If the device does not define this memory for an interrupt vector, it can be used for program memory. Table 5–3 shows the interrupt vector source(s) and corresponding address(es). Note that a system interrupt can have multiple interrupt sources.

Table 5–3. Interrupt Vector Sources

Module	Vector Address	Interrupt Source	Interrupt Flag	System Interrupt	Priority in Group†
PACT	7F9Ch, 7F9Dh	PACT SCI TXINT	PACT TXRDY	PTXINT	2
(Group 2)	7F9Eh, 7F9Fh	PACT SCI RXINT	PACT RXRDY	PRXINT	1
PACT (Group 3)	7FA0h, 7FA1h	PACT Cmd/Def Entry 0	CMD/DEF INT 0 FLAG	CDINT0	1
	7FA2h, 7FA3h	PACT Cmd/Def Entry 1	CMD/DEF INT 1 FLAG	CDINT1	2
	7FA4h, 7FA5h	PACT Cmd/Def Entry 2	CMD/DEF INT 2 FLAG	CDINT2	3
	7FA6h, 7FA7h	PACT Cmd/Def Entry 3	CMD/DEF INT 3 FLAG	CDINT3	4
	7FA8h, 7FA9h	PACT Cmd/Def Entry 4	CMD/DEF INT 4 FLAG	CDINT4	5
	7FAAh, 7FABh	PACT Cmd/Def Entry 5	CMD/DEF INT 5 FLAG	CDINT5	6
	7FACh, 7FADh	PACT Cmd/Def Entry 6	CMD/DEF INT 6 FLAG	CDINT6	7
	7FAEh, 7FAFh	PACT Cmd/Def Entry 7	CMD/DEF INT 7 FLAG	CDINT7	8
PACT (Group 1)	7FB0h, 7FB1h	PACT Circular Buffer (Half/Full)	BUFFER HALF/FULL INT FLAG	BUFINT	1
	7FB2h, 7FB3h	PACT CP6 Edge	CP6 INT FLAG	CP6INT	2
	7FB4h, 7FB5h	PACT CP5 Edge	CP5 INT FLAG	CP5INT	3
	7FB6h, 7FB7h	PACT CP4 Edge	CP4 INT FLAG	CP4INT	4
	7FB8h, 7FB9h	PACT CP3 Edge	CP3 INT FLAG	CP3INT	5
	7FBAh, 7FBBh	PACT CP2 Edge	CP2 INT FLAG	CP2INT	6
	7FBCh, 7FBDh	PACT CP1 Edge	CP1 INT FLAG	CP1INT	7
	7FBEh, 7FBFh	PACT Default Timer Overflow	DEFTIM OVRFL INT FLAG	POVRFL INT	8
A/D	7FECh, 7FEDh	A/D Conversion Complete	AD INT FLAG	ADINT	1
Timer 2	7FEEh, 7FEFh	Timer 2 Overflow	T2 OVRFL INT FLAG	T2INT	1
		Timer 2 Compare 1	T2C1 INT FLAG		
		Timer 2 Compare 2	T2C2 INT FLAG		
		Timer 2 External Edge	T2EDGE INT FLAG		
		Timer 2 Input Capture 1	T2IC1 INT FLAG		
		Timer 2 Input Capture 2	T2IC2 INT FLAG		

† 1 is the highest priority.

Table 5–3. Interrupt Vector Sources (Concluded)

Module	Vector Address	Interrupt Source	Interrupt Flag	System Interrupt	Priority in Group†
SCI TX	7FF0h, 7FF1h	SCI TX Data Register Empty	TXRDY FLAG	TXINT	1
SCI RX	7FF2h, 7FF3h	SCI RX Data Register Full	RXRDY FLAG	RXINT	1
		SCI RX Break Detect	BRKDT FLAG		
Timer 1	7FF4h, 7FF5h	Timer 1 Overflow	T1 OVRFL INT FLAG	T1INT	1
		Timer 1 Compare 1	T1C1 INT FLAG		
		Timer 1 Compare 2	T1C2 INT FLAG		
		Timer 1 External Edge	T1EDGE INT FLAG		
		Timer 1 Input Capture 1	T1IC1 INT FLAG		
		Watchdog Overflow	WD OVRFL INT FLAG		
SPI	7FF6h, 7FF7h	SPI RX/TX Complete	SPI INT FLAG	SPIINT	1
External INT	7FF8h, 7FF9h	External INT3	INT3 FLAG	INT3	1
	7FFAh, 7FFBh	External INT2	INT2 FLAG	INT2	1
	7FFCh, 7FFDh	External INT1	INT1 FLAG	INT1	1
$\overline{\text{RESET}}$	7FFEh, 7FFFh	External $\overline{\text{RESET}}$	COLD START	$\overline{\text{RESET}}$	1
		Watchdog Overflow	WD OVRFL INT FLAG		
		Oscillator Fault Detect	OCS FLT FLAG		

† 1 is the highest priority.

The application program can individually enable or disable all of the interrupt sources via local interrupt enable control bits in the associated peripheral file. Also, software can read each interrupt source's flag bit in order to determine which interrupt source generated the system interrupt.

The processor acknowledges an interrupt if its flag bit equals 1 and the interrupt is enabled. To avoid immediately re-entering the same interrupt service routine, the interrupt service routine must clear all appropriate flag bits before leaving the routine.

Interrupts are sampled and arbitrated by the CPU during every opcode fetch. If one or more requests are pending (and the appropriate enable bits are set in the status register for maskable interrupts), then at the normal completion of the opcode fetch, the interrupt context switch begins. The new opcode is discarded, and the program counter is rewound to point to the discarded instruction. Moreover, at the completion of the interrupt service routine, the discarded instruction is fetched again.

The context switch routine proceeds as follows:

1) Increment the stack pointer (SP) and store the contents of the status register (ST) at the location pointed to by the SP.
2) Set the ST to 00h (disables further interrupt recognition).
3) Obtain the identity of the interrupting peripheral.
4) Rewind the program counter (PC) to point to the aborted opcode.
5) Increment SP and store the original PC high byte (PCH) at the location pointed to by the SP.
6) Get the interrupt-service-routine address (low byte) and store it in the PC low byte (PCL).
7) Increment SP and store the original PCL at the location pointed to by SP.
8) Get address (high byte) of interrupt service routine and store it in the PCH.
9) Resume instruction execution with the new PC contents.

A minimum of fifteen cycles is required from the time that an interrupt is triggered to the reading of the first instruction of the interrupt service routine. The moment at which the interrupt is asserted and the place in the instruction at which the interrupt is asserted depend on the instruction in progress. The worst case happens if the interrupt occurs near the start of a divide instruction; the processor may require up to 78 clock cycles to enter the interrupt service routine. If wait states are needed, the appropriate number of cycles must be added. Also, an external interrupt (INT1, INT2, or INT3) requires two extra clock cycles to synchronize before the processor can detect it.

5.1.2 External Interrupts

External pins INT1, INT2 and INT3 allow external devices to interrupt the program and enter a specific interrupt service routine. The INT1, INT2, and INT3 control registers in peripheral file frame 1 govern the software configuration of the external interrupts. Figure 5–2 shows these registers.

Figure 5–2. Peripheral File Frame 1: External Interrupt Control Registers

Designation	Address	PF	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INT1	1017h	P017	INT1 FLAG	INT1 PIN DATA	—	—	—	INT1 POLARITY	INT1 PRIORITY	INT1 ENABLE
INT2	1018h	P018	INT2 FLAG	INT2 PIN DATA	—	INT2 DATA DIR	INT2 DATA OUT	INT2 POLARITY	INT2 PRIORITY	INT2 ENABLE
INT3	1019h	P019	INT3 FLAG	INT3 PIN DATA	—	INT3 DATA DIR	INT3 DATA OUT	INT3 POLARITY	INT3 PRIORITY	INT3 ENABLE

The software can configure each external interrupt individually, through the interrupt polarity bits, to trigger on either a rising edge or a falling edge. If the interrupt function is not required, the software can configure external interrupts INT2 and INT3 to be general-purpose input/output pins, and INT1 to be an input pin.

INT1 can be programmed to be a maskable or nonmaskable interrupt. When INT1 is nonmaskable, it cannot be masked by the individual or global mask bits. Remember that the INT1 NMI bit (SCCR2.1) is protected during nonprivilege operation and should be configured during the initialization sequence following reset (see INT1 NMI bit description in subsection 4.3.3, page 4-14).

The nonmaskable interrupt is used for events that you want to respond to immediately. For example, this event could be derived from monitoring the power supply and saving important data to EEPROM whenever a brownout/power loss condition occurs.

Notes:

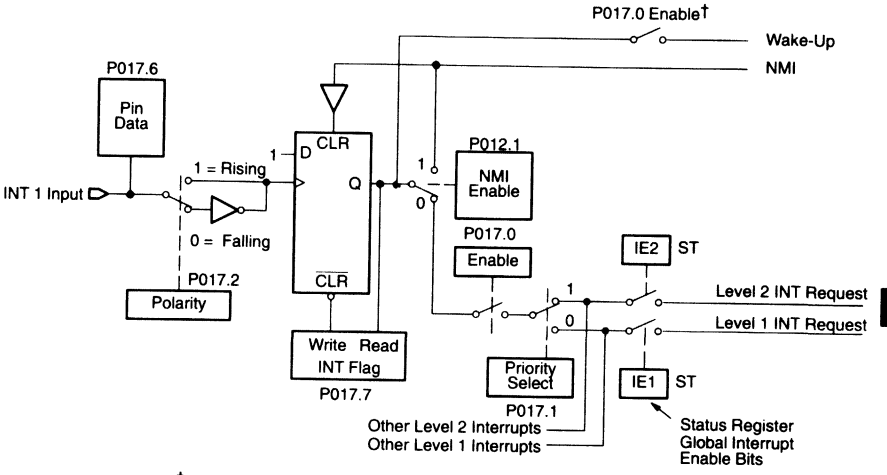
- When an NMI is used, the interrupt is taken on *every* active edge of the INT1 pin. This pin should be debounced to avoid multiple interrupts that could cause the stack to overflow. Once the stack has overflowed, the program may not operate as expected.
- To provide a method of exiting low-power modes, INT1 is automatically enabled as a nonmaskable interrupt (NMI) during low-power modes when the hard watchdog mode is selected. This means that the NMI is always generated on the interrupt enable flags and the values of these status bits: INT1 PRIORITY bit (INT1.1), INT1 ENABLE bit (INT1.0), INT1 NMI bit (SCCR2.1), and global interrupt enable flags of the status register (IE1 and IE2). See subsection 7.7.2.1, page 7-21, for more information.

The application program must configure the following bits for each interrupt to function correctly (refer to Figure 5–3 and Figure 5–4 on the following page).

- The INT PRIORITY bit configures the interrupt as either a level 1 or a level 2 interrupt.
- The INT POLARITY bit selects the trigger as either a falling edge or a rising edge.
- The INT ENABLE bit allows the request to be transmitted to the CPU if either the IE1 or IE2 enable bit, whichever is appropriate, is enabled.
- The INT FLAG indicates that the selected edge (rising or falling) has occurred. If the enables are set, an interrupt is requested. This bit remains a 1 until the software or a RESET clears it. The INT FLAG bit is useful for programs that poll the interrupt flag instead of generating a system interrupt.
- The INT PIN DATA bit shows the condition presently on the interrupt pin.
- On interrupts 2 and 3, the INT DATA DIR determines whether the pin functions as a general-purpose output pin or as an input/interrupt pin.
- If you select the general-purpose output function for a pin, then the value written by software to the INT DATA OUT bit determines the value of the output.

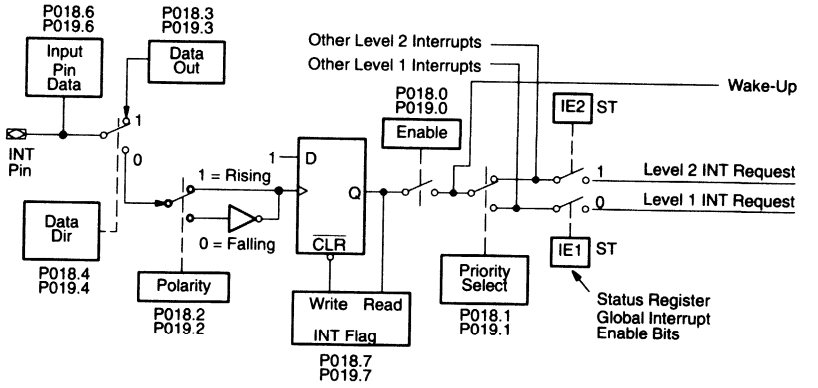
All external interrupts can bring the processor out of both the halt and the standby low-power modes if the interrupt enable and the interrupt level mask are enabled. Note that in halt mode, the interrupt is detected on the level and not the edge. For further information, refer to subsection 4.2.3, page 4-8.

Figure 5–3. Interrupt 1 Block Diagram



† This bit is ignored if you are using the hard watchdog option.

Figure 5–4. Interrupts 2 and 3 Block Diagram



5.2 Interrupt Control Registers

The interrupt control registers control the configuration of the external interrupts.

5.2.1 Interrupt 1 Control Register (INT1)

The INT1 register controls the interrupt configuration for the INT1 pin.

Interrupt 1 Control Register (INT1)
[Memory Address – 1017h]

Bit #	7	6	5	4	3	2	1	0
P017	INT1 FLAG	INT1 PIN DATA	—	—	—	INT1 POLARITY	INT1 PRIORITY	INT1 ENABLE
	RC–0	R–0				RW–0	RW–0	RW–0

R = Read, W = Write, C = Clear only, *–n* = Value of the bit after the register is reset

Bit 0 **INT1 ENABLE.** Interrupt 1 Enable.

When set, this bit enables the interrupts for the INT1 pin. This bit is ignored if INT1 NMI=1.

1 = Enables INT1 interrupts.
0 = Disables INT1 interrupts.

Bit 1 **INT1 PRIORITY.** Interrupt 1 Priority.

This bit determines the interrupt level of the INT1 pin—either a high, level-1 interrupt or a low, level-2 interrupt. This bit is ignored if INT1 NMI=1.

1 = Level 2 interrupt (low level).
0 = Level 1 interrupt (high level).

Bit 2 **INT1 POLARITY.** Interrupt 1 Polarity.

This bit determines whether INT1 triggers on a rising edge or on a falling edge.

1 = Triggers on a rising edge (low-to-high transition).
0 = Triggers on a falling edge (high-to-low transition).

Bits 3, 4, 5 **Reserved.** Read data is indeterminate.

Bit 6 **INT1 PIN DATA.** Interrupt 1 Pin Data.

This bit displays the current condition of the INT1 pin.

1 = High-level input voltage (V_{IH}) at the INT1 pin.
0 = Low-level input voltage (V_{IL}) at the INT1 pin.

Bit 7 **INT1 FLAG.** Interrupt 1 Flag.

When set, this bit indicates that the selected transition on INT1 has occurred. An interrupt can occur as long as this bit remains set; as a result, the application program must clear this bit during the interrupt handling routine. This bit will be set, even if the INT1 ENABLE bit is cleared. This flag will not be set if INT1 is configured as a nonmaskable interrupt (NMI).

1 = Transition detected.
0 = No transition.

5.2.2 Interrupt 2 Control Register (INT2)

The INT2 register controls the interrupt configuration for the INT2 pin.

Interrupt 2 Control Register (INT2)
[Memory Address – 1018h]

Bit #	7	6	5	4	3	2	1	0
P018	INT2 FLAG	INT2 PIN DATA	—	INT2 DATA DIR	INT2 DATA OUT	INT2 POLARITY	INT2 PRIORITY	INT2 ENABLE
	RC–0	R–0		RW–0	RW–0	RW–0	RW–0	RW–0

R = Read, W = Write, C = Clear only, *–n* = Value of the bit after the register is reset

Bit 0 **INT2 ENABLE.** Interrupt 2 Enable.

When set, this bit enables the interrupts for the INT2 pin.

1 = Enables INT2 interrupts.
0 = Disables INT2 interrupts.

Bit 1 **INT2 PRIORITY.** Interrupt 2 Priority.

This bit determines the interrupt level of the INT2 pin—either a high, level-1 interrupt or a low, level-2 interrupt.

1 = Level 2 interrupt (low level).
0 = Level 1 interrupt (high level).

Bit 2 **INT2 POLARITY.** Interrupt 2 Polarity.

This bit determines whether INT2 triggers on a rising edge or on a falling edge.

1 = Triggers on a rising edge (low-to-high transition).
0 = Triggers on a falling edge (high-to-low transition).

Bit 3 **INT2 DATA OUT.** Interrupt 2 Data Out.

If software configures the INT2 pin as an output pin (INT2 DATA DIR=1), then the value that the software writes to the INT2 DATA OUT bit determines the value of that output pin.

Bit 4 **INT2 DATA DIR.** Interrupt 2 Data Direction.

The INT2 pin can be configured as either an output pin or as an input/interrupt pin.

1 = INT2 pin is an output pin.
0 = INT2 pin is an input/interrupt pin.

Bit 5 **Reserved.** Read data is indeterminate.

Bit 6 **INT2 PIN DATA.** Interrupt 2 Pin Data.

This bit displays the current value of the INT2 pin.

1 = High-level input voltage (V_{IH}) at the INT2 pin.
0 = Low-level input voltage (V_{IL}) at the INT2 pin.

Bit 7 **INT2 FLAG.** Interrupt 2 Flag.

This bit indicates that the selected transition on INT2 has occurred. An interrupt can occur as long as this bit remains set; as a result, the program must clear this bit during the interrupt handling routine. This bit will be set, even if the INT2 ENABLE bit is cleared.

1 = Transition detected.
0 = No transition.

5.2.3 Interrupt 3 Control Register (INT3)

The INT3 register controls the interrupt configuration for the INT3 pin.

Interrupt 3 Control Register (INT3)
[Memory Address – 1019h]

Bit #	7	6	5	4	3	2	1	0
P019	INT3 FLAG	INT3 PIN DATA	—	INT3 DATA DIR	INT3 DATA OUT	INT3 POLARITY	INT3 PRIORITY	INT3 ENABLE
	RC–0	R–0		RW–0	RW–0	RW–0	RW–0	RW–0

R = Read, W = Write, C = Clear only, –*n* = Value of the bit after the register is reset

Bit 0 **INT3 ENABLE**. Interrupt 3 Enable.

This bit enables the interrupts for the INT3 pin.

1 = Enables INT3 interrupts.
0 = Disables INT3 interrupts.

Bit 1 **INT3 PRIORITY**. Interrupt 3 Priority.

This bit determines the interrupt level of the INT1 pin—either a high, level-1 interrupt or a low, level-2 interrupt.

1 = Level-2 interrupt (low level).
0 = Level-1 interrupt (high level).

Bit 2 **INT3 POLARITY**. Interrupt 3 Polarity.

This bit determines whether INT3 triggers on a rising edge or on a falling edge.

1 = Triggers on a rising edge (low-to-high transition).
0 = Triggers on a falling edge (high-to-low transition).

Bit 3 **INT3 DATA OUT**. Interrupt 3 Data Out.

If software configures the INT3 pin as an output pin (INT3 DATA DIR=1), then the value that the software writes to the INT3 DATA OUT bit determines the value of that output pin.

Bit 4 **INT3 DATA DIR**. Interrupt 3 Data Direction.

The INT3 pin can be configured as either an output pin or as an input/interrupt pin.

1 = INT3 pin is an output pin.
0 = INT3 pin is an input/interrupt pin.

Bit 5 Reserved. Read data is indeterminate.

Bit 6 **INT3 PIN DATA**. Interrupt 3 Pin Data.

This bit displays the current condition of the INT3 pin.

1 = High-level input voltage (V_{IH}) at the INT3 pin.
0 = Low-level input voltage (V_{IL}) at the INT3 pin.

Bit 7 **INT3 FLAG**. Interrupt 3 Flag.

This bit indicates that the selected transition on INT3 has occurred. An interrupt can occur as long as this bit remains set; as a result, the program must clear this bit during the interrupt handling routine. This bit will be set, even if the INT3 ENABLE bit is cleared.

1 = Transition detected.
0 = No transition.

5.3 Multiple Interrupt Servicing

When servicing an interrupt, the processor automatically clears the global interrupt enable bits IE1 and IE2 in the status register. This prevents all other interrupts from being recognized during the execution of the interrupt service routine. Once the service routine is completed by executing the RTI (return from interrupt) instruction, the old status register contents are popped from the stack. This returns the IE1 and IE2 to their original conditions and allows any pending interrupts to be recognized.

An interrupt service routine can allow nested interrupts by executing the EINT, EINTL, or EINTH instructions to set the global interrupt enable bits in the status register. This permits other interrupts to be recognized during the service routine execution. Since a nested interrupt service routine completes, it returns to the previous interrupt service routine when the RTI instruction executes. Too many nested interrupts could overflow the stack, causing program failure.

5.4 Resets

The TMS370 has three possible reset sources:

- A low input to the $\overline{\text{RESET}}$ pin
- A programmable watchdog timer timeout (described in Section 7.7 and Section 12.2)
- A programmable oscillator fault failure (described in subsection 4.1.2)

After a reset, the program can interrogate the status bits (shown in Table 5–4) to determine the source of the reset in order to take appropriate action. If none of the sources shown in Table 5–4 caused the reset, then the $\overline{\text{RESET}}$ pin was pulled low by external hardware or the PACT module watchdog.

Table 5–4. Reset Sources

Register	Address	PF	Bit #	Control Bit	Source of Reset
SCCR0	1010h	P010	7	COLD START	Cold or warm start reset
SCCR0	1010h	P010	4	OSC FLT FLAG	Oscillator out of range
T1CTL2	104Ah	P04A	5	WD OVRFL INT FLAG	Watchdog timer timeout

The $\overline{\text{RESET}}$ pin starts the hardware initialization and ensures an orderly software startup. The $\overline{\text{RESET}}$ pin is an input/output pin. A low-level pulse initiates the reset sequence. The processor may detect short reset pulses of a few nanoseconds, but a low level (active) of one cycle is necessary to guarantee that the device sees the reset signal. The microcontroller is held in reset until the $\overline{\text{RESET}}$ pin goes inactive (high). If the reset input signal remains low for less than eight system clock cycles, the processor holds the external $\overline{\text{RESET}}$ pin low for eight system clock cycles to reset external system components.

Recall that the basic operating mode, microcomputer or microprocessor, is determined by the voltage level applied to the MC pin when the $\overline{\text{RESET}}$ pin goes inactive (high). The $\overline{\text{RESET}}$ pin can be pulled low at any time during operation to start the reset sequence immediately.

The sequence of events during reset is as follows:

1) Initialize CPU registers: ST = 00h, SP = 01h.
2) Initialize registers A and B to 00h (no other RAM is changed).
3) Read the contents of 7FFFh and store in the PCL.
4) Read the contents of 7FFEh and store in the PCH.
5) Start program execution with an opcode fetch from the address pointed to by the PC.

The reset sequence takes 20 cycles from the time the reset pulse is released until the first opcode fetch in the microcomputer mode (22 cycles in the microprocessor mode) is begun.

When the watchdog overflow or the oscillator fault detection circuit generates a reset, the $\overline{\text{RESET}}$ pin is pulled low in order to reset other external components in the system.

During a reset, RAM contents (except for register A and register B) are unchanged, and the majority of the peripheral file bits are cleared to 0, with the exception of the control bits shown in Table 5–5.

5

Table 5–5. Control-Bit States Following Reset

Register	Control Bit	Power up Microcomputer	Warm Reset	
			Microcomputer	Microprocessor
SCCR0	µP/µC Mode MC PIN DATA COLD START OSC FLT FLAG	0 0 1 0	0 0 see Note 1 see Note 1	1 1 see Note 1 see Note 1
XPORT1 (see Note 2)	all 8 bits	0	0	1
XPORT2 (see Note 2)	all 8 bits	0	0	1
T1CTL2	WD OVRFL FLAG	0	see Note 1	see Note 1
TXCTL	TX EMPTY TXRDY	1 1	1 1	1 1
ADSTAT	AD READY	1	1	1
PACT	PACT TXRDY	1	1	1

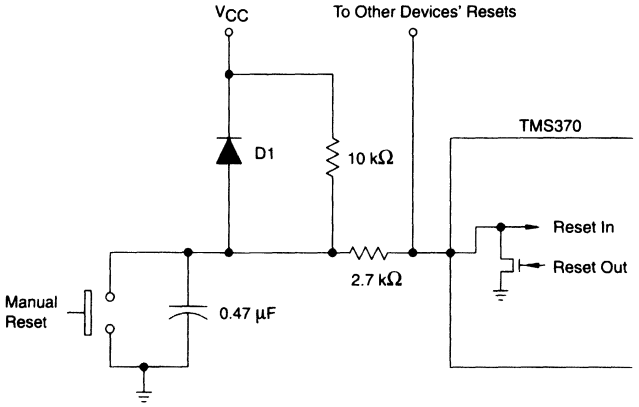
Notes: 1) State determined by cause of reset. See bit descriptions.

2) Refers to port control registers A, B, C, and D.

5.4.1 Simple Reset Circuitry

An application must activate the $\overline{\text{RESET}}$ pin at power-up with an external input to $\overline{\text{RESET}}$ or an RC power-up circuit. The $\overline{\text{RESET}}$ pin must be held low until the clock signal is valid and V_{CC} is within operating range. Figure 5–5 shows a simple reset circuit that will hold $\overline{\text{RESET}}$ low during power-up.

Figure 5–5. Typical Reset Circuit



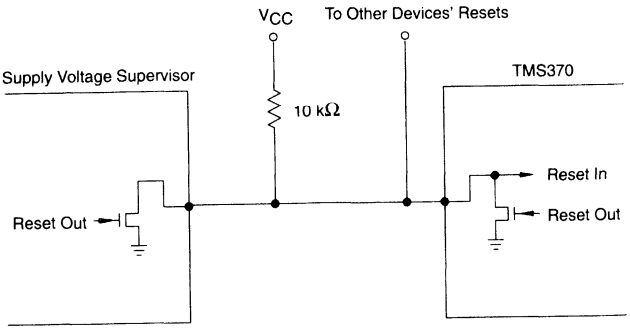
- The RC network of 10 kΩ and 0.47µF provides a power-up rise time. If this power-up rise time is not long enough (depending on the rise time of the power supply you are using), you can use a larger capacitor. However, replacing the 10 kΩ resistor with a larger resistor may cause the voltage at the $\overline{\text{RESET}}$ pin to be less than V_{IH}.
- The 2.7-kΩ resistor protects the $\overline{\text{RESET}}$ pin from the capacitor discharging directly into the pin when the pin is pulled low internally.
- The diode allows the capacitor to discharge quickly during a brownout or power-off situation.

Capacitors should not discharge directly into the $\overline{\text{RESET}}$ pin. Protect this pin from damage by using a resistor such as the 2.7 kΩ resistor shown in Figure 5–5.

5.4.2 Reset Circuitry With Low-Voltage Detection

It is recommended to have an asserted $\overline{\text{RESET}}$ during low-power or brownout conditions. In these instances, an active reset circuit with a low-voltage detection feature can be connected to the $\overline{\text{RESET}}$ pin. Figure 5–6 shows a typical circuit for using a supply voltage supervisor to assert $\overline{\text{RESET}}$.

Figure 5–6. Typical Reset Circuit Using a Supply Voltage Supervisor



The supply voltage supervisor must **not** cause a drive conflict with the TMS370 $\overline{\text{RESET}}$ pin. Moreover, the supply voltage supervisor should not drive $\overline{\text{RESET}}$ high since the TMS370 can drive the pin low. However, a pull-up resistor is needed.

To ensure the integrity of the contents of volatile memory (EEPROM, RAM), devices incorporating such memory require that the external $\overline{\text{RESET}}$ pin must be active (low) while V_{CC} is below its minimum specified operating level. Active reset circuitry prevents the EEPROM contents from being corrupted by improper instruction execution due to insufficient V_{CC} supply voltage and insures that the EEPROM write control register (DEECTL) powers up in the correct state when V_{CC} returns to its specified operating range.

To guarantee RAM data retention from 3.0 V to 4.5 V, $\overline{\text{RESET}}$ must be externally asserted and released only while V_{CC} is within the recommended operating range of 4.5 V to 5.5 V.

Chapter 6

EPROM and EEPROM Modules

This chapter discusses the architecture and programming of the data EEPROM modules of the TMS370 family and the program EPROM module of the TMS370C6xx and TMS370C7xx devices. Additional information about these modules is included in Chapter 14, *Design Aids*, and in Chapter 16, *Electrical Specifications and Timings*.

This chapter covers the following topics:

Topic **Page**

6.1 Data EEPROM Module

The TMS370 data EEPROM module contains a 256-byte array configured into eight 32-byte blocks. Devices can have multiple 256-byte arrays. Each additional array is also configured with eight 32-byte blocks. The first byte of each 256-byte array is the write protection register (WPR) for that array. This module also contains a voltage generator that provides a special precise programming voltage to the EEPROM array. This special voltage helps increase the reliability of the EEPROM and allows the TMS370 to program the EEPROM with a single V_{CC} source.

Reading the EEPROM module is identical to reading other internal memory and takes two system clock cycles. The CPU can fetch data and execute instructions from the EEPROM arrays. The data EEPROM module can be programmed on an array, a byte, or a single-bit basis. The memory can also be protected from inadvertent writing with a write-protect feature.

The data EEPROM is controlled by the data EEPROM control register (DEECTL) and by the WPR. The DEECTL register contains the bits needed to initiate and monitor EEPROM programming. The WPR of the given array contains the write protection bits for each 32-byte block of that data EEPROM array.

6.2 Data EEPROM Control Registers

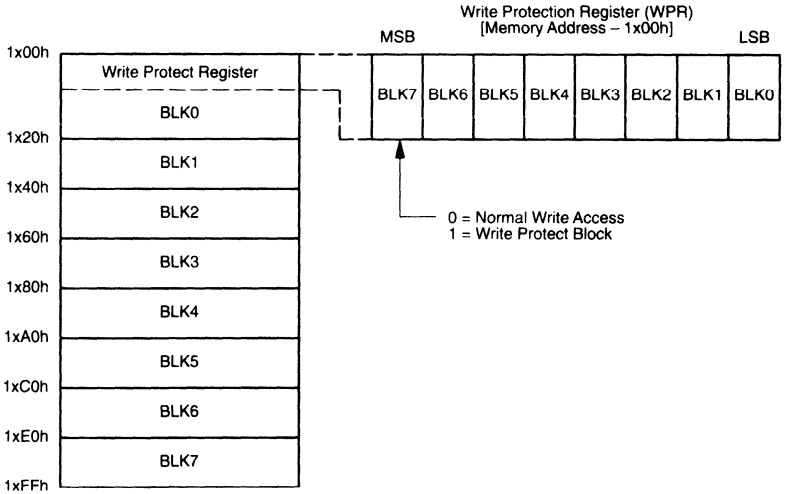
The data EEPROM can be write-protected, block by block (32 bytes), with the WPR(s). The DEECTL register determines the mode of programming and when programming is initiated.

6.2.1 Write Protection Register (WPR)

The WPR(s) provide write protection for the data EEPROM contents. The WPR is the first byte of each 256-byte data EEPROM array and is located in BLK0 of this array, generally at address 1x00h (where x is in the range 1 to F). This implies that the WPR, for the 256-byte array, is itself protected whenever BLK0 of that array is protected.

There are eight blocks of equal size in the data EEPROM array. Each bit in the WPR corresponds to one of the blocks. Setting a bit in this register to a 1 protects the corresponding block. Figure 6–1 shows the block protected by each bit.

Figure 6–1. Write Protection Bits in an EEPROM Array



Once block 0 is protected, the write-protection configuration cannot be altered unless write protection is overridden by placing the microcomputer into the write protection override mode (by applying 12 volts on the MC pin while the $\overline{\text{RESET}}$ pin is a logic 1). There is no write protection during a write protection override, and the WPR is considered a normal data location within the data EEPROM array during this time.

Example 6–1 illustrates one way to program the WPR. In this example, the program protects blocks 0 and 2. Also, assume that the WPR contains the value 00h before the example begins.

Example 6–1. Write Protection Register Programming

```
        DINT                    ;Disable interrupt
        MOV     #05,A           ;Protect bits for BLK0 and
                                ;BLK2
        MOV     A,1F00h         ;Set DEECTL to program 1's
        MOV     #3,P01A         ;Set W1W0 and EXE bits
        EINT                    ;Enable interrupt
        MOVW    #2778,R011      ;10 ms delay loop
DELAY   INCW    #-1,R011
        JC      DELAY
        MOV     #0,P01A         ;Clear W1W0 and EXE bits
        .
        .
        .
```


See Chapter 14 for more examples of programming the EEPROM module.

6.2.2 Data EEPROM Control Register (DEECTL)

The DEECTL register is located in the peripheral file at address P01A (101Ah). Data EEPROM programming is controlled through this register.

Data EEPROM Control Register (DEECTL)
[Memory Address – 101Ah]

Bit #	7	6	5	4	3	2	1	0
P01A	BUSY	—	—	—	—	AP	W1W0	EXE
	R–*					RW–0	RW–0	RW–0

R = Read, W = Write, –*n* = Value of the bit after the register is reset (–* = see the individual bit description)

Bit 0 **EXE**. Execute.

This bit initiates the write operation defined by the remaining control register bits. When cleared, this bit terminates a programming operation in progress. If the application program reads a data EEPROM location while the EXE bit is set, the processor reads the data being programmed into the EEPROM. If software attempts a write to the EEPROM while the EXE bit is set, the data byte is ignored.

0 = Inactive.
1 = Active.

Bit 1 **W1W0**. Write1/Write0.

This bit determines whether the ones or zeroes programming mode is to be used (see Section 6.3). This bit is write protected whenever the EXE bit is set.

0 = Write zeros.
1 = Write ones.

Bit 2 **AP**. Array Program.

Devices with a single 256-byte array:

In a single programming cycle, this bit programs the entire array with the value specified by the W1W0 bit (refer to Chapter 16 for timing). Blocks protected in the WPR register are *not* programmed. If BLK0 is unprotected and W1W0 is zero, this function clears the WPR; any array locations previously protected will lose their protection, but their contents are not altered during the current programming cycle.

Devices with multiple 256-byte arrays:

In a single programming cycle, this bit programs the entire multiple 256-byte array space with the value specified by the W1W0 bit. However, the device must be in the WPO mode for the multiple 256-byte arrays to be programmed. Moreover, there is no write protection during WPO mode; the WPR is considered a normal data location within the data EEPROM array during this time.

If the device is not in the WPO mode, the AP bit has no effect on the programming operation, and a single byte is programmed.

0 = Disables array programming.
1 = Enables array programming.

Bit 3–6 Reserved. Read data is indeterminate.

Bit 7 **BUSY**.

This bit is set during data EEPROM programming to indicate that an operation is in progress. Reading any location of the EEPROM during programming returns the data being programmed. In order to let the EEPROM voltages stabilize, the BUSY bit is set for 128 cycles:

- ❑ After a reset,
- ❑ After an exit from a low-power mode, and
- ❑ After programming the EEPROM.

If an attempt is made to access the EEPROM during this 128-cycle period, the data EEPROM holds execution of the processor by asserting the WAIT signal until the 128 cycles are complete.

0 = EEPROM array is ready for access.
1 = EEPROM array is not ready for access.

6.3 Programming the Data EEPROM

The procedure for programming the data EEPROM is controlled by the DEECTL (P01A) register and the associated array's WPR (1x00h) register. Individual bits are programmed to a 1 or 0 under the control of the W1W0 bit and the EXE bit in the DEECTL register.

- When the W1W0 bit is set, bit positions set to 1 in the data byte are programmed to 1 in the EEPROM byte; zeros are not changed.
- When the W1W0 bit is cleared, bit positions cleared to 0 in the data byte are programmed to 0 in the EEPROM byte; ones are not changed.

The EXE bit initiates EEPROM programming when set and disables programming when cleared. The WPR (1x00h) registers must have the corresponding protection bit cleared or be in the WPO mode to enable a data EEPROM write operation. (To enter the WPO mode, place 12 volts to the MC pin while the $\overline{\text{RESET}}$ pin is a logic 1.)

To load the data byte into the EEPROM module:

1) Perform a memory write operation to the EEPROM at the desired address. The data byte is latched in the module, ready for the Execute command (EXE bit=1).

   You must ensure that nonmaskable interrupt routines do not access the EEPROM between the EEPROM write instruction and the point when the EXE bit is set to 1, or data will be corrupted.

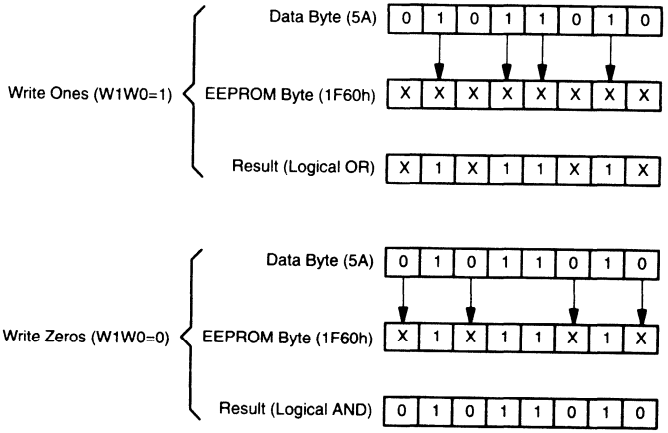
2) Following the memory cycle to the EEPROM address, write 03h (for W1W0=1) or 01h (for W1W0=0) to the DEECTL register to set the W1W0 and EXE bits. The W1W0 and the EXE bits must remain unchanged for the duration of the EEPROM timing parameter of $t_{W(PGM)B}$ to insure proper programming.

3) When the program time has elapsed, reset the EXE bit with another write operation to the DEECTL register.

If W1W0=1, the data that now resides in the programmed EEPROM location is the logical OR of the previous data stored in the location and the data written to the location. If W1W0=0, the data that now resides in the programmed EEPROM location is the logical AND of the previous data stored in the location and the data written to the location.

If a data value cannot be programmed by writing only ones or zeros, first perform the write-ones operation and follow it with a write zeros operation (or write zeros followed by write ones). Figure 6–2 illustrates these operations. In the programming operations, only the EEPROM bits that do not match the data bits are programmed. Therefore, there is no need to read the EEPROM value to determine what bits to program.

Figure 6–2. EEPROM Programming Example



The software should end the programming operation before entering a halt or standby state. When the microcomputer is in the halt or standby low-power mode, all operations of the data EEPROM module are stopped, and all DEECTL bits are cleared. Any EEPROM programming operation in progress is aborted when the halt is entered, and the data at the address being programmed is indeterminate.

The subroutine in Example 6–2 loads the data byte 5A into the data EEPROM location 1F60h. Figure 6–2 illustrates the result of this subroutine.

Example 6–2. Data EEPROM Programming

```
(a)             DINT                ;Disable all interrupts
(b)  DATA       MOV   #5Ah,A        ;Write 5A to location 1F60h
                MOV   A,1F60h
(c)             MOV   #03,P01A      ;Write Ones: W1W0=1, EXE=1
(d)             EINT                ;Enable all interrupts
(e)             MOVW  #2778,R017    ;Begin tW(PGM)B delay (10 ms)
(f)  DELAY1     INCW  #-1,R017      ;Decrement R017
(g)             JC    DELAY1        ;Jump to DELAY1 if R017>0
(h)             MOV   #0,P01A       ;Clear DEECTL. EXE=0
(i)             DINT                ;Disable all interrupts
(j)             MOV   #5Ah,A        ;Write 5A to location 1F60h
                MOV   A,1F60h
(k)             MOV   #01,P01A      ;Write zeros: W1W0=0 EXE=1
(l)             EINT                ;Enable all interrupts
(m)             MOVW  #2778,R017    ;Begin tW(PGM)B delay (10 ms)
(n)  DELAY2     INCW  #-1,R017      ;Decrement R017
(o)             JC    DELAY2        ;Jump to DELAY2 if R017>0
(p)             MOV   #0,P01A       ;Clear DEECTL. EXE=0
                .
                .
                .
```

- Disable all interrupts. When programming the data EEPROM, you must ensure that nonmaskable interrupt routines do not access the EEPROM between an EEPROM write instruction and the point when the EXE bit is set to 1 (a, i), or data will be corrupted.
- Load the value 5A into the data EEPROM address 1F60h (b).
- Begin a write ones programming sequence (c) by setting the W1W0 and EXE bits in the DEECTL register to a 1.
- Re-enable all interrupts (d).
- The programming delay parameter, $t_{W(PGM)B}$, (10 ms for this example—see Chapter 16 for required timing) is taken care of with a delay loop (f, g).
- The number of loops required is #2278 (e) and can be derived in the following manner:
  - Delay loop (f, g) requires 18 cycles to complete if a jump is taken.
  - An operating frequency of 20 MHz results in a system cycle time of 200 ns.
  - The number of loops required is calculated as follows:

    loop count = $t_{W(PGM)B}$ / (system cycle time × delay loop cycle count)

    loop count = 10 ms / (200 ns × 18) = 10 ms / 3.6 µs = 2778

    Note: Alternatively, a timer can be used for this delay.

- ❑ After the delay, clear the EXE bit (h), re-enable all interrupts, and continue the write zeros routine (i through p). The value 5A has now been programmed into location 1F60h of the data EEPROM.

Following an EEPROM write operation, the EEPROM voltage must stabilize before an EEPROM read operation is performed. The BUSY flag indicates the status of the EEPROM voltage. When BUSY is set, the EEPROM is not ready for a read operation. The BUSY flag is cleared to 0 by the EEPROM control logic when 128 system clock cycles have elapsed following the time that the EXE bit is cleared to 0.The BUSY bit remains set for 128 cycles:

- ❑ After a reset,
- ❑ After exit from a low-power mode, and
- ❑ After programming the EEPROM.

If an attempt is made to access the EEPROM during this 128 cycle period, the data EEPROM holds execution of the processor by asserting the WAIT signal until the 128 cycles are complete.

To prevent data corruption of the read or write EEPROM location, do not access EEPROM locations between writing data to the EEPROM address and setting the EXE bit to 1. In addition, you should disable interrupts during this time.

6.4 Program EPROM Modules

The program EPROM modules used in the TMS370 family replace the 4K-, 8K-, 16K-, or 32K-byte program ROM within the TMS370 families for system prototypes or small production runs.

These modules consist of a 8K-byte array and a 16K-byte array of EPROM at address locations 6000h through 7FFFh and 4000h through 7FFFh, respectively. The 32K-device is made up of two 16K-byte arrays; the first 16K-byte array is located at address locations 2000h through 5FFFh, and the second 16K-byte array is located at address locations 6000h through 9FFFh. The CPU can fetch data and execute instructions from these memory spaces.

The programming control register for the program EPROM (EPCTL) for the 8K-byte and 16K-byte EPROMs is located at address 101Ch (P01C). For the 32-K byte EPROM, the first 16-K byte array is controlled by the first EPCTL register, located at 101Ch (P01C); the second 16-K byte array is controlled by the second EPCTL register, located at 101Eh (P01E).

The CPU accesses the arrays with normal memory read cycles. Write cycles to the program EPROM require a special sequence of events. This sequence is described in subsection 6.4.2.

An external voltage supply is needed at the MC pin to provide the necessary V_{PP} for programming. Programming is controlled through the EPCTL register in the peripheral file.

Before programming (windowed versions), the EPROM module is erased by exposing the device through the transparent window to high-intensity ultraviolet light (wavelength 2537 angstroms). The recommended minimum exposure dose (UV intensity $\times$ exposure time) is 15 watt-seconds per square centimeter. A typical 12 milliwatt-per-square-centimeter, filterless UV lamp will erase the device in 21 minutes. The lamp should be located about 2.5 centimeters above the chip during erasure. After erasure, the entire array is at logic 1 state. A programmed 0 can be erased to 1 only by exposure to ultraviolet light. Note that normal ambient light contains the correct wavelength for erasure. Therefore, when using a programmed device, you should cover the window with an opaque label. All devices are erased to logical 1 at the factory.

Exposing the EPROM module to the ultraviolet light may also cause erasure in any EEPROM module. Any useful data stored in the EEPROM must be reprogrammed after exposure to UV light.

6.4.1 Program EPROM Control Register (EPCTL)

The EPCTL register at address 101Ch or 101Eh in the peripheral file controls programming of the program EPROM.

Program EPROM Control Register (EPCTL)
[Memory Address – 101Ch or 101Eh]

Bit #	7	6	5	4	3	2	1	0
P01C or P01E	BUSY	VPPS	—	—	—	—	W0	EXE
	R–0	RW–0					RW–0	RW–0

R = Read, W = Write, –*n* = Value of the bit after the register is reset

Bit 0 **EXE.** Execute.

This bit initiates the write operation defined by the other control register bits. When cleared, this bit terminates the operation.

0 = Inactive.
1 = Active.

Bit 1 **W0.** Write 0.

This bit determines whether the programming of the zero bits (in the byte written) is enabled.

0 = Enables programming of 0 bits.
1 = Disables programming of 0 bits.

Bit 2–5 **Reserved.** Read data is indeterminate.

Bit 6 **VPPS.**

This bit determines whether the programming voltage (V_{PP}) at the MC pin is connected to the EPROM module.

0 = Disables programming.
1 = Enables programming.

Bit 7 **BUSY.**

This bit reflects the value of the EXE bit.

6.4.2 Programming the Program EPROM

Programming 0 to the EPROM is controlled by the EPCTL register via the EXE bit and the VPPS bit.

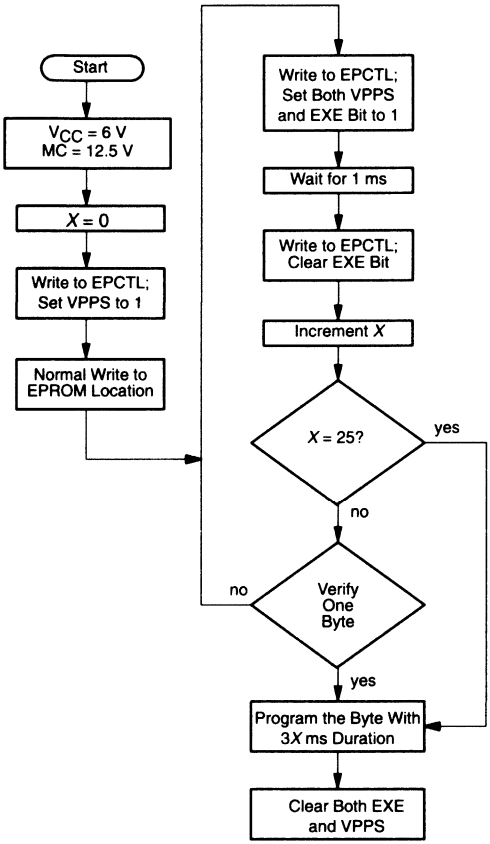
- ❑ The EXE bit initiates EPROM programming when set and disables programming when cleared.
- ❑ The VPPS bit connects the programming voltage (V_{PP}) at the MC pin to the EPROM module.

VPPS (EPCTL.6) and EXE (EPCTL.0) should be set separately, and the VPPS bit should be set at least two microseconds before the EXE bit is set. After programming, the application should wait for four microseconds before any read attempt is made.

The programming operation (see Figure 6–3) should be performed in this sequence:

1) Supply the programming voltage to the MC pin.
2) Write to the EPCTL to set the VPPS bit to 1.
3) Perform a normal memory write to the target EPROM location.
4) Write to the EPCTL to set the EXE bit to 1. (Wait at least two microseconds after step 2.)
5) Wait for program time to elapse (one millisecond).
6) Write to the EPCTL to clear the EXE bit (leave VPPS set to 1).
7) Read the byte being programmed; if correct data is not read, repeat steps 4 through 6 *X* times up to a maximum of 25.
8) Write to the EPCTL to set the EXE bit to 1 for final programming.
9) Wait for program time to elapse (3*X* milliseconds duration).
10) Write to the EPCTL to clear the EXE and VPPS bits.

Figure 6–3. EPROM Programming Operation



An external power supply at V_{PP}, I_{PP} (30 mA) is required for programming operation. Programming voltage (V_{PP}) is supplied via the MC pin. This also automatically puts the microcontroller in the write protection override (WPO) mode. Programming voltage can be applied via the MC pin anytime after reset and remain at V_{PP} after programming (after the EXE bit is cleared). Applying programming voltage while RESET is active will put the microcontroller in reserved mode, where programming operation is inhibited.

6.4.3 Write Protection of the Program EPROM

To override the EPROM write protection, the V_{PP} must be applied to the MC pin, *and* the VPPS bit (EPCTL.6) must be set. This dual requirement ensures that the program EPROM will not accidentally be overwritten during data EEPROM operations when V_{PP} is applied to the MC pin. Data EEPROM can be programmed when the VPPS bit is set.

Chapter 7

Timer 1 Module

This chapter discusses the architecture and programming of the timer 1 module and covers the following topics:

Topic **Page**

7.1 Timer 1 Overview

The timer 1 module of the TMS370 family provides enhanced timer resources to perform realtime system control. This module contains a general-purpose timer and a watchdog timer (WD). Both timers allow program selection of input clock sources (realtime, external event, or pulse accumulate) with multiple 16-bit registers (input capture and compare) for special timer function control. These timers provide the capabilities for:

System Requirements	Timer Resource
Realtime system control	Interval timers with interrupts
Input pulse-width measurement	Pulse accumulate or input capture functions
External event synchronization	Event count function
Timer output control	Compare function
PWM output control	PWM output function
System integrity	Watchdog function

7

7.1.1 Physical Description

The timer 1 module, shown in Figure 7–1, has the following components:

- **16-bit general-purpose timer** that provides capture, compare, and event functions.
  - The capture function latches the counter value on the occurrence of an external input.
  - The event function keeps a cumulative total of the transitions on the T1EVT pin.
  - The compare function triggers when the counter matches the contents of a compare register.
- **16-bit watchdog timer** that software can reconfigure as a simple counter/timer, an event counter, or a pulse accumulator if the watchdog feature is not needed.
- **Prescaler/clock source** that determines the independent clock sources for the general-purpose timer and for the watchdog timer.
- **Selectable edge-detection circuitry** that senses active transitions on the T1IC/CR pin.

❑ **Interrupts**

The module can be programmed to issue interrupts on the occurrence of:

- A capture
- A compare equal
- A counter overflow
- An external edge detect

❑ **I/O pins**

The timer 1 module has three I/O pins that can be dedicated for counter functions or as general-purpose I/O pins. They are:

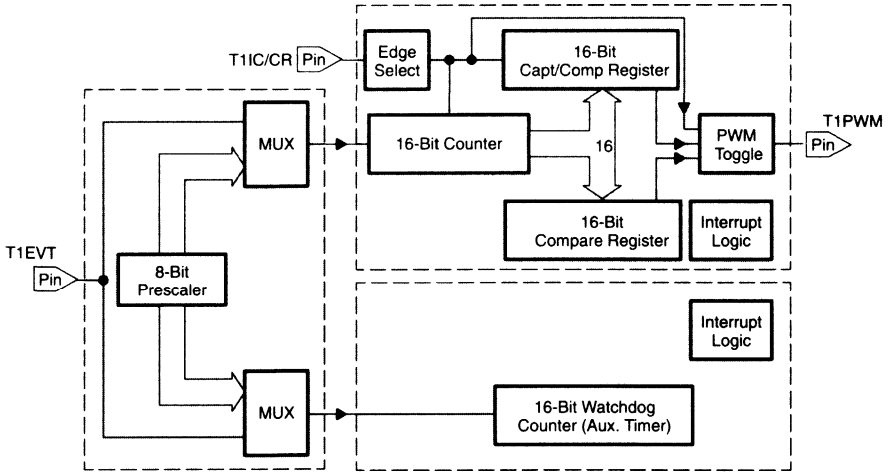
- T1EVT, an input to the event counter or the external clock source
- T1IC/CR, an input to the input capture, counter reset, or PWM circuit
- T1PWM, the pulse-width modulation output

Table 7–1 shows the definitions of these pins, according to operating mode.

Table 7–1. Timer 1 I/O Pin Definitions

Pin	Dual Compare Mode	Capture/Compare Mode
T1IC/CR	Counter reset input	Input capture 1 input
T1PWM	PWM output	PWM output
T1EVT	External event input or pulse accumulate input	External event input or pulse accumulate input

Figure 7–1. Timer 1 Block Diagram



7.1.2 Operating Modes

The general-purpose timer 1 module has two modes of operation:

- **Dual compare mode.** The timer is configured to provide two compare registers, external or software reset of the timer, internal or external clock source, and a programmable pulse-width modulated (PWM) output. The PWM output can be configured to toggle on selected events.
- **Capture/compare mode.** The timer is configured to provide one input capture register and one compare register for use with the general-purpose timer. The compare register can be used to provide periodic interrupts to the TMS370 CPU. The capture register can be configured to capture the current timer value upon either edge of an external input.

7.1.3 Control Registers

The timer 1 control registers are located at addresses 1040h to 104Fh and occupy peripheral file frame 4. The function of each location is shown in Table 7–2.

Table 7–2. Timer 1 and Watchdog Timer Memory Map

Peripheral File Location	Symbol	Name	Description
P040 P041	T1CNTR	T1 Counter — MSbyte T1 Counter — LSbyte	16-bit resettable counter.
P042 P043	T1C	Compare Register — MSbyte Compare Register — LSbyte	16-bit compare register.
P044 P045	T1CC	Capture/Compare Register — MSbyte Capture/Compare Register — LSbyte	16-bit capture/compare register.
P046 P047	WDCNTR	Watchdog Counter — MSbyte Watchdog Counter — LSbyte	16-bit watchdog counter.
P048	WDRST	Watchdog Reset Key	Resets the watchdog timer.
P049	T1CTL1	Timer 1 Control Register 1	Controls the prescaler inputs to the watchdog timer and to the general-purpose timer.
P04A	T1CTL2	Timer 1 Control Register 2	Controls the timer 1 and watchdog overflow interrupts and contains the timer 1 software reset bit.
P04B	T1CTL3	Timer 1 Control Register 3	Controls the edge-detect and compare interrupts.
P04C	T1CTL4	Timer 1 Control Register 4	Controls the mode of operation and various functions of the timer 1 input and output pins.
P04D	T1PC1	Timer 1 Port Control Register 1	Controls the I/O functions of the timer 1 module and T1EVT pin.
P04E	T1PC2	Timer 1 Port Control Register 2	Controls the I/O functions of the timer 1 module, T1IC/CR pin, and T1PWM pin.
P04F	T1PRI	Timer 1 Interrupt Priority Control Register	Controls the level of the timer 1 interrupt.

7.2 General-Purpose Timer Components

The general-purpose timer uses a 16-bit counter, a compare register, and a capture/compare register to provide event, compare, and capture functions.

7.2.1 16-Bit Resettable Counter

The free-running, 16-bit counter (T1CNTR) is clocked by the output of the prescaler/clock source. The program can access the 16-bit counter at P040 (T1 counter MSbyte) and P041 (T1 counter LSbyte) in peripheral file frame 4.

- ❑ During initialization, the counter is loaded with 0000h and begins its count.
- ❑ If the counter is not reset before reaching FFFFh, the counter rolls over to 0000h and continues counting. Upon counter rollover, the T1 OVRFL INT FLAG bit (T1CTL2.3) is set, and a timer interrupt is generated if the T1 OVRFL INT ENA bit (T1CTL2.4) is set.
- ❑ During counting, the counter can be reset to 0000h by:
  - A 1 written to the T1 SW RESET bit (T1CTL2.0),
  - A compare equal condition from the dedicated T1 compare function,
  - System reset, or
  - An external pulse on the T1IC/CR pin (dual compare mode only).

You can select the external-transition direction on the T1IC/CR pin, low-to-high or high-to-low, to reset the counter. To do this, use the T1EDGE POLARITY bit (T1CTL4.2).

Special circuitry prevents the contents of the T1CNTR register from changing in the middle of a 16-bit read operation. See the note in Section 7.9.

7.2.2 Compare Register

The compare register circuit consists of a 16-bit wide, read/write data register (T1C) and logic to compare the counter's current value with the value stored in the compare register. The program can access the 16-bit compare register at P042 (compare register MSbyte) and P043 (compare register LSbyte) in peripheral file frame 4.

When the counter's value matches the compare register value, then the circuit:

- Sets the T1C1 INT FLAG bit (T1CTL3.5) to 1,
- Clocks the output latch to toggle the T1PWM output pin if the T1C1 OUT ENA bit (T1CTL4.6) is set,
- Generates a timer 1 interrupt if the T1C1 INT ENA bit (T1CTL3.0) is set, and
- Resets the counter if the T1C1 RST ENA bit (T1CTL4.4) is set (dual compare mode only).

The compare register is initialized to 0000h following reset.

Special circuitry prevents the contents of the T1C register from changing in the middle of a 16-bit read operation. See the note in Section 7.9.

Note:

If the counter is programmed to reset when its value equals the contents of the compare register, the reset occurs on the following counter clock cycle (after prescale). However, the compare flag is set and the interrupt event occurs during the clock cycle that incremented the counter to equal the compare equal value. As a result, there could be a delay of up to 256 system clock cycles (depending on the prescale tap in use) from the time that the event is recognized by the program until the counter actually resets to zero. If the program writes to the compare register during this interval, the counter cannot be reset on the following counter clock cycle.

The compare register value required for a specific timing application can be calculated using the following formula:

$$Compare\ Value = \frac{t}{PS \times SYSCLK} - 1$$

where:

t = desired timer compare period (seconds)
SYSCLK = 4 /CLKIN (external clock frequency)
PS = 1, 4, 16, 64, or 256, depending on the prescale tap selected

Table 7–3 provides some sample compare register values to achieve various desired timings using a 20-MHz crystal.

Table 7–3. Timer 1 Compare Values: (CLKIN = 20 MHz)

Time		Prescale	T1 Compare Register Value (N)		% Error (See Note)
Seconds	mSeconds		Decimal	Hex	
0.0005	0.5	None	2499	009C3h	0.000
0.001	1	None	4999	01387h	0.000
0.002	2	None	9999	0270Fh	0.000
0.005	5	None	24999	061A7h	0.000
0.01	10	None	49999	0C34Fh	0.000
0.02	20	/4	24999	061A7h	0.000
0.05	50	/4	62499	0F423h	0.000
0.1	100	/16	31249	07A11h	0.000
0.2	200	/16	62499	0F423h	0.000
0.5	500	/64	39062	09896h	0.000
1.0	1000	/256	19530	04C4Ah	0.001
2.0	2000	/256	39061	09895h	0.001
3.0	3000	/256	58593	0E4E1h	0.001

Note: % error induced by the timer 1 formula. This error margin will vary, depending on the desired timer compare period and the minimum timer resolution (PS × SYSCLK).

7.2.3 Capture/Compare Register

The 16-bit wide capture/compare register (T1CC) can serve one of two functions, depending on the operating mode. The T1CC register is located at P044 (capture/compare register MSbyte) and P045 (capture/compare register LSbyte) in peripheral file frame 4.

Special circuitry prevents the contents of the T1CC register from changing in the middle of a 16-bit read operation. See the note in Section 7.9.

Dual Compare Mode

In the dual compare mode, the T1CC register acts as a read/write compare register. It functions exactly like the compare register described in subsection 7.2.2 except that T1CC **cannot** reset the counter.

When the counter value matches the capture/compare register value, then the circuit:

- ❑ Sets the T1C2 INT FLAG bit (T1CTL3.6) to 1,
- ❑ Clocks the output latch to toggle the T1PWM output pin if the T1C2 OUT ENA bit (T1CTL4.5) is set, and
- ❑ Generates a timer 1 interrupt if the T1C2 INT ENA bit (T1CTL3.1) is set.

Capture/Compare Mode

In the capture/compare mode, the edge detection signal captures the current counter content, loads it into the T1CC register, and sets the T1EDGE INT FLAG bit (T1CTL3.7).

7.3 Operating Modes of the General-Purpose Timer

The operating mode of the timer 1 general-purpose timer determines whether the capture/compare register functions as a capture register in the capture/compare mode or as a compare register in the dual compare mode. The T1 MODE bit (T1CTL4.7) selects the mode as follows:

T1 MODE = 0 — dual compare mode
T1 MODE = 1 — capture/compare mode

7.3.1 Dual Compare Mode

The dual compare mode provides the following:

- A 16-bit compare register (called compare 1)
- A 16-bit capture/compare register that acts as a compare register (called compare 2)
- A 16-bit external, resettable counter
- A timer output pin

These components allow the timer to act as an interval timer, a PWM output, simple output toggle, or other timer functions. The dual compare mode is shown in Figure 7–2.

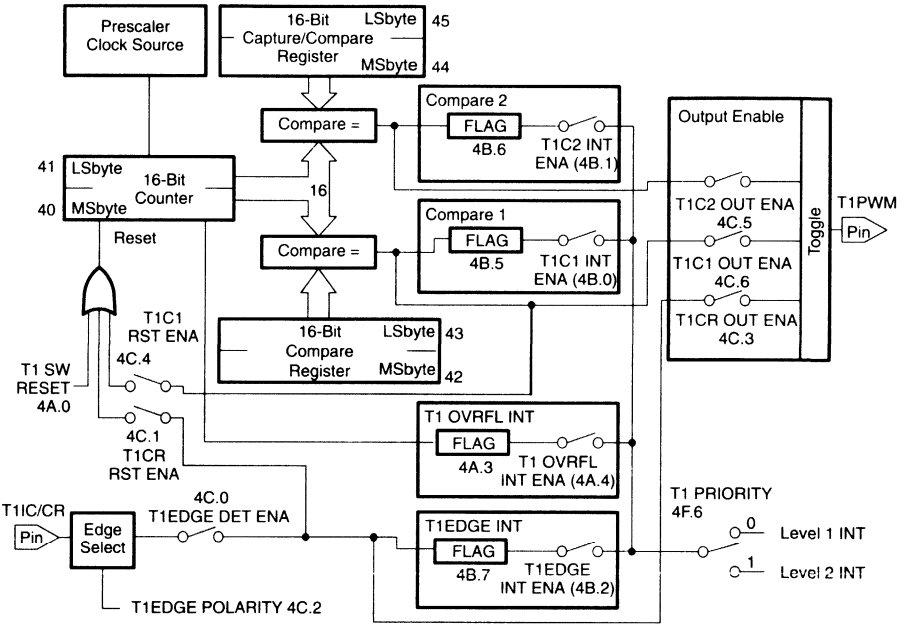
The dual compare mode continuously compares the contents of the two compare registers to the current value of the 16-bit counter.

- If the compare 1 register equals the counter, then the circuit:
  - Sets the T1C1 INT FLAG bit (T1CTL3.5) to 1,
  - Clocks the output latch to toggle the T1PWM output pin if the T1C1 OUT ENA bit (T1CTL4.6) is set,
  - Generates a timer 1 interrupt if the T1C1 INT ENA bit (T1CTL3.0) is set, and
  - Initiates a counter reset if the T1C1 RST ENA bit (T1CTL4.4) is set.

  Additionally, you can program an interval timer function by using the compare equal condition to generate a system interrupt combined with the counter reset function.
- If the compare 2 register equals the counter, then the circuit:
  - Sets the T1C2 INT FLAG bit (T1CTL3.6) to 1,
  - Clocks the output latch to toggle the T1PWM output pin if the T1C2 OUT ENA bit (T1CTL4.5) is set, and
  - Generates a timer 1 interrupt if the T1C2 INT ENA bit (T1CTL3.1) is set.

  The compare 2 register can be used as an additional system timing function.

Figure 7–2. Dual Compare Mode



Note: The bit locations in this figure are shown in following format: *xx.n*. The *xx* is the hexadecimal address of the peripheral register that contains the bit, and *n* is the bit number (7 = MSB, 0 = LSB). Register locations are designated by *xx*, the hexadecimal address of the peripheral file.

7.3.1.1 PWM Applications

Either compare register can be used to toggle the T1PWM output pin when a compare-equal condition occurs. Using both compare registers to control the T1PWM pin allows direct PWM generation with minimal CPU software overhead.

In typical PWM applications, the compare registers are loaded as follows:

- The compare 1 register is loaded with the periodic interval and configured to allow a counter reset on a compare-equal condition.
- The compare 2 (capture/compare) register is loaded with the pulse width to be generated within that interval. The program pulse width can be changed by the application program during the timer operation to alter the PWM output. For high-speed control applications, a minimum pulse width of 200 ns and a period as low as 400 ns can be maintained when a CLKIN of 20 MHz is used.

The PWM output can be used to support time-critical control applications. In these applications, an external input (T1IC/CR) is typically used to:

- ❑ Reset the counter,
- ❑ Generate a timer interrupt, and
- ❑ Toggle the T1PWM pin to start the PWM output.

The compare function then toggles the output after the programmed pulse width has elapsed.

7.3.1.2 Input Edge Detect

The input edge detect function is enabled under program control by the T1EDGE DET ENA bit (T1CTL4.0); upon the next occurrence of the selected edge transition:

- ❑ The T1EDGE INT FLAG bit (T1CTL3.7) is set,
- ❑ A timer interrupt is generated (if T1EDGE INT ENA = 1), and
- ❑ The T1PWM output pin is toggled (if T1CR OUT ENA = 1).

The T1EDGE POLARITY bit (T1CTL4.2) selects the active input transition. In the dual compare mode, the edge detect function must be re-enabled after each valid edge detect.

7.3.1.3 Clock Input

The clock input to the 16-bit counter (T1CNTR) is either the internal system clock, with or without prescale, or the external clock (T1EVT). The clock pulse to the counter is always synchronized with the system clock.

The counter (T1CNTR) is free-running except when it receives a reset pulse from one of the following sources:

- ❑ A 1 written to the T1 SW RESET (T1CTL2.0) bit,
- ❑ A compare equal condition from the dedicated T1 compare function,
- ❑ A system reset, or
- ❑ An external pulse on the T1IC/CR pin (dual compare mode).

The counter rolls over to 0000h if not reset before a count of FFFFh. When this rollover occurs, the counter sets the T1 OVRFL INT FLAG (T1CTL2.3), generates an interrupt if the T1 OVRFL INT ENA bit (T1CTL2.4) is set, and continues counting.

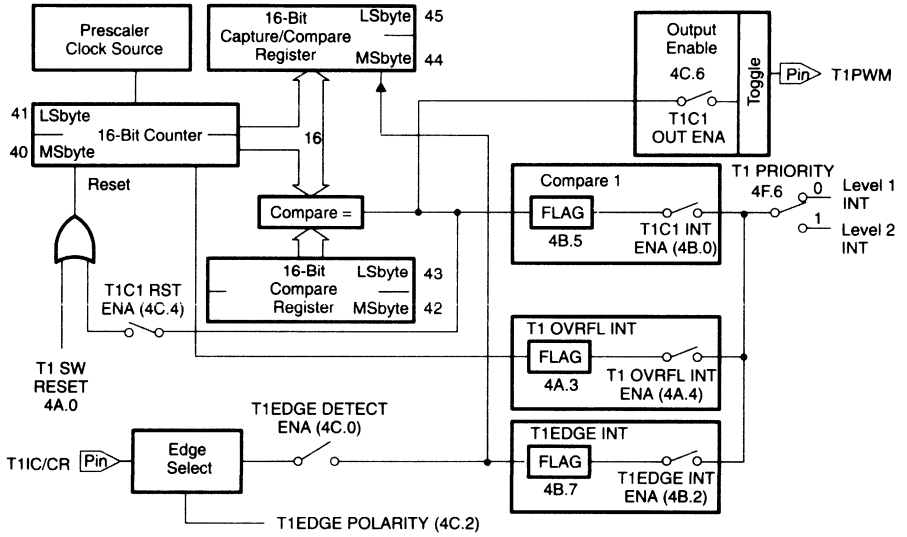
7.3.2 Capture/Compare Mode

In the capture/compare mode, timer 1 provides:

- A 16-bit input capture register for external timing and pulse-width measurement, and
- A 16-bit compare register for use as a programmable interval timer. This register functions the same as compare 1 does in the dual compare mode described in subsection 7.3.1, including the ability to toggle the PWM pin.

The capture/compare mode is shown in Figure 7–3.

Figure 7–3. Capture/Compare Mode



Note: The bit locations in this figure are shown in following format: *xx.n*. The *xx* is the hexadecimal address of the peripheral file register that contains the bit, and *n* is the bit number (7 = MSB, 0 = LSB). Register locations are designated by *xx*, the hexadecimal address of the peripheral file.

On the occurrence of valid input on the T1IC/CR pin:

- The current counter value is loaded into the 16-bit input capture register,
- The T1EDGE INT FLAG bit (T1CTL3.7) is set, and
- If T1EDGE INT ENA bit (T1CTL3.2) is set, a timer interrupt is generated.

The input detect function is enabled by the T1EDGE DET ENA bit (T1CTL4.0), with the T1EDGE POLARITY bit (T1CTL4.2) selecting the active input transition. In the capture/compare mode, the edge detect function, once enabled, remains enabled following a valid edge detect.

7.4 Edge-Detection Circuitry

The edge detection circuitry senses active transitions on the timer 1 input/capture/counter reset pin (T1IC/CR). The T1EDGE POLARITY bit (T1CTL4.2) determines whether the active transition is low-to-high or high-to-low. The module sets the T1EDGE INT FLAG (T1CTL3.7) when an active transition is detected. The program must reset this flag.

Dual Compare Mode

In this mode, the program must set the T1EDGE DET ENA bit (T1CTL4.0) to re-enable the circuit after each edge detection. Writing a 1 to this bit enables the detect circuit to look for the next correct level transition. After this active transition occurs, the T1EDGE DET ENA bit is cleared.

When the edge detection circuit is enabled and detects the appropriate edge transition, the T1EDGE INT FLAG bit (T1CTL3.7) is set.

When the T1CR RST ENA bit (T1CTL4.1) is set, the selected edge resets the counter. If the T1CR OUT ENA bit (T1CTL4.3) is set, the selected edge toggles the T1PWM output latch.

The T1EDGE POLARITY bit (T1CTL4.2) determines which edge polarity (rising or falling) is detected.

Capture/Compare Mode

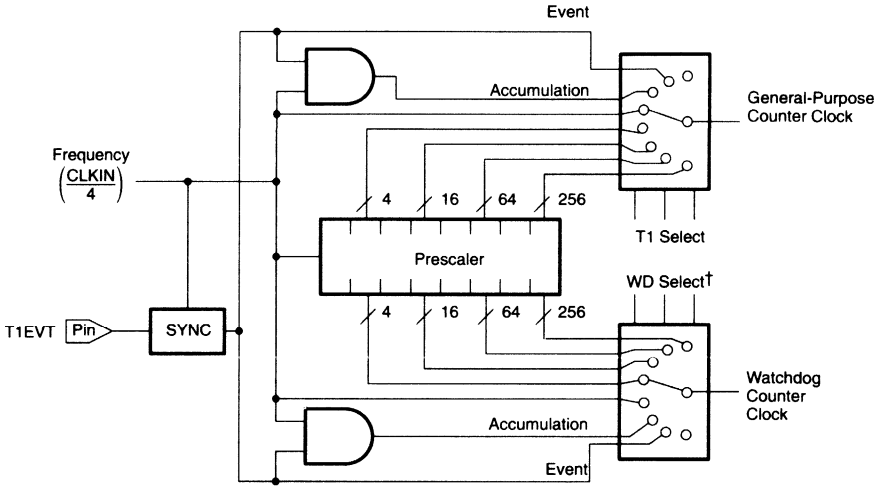
When the appropriate (rising or falling) transition is detected, the edge detection circuit signals the capture register to load the current counter value if the T1 EDGE DET ENA bit is set. The T1EDGE POLARITY bit determines which edge of the signal on the T1IC/CR pin to detect.

The input detect function is enabled by the T1EDGE DET ENA bit, with T1EDGE POLARITY selecting the active input transition. In the capture/compare mode, the edge detect function, once enabled, remains enabled following a valid edge detect.

7.5 Clock Prescaler/External Clock Source

A prescaler is a circuit that slows the rate of a clocking source to a counter. This block, as illustrated in Figure 7–4, allows the selection of the clock inputs (sources) to the general-purpose counter and to the watchdog counter independently. Each counter has three bits in the T1CTL1 register (see subsection 7.9.1, page 7-27) that determine whether the counter is clocked by one of the prescaled system clock values or by the external clock source (T1EVT).

Figure 7–4. Timer 1 System Clock Prescaler



† For the hard watchdog configuration of the mask-ROM device, the clock source comes only from one of the four taps from the prescaler that provide a system clock divided by 4, 16, 64, or 256.

The counter clock sources can be any of the following:

- ❑ System clock with no prescale
- ❑ No clock (the counter is stopped)
- ❑ External source synchronized with the system clock (event counter operation
- ❑ System clock while the external input is high (pulse accumulation)
- ❑ One of four taps from the prescaler that provide a system clock divided by 4, 16, 64, or 256

The external clock input to the module (T1EVT) must not exceed CLKIN/8. If the application does not require the external clock, the T1EVT pin can be reconfigured as a digital I/O pin.

The event input is not routed through the prescaler, so the timer 1 module can use different taps of the prescaler for timer 1 and the watchdog timer.

The maximum counter duration when the internal clock is used is determined by the internal system clock time (SYSCLK) and the prescale tap. These relationships are shown below:

$$\begin{aligned}\text{Maximum Counter Duration (seconds)} &= 2^{16} \times PS \times SYSCLK\\ \text{Counter Resolution} &= PS \times SYSCLK\\ \text{where: } SYSCLK &= 4/CLKIN\\ PS &= 1 \text{ for no prescale}\\ &= 4 \text{ for divide by 4}\\ &= 16 \text{ for divide by 16}\\ &= 64 \text{ for divide by 64}\\ &= 256 \text{ for divide by 256}\end{aligned}$$

Table 7–4 gives the real-time counter overflow rates for various crystal and prescaler values.

Software can configure the overflow rates for the watchdog counter as shown in Table 7–4 or as the value shown divided by two if the WD OVRFL TAP SEL bit (T1CTL1.7) is set (see Section 7.7). This bit configures the watchdog counter as either a 15-bit counter when set or a 16-bit counter when cleared.

7

Table 7–4. Counter Overflow Rates

				Crystal Oscillator Frequency (MHz)			
				2.0	4.0	10	20
Select 2	Select 1	Select 0	Divide By	System Clock Period (ns) 2000	1000	400	200
0	0	0	2^{16}	0.131†	0.066	0.026	0.013
0	0	1	(P.A.)	‡	‡	‡	‡
0	1	0	(Event)	‡	‡	‡	‡
0	1	1	(Stop)	‡	‡	‡	‡
1	0	0	2^{18}	0.524	0.262	0.105	0.052
1	0	1	2^{20}	2.10	1.05	0.419	0.210
1	1	0	2^{22}	8.39	4.19	1.68	0.839
1	1	1	2^{24}	33.6	16.8	6.71	3.355

† Time is given in seconds.
‡ Not applicable.

7.5.1 Event Counter Mode

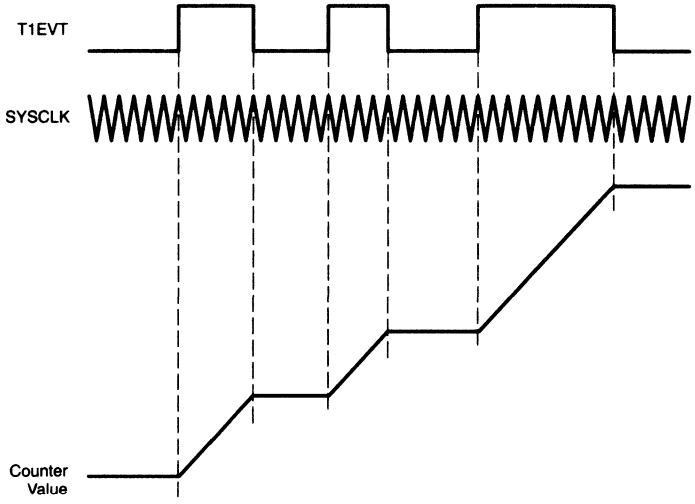
When you use the event counter clock source, the 16-bit counter is programmable as a 16-bit event counter. An external high-to-low transition on the T1EVT pin provides the clock for the internal timer.

7.5.2 Pulse Accumulator Mode

When you use the pulse accumulator clock source, the 16-bit counter is programmable as a 16-bit pulse accumulator. An external input on the T1EVT pin is used to gate the internal system clock to the internal timers. While T1EVT input is logic one (high), the timer is clocked at the system clock rate and counts system clock pulses until the T1EVT pin returns to logic zero.

The pulse accumulator mode keeps a cumulative count of SYSCLK pulses gated by the T1EVT signal as shown in Figure 7–5.

Figure 7–5. Pulse Accumulation



7.6 Interrupts

In dual compare mode, four separate events can generate an interrupt. These events are:

- ❑ Compare equal from compare register 2 if the T1C2 INT ENA bit (T1CTL3.1) is set,
- ❑ Compare equal from compare register 1 if the T1C1 INT ENA bit (T1CTL3.0) is set,
- ❑ Counter overflow if the T1 OVRFL INT ENA bit (T1CTL2.4) is set, or
- ❑ Edge detect is set if the T1EDGE INT ENA bit (T1CTL3.2) is set.

In the capture/compare mode, three separate events can generate an interrupt. These events are:

- ❑ Compare equal if the T1C1 INT ENA bit (T1CTL3.0) is set,
- ❑ Counter overflow if the T1 OVRFL INT ENA bit (T1CTL2.4) is set, and
- ❑ Input capture acknowledge if the T1EDGE INT ENA bit (T1CTL3.2) is set.

7

Note:

All set and enabled interrupt flags must be cleared before the processor exits the T1 interrupt routine. If the flags are not reset, the processor will enter the T1 interrupt routine again before continuing with the mainstream program. If the flag bits are never reset, the program will lock.

7.7 Watchdog Timer

The watchdog (WD) timer can be configured as one of the three different mask options.

- A **standard watchdog** for ROMless, EPROM, and mask-ROM devices.
- A **hard watchdog** for mask-ROM devices (see note).
- A **simple counter** for mask-ROM devices (see note).

Note:

Two mask-ROM options provide an improvement to the watchdog counter circuitry: the hard watchdog and the simple counter. These mask options serve to permanently enable the watchdog counter reset ability at all times or to permanently disable the watchdog counter reset ability at all times.

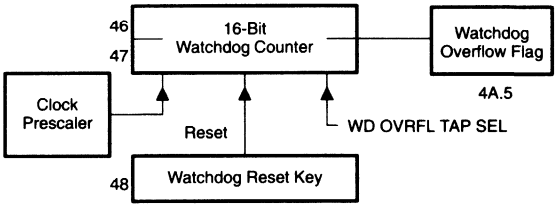
These additional mask-ROM options are available only on TMS370CxxxA devices. Refer to Section A.1, page A-2 for TMS370Cxxx differences.

ROM devices can be configured with any of the three options listed above. All EPROM devices will be configured with a standard watchdog option.

The watchdog timer, shown in Figure 7–6, consists of the following blocks:

- A 16-bit, resettable watchdog/event counter that provides up to 2^{24} clock cycles between counter overflows, depending on the prescaler tap used. The program can read the contents of this counter at locations P046 (watchdog counter MSbyte) and P047 (watchdog counter LSbyte) in the peripheral file.
- A prescaled clock input selection or external clock that functions the same as in the general-purpose timer (see Section 7.5).
- A watchdog reset key that is used to reset the watchdog counter (WDRST—P048).
- An overflow flag that is set whenever the watchdog counter overflows.

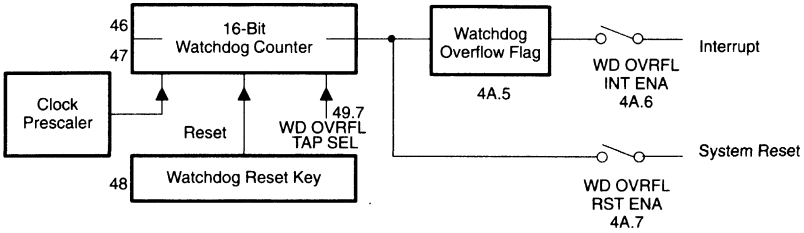
Figure 7–6. Watchdog Timer



7.7.1 Standard Watchdog Configuration (OTP/Reprogrammable EPROM Devices)

The standard watchdog is the only WD option for all EPROM devices that can be configured as either a watchdog or as a simple counter through setting or clearing the WD OVRFL RST ENA bit (T1CTL2.7) in the software. Figure 7–7 illustrates the block diagram of the standard watchdog.

Figure 7–7. Standard Watchdog Block Diagram



The standard watchdog can be configured in one of two modes: watchdog mode or nonwatchdog mode.

7

7.7.1.1 Watchdog Mode

In the watchdog mode (WD OVRFL RST ENA = 1), the WD timer generates a system reset if the counter overflows or if the WD counter is reinitialized by an incorrect value; a system reset pulls the $\overline{\text{RESET}}$ pin low for eight system clock cycles. The required reinitialization frequency is determined by the system clock frequency, the prescaler/clock source selected, and whether the WD OVRFL TAP SEL bit (T1CTL1.7) is set for 15- or 16-bit counter rollover.

The watchdog overflow times are the same as those given in Table 7–4, page 7-14, when the timer is configured as a 16-bit counter (WD OVRFL TAP SEL = 0). Divide the times in Table 7–4 in half when the timer is configured as a 15-bit counter (WD OVRFL TAP SEL =1).

With a 20-MHz clock, the watchdog-counter overflow times range from 6.55 ms to 3.35 seconds. These values are selected before the timer enters the watchdog mode because once the software enables the watchdog reset function (WD OVRFL RST ENA = 1), subsequent writes to these control bits are ignored. Writes to these watchdog control bits can occur only following a reset.

To reinitialize the watchdog counter, write a predefined value to the watchdog reset key (WDRST) located in the peripheral file at P048. The correct reset key alternates between 55h and AAh, beginning with 55h following the enable of the watchdog reset function. Writes of the correct value must occur before the timer overflow period.

A write of any value other than the correct predefined value to the watchdog reset key is interpreted as a lost program, and a system reset is initiated. A watchdog-counter overflow or incorrect reset key sets the WD OVRFL INT FLAG bit (T1CTL2.5) to 1. The program can read this flag after a reset to determine the source of the reset. Watchdog resets are not prevented when the flag is set.

Note:

A standard watchdog is disabled in low-power mode (see Section 7.8).

The routine in Example 7–1 initializes the watchdog mode in the standard watchdog to generate a system reset when the counter overflows. The watchdog counter is set to 16 bits in length, and the full 8-bit prescale tap is used.

Example 7–1. Standard Watchdog Initialization

```
.
.
;Set up watchdog timer for a 24-bit countdown time.
;
OR     #70h,P049 ;Set the watchdog overflow tap to 16 bits
                 ;and select the /256 prescale value
OR     #C0h,P04A ;Watchdog timer reset is enabled along
                 ; with clearing and enabling the
                 ; Watchdog timer interrupt.

.
.
The watchdog timer has now been initialized to cause a
system reset if the counter is not reset before reaching
FFFFh. To reset the counter, the code must write an alter-
nating 55h and AAh, starting with 55h, to the watchdog
timer reset key register (P048), e.g.:
.
.
MOV    #55h,P048    ;First write to WD RESET KEY
.
.
MOV    #0AAh,P048   ;Next write to WD RESET KEY
.
.
MOV    #55h,P048    ;Next write to WD RESET KEY
.
.
```

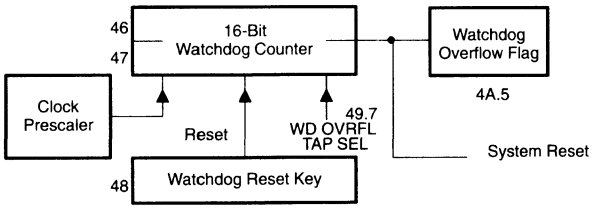
7.7.1.2 Nonwatchdog Mode

In the nonwatchdog mode (WD OVRFL RST ENA bit = 0), the watchdog counter can be used as an event counter, a pulse accumulator, or an interval timer. In this mode, the system reset function is disabled; to reinitialize the watchdog counter, write any value to the watchdog reset key (WDRST). In real-time control applications, the timer overflow rates are determined by the system clock frequency, the prescaler/clock source value selected, and the value of the WD OVRFL TAP SEL bit. If the WD counter is not reset before overflowing, the counter rolls over to either 0000h or 8000h, as determined by the WD OVRFL TAP SEL bit, and continues counting. Upon counter overflow, the WD OVRFL INT FLAG bit is set, and a timer interrupt is generated if the WD OVRFL INT ENA bit is set. Alternately, an external input on the T1EVT pin can be used with the watchdog timer to provide an additional 16-bit event counter or pulse accumulator.

7.7.2 Hard Watchdog Configuration (Mask-ROM Devices Only)

In the hard watchdog configuration, you can operate the watchdog timer only as a watchdog. Upon the power-up reset, the hard watchdog will be enabled, and the WD INPUT SELECT0–1 bits (T1CTL1.4–5) are cleared (system clock/4). Figure 7–8 is a block diagram of the hard watchdog.

Figure 7–8. Hard Watchdog Block Diagram



The hard watchdog provides additional system integrity. If the counter overflows or if the WD timer is reinitialized by an incorrect value, the hard watchdog generates a system reset, which pulls the $\overline{\text{RESET}}$ pin low for eight system clock cycles. The required reinitialization frequency is determined by the system clock frequency, WD INPUT SELECT0 and 1 (T1CTL1.4 and T1CTL1.5), the prescaler/clock source selected, and whether the WD OVRFL TAP SEL bit is set for 15- or 16-bit counter rollover. The WD INPUT SELECT2 bit (T1CTL1.6) is functionally interpreted as 1 at all times.

The WD INPUT SELECT0–1 bits and WD OVRFL TAP SEL bit can be modified at any time. Your program should reinitialize these bits after reset and periodically thereafter to ensure a corrected counter overflow rate and to protect against any hardware or software corruptions.

The watchdog timer is reinitialized by writing a predefined value to the watchdog reset key (WDRST) located in the peripheral file at P048. The correct reset key alternates between 55h and AAh, beginning with 55h following the enable of the watchdog reset function. Writes of the correct value must occur before the timer overflow period, or the watchdog will generate a reset.

A write to the watchdog reset key of any value other than the correct predefined value is interpreted as a lost program, and a system reset is initiated. A watchdog-counter overflow or incorrect reset key sets the WD OVRFL INT FLAG bit (T1CTL2.5) to 1. After a reset, the program can read this flag to determine the source of the reset. Watchdog resets are not prevented when the flag is set.

Note:

A hard watchdog is disabled in low-power mode (see Section 7.8).

7.7.2.1 INT1 Operation During an Inadvertent Low-Power Mode

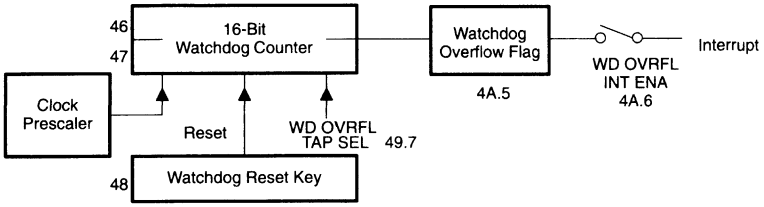
When the hard watchdog mask option is selected, INT1 is enabled as a nonmaskable interrupt (NMI) during low-power modes. This NMI is generated regardless of the interrupt enable flags and the values of the following status bits: INT1 PRIORITY bit (INT1.1), INT1 ENABLE bit (INT1.0), INT1 NMI bit (SCCR2.1), and the global interrupt enable flags in the status register (IE1 and IE2).

INT1 is configured as an NMI in the hard watchdog to provide a method of exiting a low-power mode. Normally, when the halt or standby mode is entered, the watchdog counter clock source is disabled, which disables the ability of the watchdog counter to generate a reset. Note that an active edge on the NMI INT1 pin will bring the device out of the low-power mode, and the watchdog counter will again be active. Additionally, if the halt or standby mode is entered while INT1 pin is active (low if the INT1 POLARITY bit is cleared to 0 or high if the INT1 POLARITY bit is set to 1), an NMI will be generated immediately.

7.7.3 Simple Counter Configuration (Mask-ROM Devices Only)

In the simple counter configuration, the watchdog timer can be used as an event counter, a pulse accumulator, or an interval timer (similar to the non-watchdog mode in the standard watchdog configuration). However, in this configuration, the system reset function of the watchdog timer is *disabled.* Figure 7–9 is a block diagram of a simple counter.

Figure 7–9. Simple Counter Block Diagram



To reinitialize the watchdog counter, write any value to the watchdog reset key (WDRST). The timer overflow rates are determined by the system clock frequency, the WD INPUT SELECT0–2 bits (T1CTL1.4–6), and the value of the WD OVRFL TAP SEL bit (T1CTL1.7). If the WD OVRFL RST ENA bit is set to 1, subsequent writes to WD INPUT SELECTs and WD OVRFL TAP SEL bits are ignored. Once the WD OVRFL RST ENA bit is set, these control bits can be changed only after a power-up reset.

7.7.4 Summary of Watchdog Options

Table 7–5 summarizes the features of each watchdog option and specifies the options available to mask-ROM and EPROM devices.

Table 7–5. Watchdog Option Summary

	Standard Watchdog		Hard Watchdog	Simple Counter
	Watchdog	**Nonwatchdog**		
WD OVRFL TAP SEL bit and WD INPUT SELECT0–2 bits	Once the WD OVRFL RST ENA is set, the values of these bits can be changed only after any system reset.	These bits can be changed at any time, as long as WD OVRFL RST ENA is not set.	The values of these WD bits can be changed at any time, even if WD OVRFL RST ENA bit is set. However, the WD INPUT SELECT2 bit is not available.	Once the WD OVRFL RST ENA is set, the values of these bits can be changed only after any system reset.
Generates an interrupt when the watchdog counter overflows?	No	Yes	No	Yes
Generates a system reset?	Yes	No	Yes	No
WD OVRFL RST ENA bit	Select to be a watchdog. If bit=1, watchdog counter does initiate a reset upon overflow. This bit is cleared by any system reset.	Select to be a non-watchdog. If bit = 0, watchdog counter does not initiate reset upon overflow.	This bit is ignored.	If bit=0, WD bits and WD OVRFL TAP SELECT are not locked. If bit=1, WD bits and WD OVRFL TAP SELECT are locked.
INT1 during low-power modes	Controlled by INT1 ENABLE bit (INT1.0) and INT1 NMI bit (SCCR2.1).	Controlled by INT1 ENABLE bit (INT1.0) and INT1 NMI bit (SCCR2.1).	Enabled as an NMI.	Controlled by INT1 ENABLE bit (INT1.0) and INT1 NMI bit (SCCR2.1).
Available Devices	All devices	All devices	Mask-ROM devices	Mask-ROM devices

7.8 Low-Power Modes

The timer 1 module supports low-power (powerdown) modes that aid in reducing power consumption during periods of inactivity. These modes are the halt and the standby modes. For more information on low-power modes, see Section 4.2, page 4-6.

7.8.1 Halt Mode

The halt mode is entered when the CPU executes an IDLE instruction while the HALT/STANDBY bit (SCCR2.7) and the PWRDWN/IDLE bit (SCCR2.6) are set (the SCCR2 register is described in detail in subsection 4.3.3, page 4-14). During the halt mode, all timer 1 module functions (including the watchdog timer) hold the prehalt status of all other storage elements.

The module holds the state of each external pin constant, regardless of whether the pins are used as timer pins or as dedicated I/O pins. That is, inputs remain inputs, output low levels remain low, and output high levels remain high.

7.8.2 Standby Mode

You can put the timer in standby mode by executing an IDLE instruction when the PWRDWN/IDLE (SCCR2.6) bit is set and the HALT/STANDBY bit (SCCR2.7) is cleared. During the standby mode, the watchdog counter clock input is halted while the rest of the timer 1 module remains fully functional.

7.9 Timer 1 Control Registers

Seven registers control the configuration of timer 1 global functions, prescale values, watchdog timing, optional uses for the associated I/O pins, and other counter functions (refer to Table 7–6). The bits that are shown in shaded boxes are privilege mode bits; that is, they can be written to only in the privilege mode.

Note:

Special circuitry prevents 16-bit registers from changing in the middle of a 16-bit read or write operation. When you read a 16-bit register, read the least significant byte (LSbyte) first to lock in the value, and then read the most significant byte (MSbyte). When you write to a 16-bit register, write the MSbyte first and then write the LSbyte. The register value does not change between reading and writing the bytes when they are done in this order. While you access a 16-bit register, do not read or write from a second 16-bit register within this module; if you do, the correct value for the first register's MSbyte will not be correct. The 16-bit read/write operation actually occurs when you access the LSbyte.

Read: **LSbyte then MSbyte**
Write: **MSbyte then LSbyte**

Table 7–6. Peripheral File Frame 4: Timer 1 Control Registers

Designation	ADDR	PF	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1CNTR	1040h	P040	Bit 15	T1 Counter MSbyte						Bit 8
T1CNTR	1041h	P041	Bit 7	T1 Counter LSbyte						Bit 0
T1C	1042h	P042	Bit 15	Compare Register MSbyte						Bit 8
T1C	1043h	P043	Bit 7	Compare Register LSbyte						Bit 0
T1CC	1044h	P044	Bit 15	Capture/Compare Register MSbyte						Bit 8
T1CC	1045h	P045	Bit 7	Capture/Compare Register LSbyte						Bit 0
WDCNTR	1046h	P046	Bit 15	Watchdog Counter MSbyte						Bit 8
WDCNTR	1047h	P047	Bit 7	Watchdog Counter LSbyte						Bit 0
WDRST	1048h	P048	Bit 7	Watchdog Reset Key						Bit 0
T1CTL1	1049h	P049	WD OVRFL TAP SEL † (RP–0)	WD INPUT SELECT2† (RP–0)	WD INPUT SELECT1† (RP–0)	WD INPUT SELECT0† (RP–0)	—	T1 INPUT SELECT2 (RW–0)	T1 INPUT SELECT1 (RW–0)	T1 INPUT SELECT0 (RW–0)
T1CTL2	104Ah	P04A	WD OVRFL RST ENA † (RS–0)	WD OVRFL INT ENA (RW–0)	WD OVRFL INT FLAG (RC–*)	T1 OVRFL INT ENA (RW–0)	T1 OVRFL INT FLAG (RC–0)	—	—	T1 SW RESET (S–0)
T1CTL3	104Bh	P04B	Dual Compare Mode							
			T1EDGE INT FLAG (RC–0)	T1C2 INT FLAG (RC–0)	T1C1 INT FLAG (RC–0)	—	—	T1EDGE INT ENA (RW–0)	T1C2 INT ENA (RW–0)	T1C1 INT ENA (RW–0)
			Capture / Compare Mode							
			T1EDGE INT FLAG (RC–0)	—	T1C1 INT FLAG (RC–0)	—	—	T1EDGE INT ENA (RW–0)	—	T1C1 INT ENA (RW–0)
T1CTL4	104Ch	P04C	Dual Compare Mode							
			T1 MODE = 0 (RW–0)	T1C1 OUT ENA (RW–0)	T1C2 OUT ENA (RW–0)	T1C1 RST ENA (RW–0)	T1CR OUT ENA (RW–0)	T1EDGE POLARITY (RW–0)	T1CR RST ENA (RW–0)	T1EDGE DET ENA (RW–0)
			Capture / Compare Mode							
			T1 MODE = 1 (RW–0)	T1C1 OUT ENA (RW–0)	—	T1C1 RST ENA (RW–0)	—	T1EDGE POLARITY (RW–0)	—	T1EDGE DET ENA (RW–0)
T1PC1	104Dh	P04D	—	—	—	—	T1EVT DATA IN (R–0)	T1EVT DATA OUT (RW–0)	T1EVT FUNCTION (RW–0)	T1EVT DATA DIR (RW–0)
T1PC2	104Eh	P04E	T1PWM DATA IN (R–0)	T1PWM DATA OUT (RW–0)	T1PWM FUNCTION (RW–0)	T1PWM DATA DIR (RW–0)	T1IC/CR DATA IN (R–0)	T1IC/CR DATA OUT (RW–0)	T1IC/CR FUNCTION (RW–0)	T1IC/CR DATA DIR (RW–0)
T1PRI	104Fh	P04F	T1 STEST (RP–0)	T1 PRIORITY (RP–0)	—	—	—	—	—	—

† Once the WD OVRFL RST ENA bit is set, these bits cannot be changed until a reset; this applies only to the standard watchdog and to the simple counter. In the hard watchdog, these bits can be modified at any time; the WD INPUT SELECT2 bit is ignored.

7.9.1 Timer 1 Control Register 1 (T1CTL1)

The T1CTL1 register controls the prescaler inputs to the watchdog timer and the general-purpose timer.

Timer 1 Control Register 1 (T1CTL1)
[Memory Address – 1049h]

Bit #	7	6	5	4	3	2	1	0
P049	WD OVRFL TAP SEL	WD INPUT SELECT2	WD INPUT SELECT1	WD INPUT SELECT0	—	T1 INPUT SELECT2	T1 INPUT SELECT1	T1 INPUT SELECT0
	RP–0	RP–0	RP–0	RP–0		RW–0	RW–0	RW–0

R = Read, W = Write, P = Write protected when WD OVRFL RST ENA=1 (only in standard watchdog and simple counter configurations), –*n* = Value of the bit after the register is reset

Bit 0–2 **T1 INPUT SELECT0–2.** Timer 1 Input Select 0–2.

These three bits select one of eight possible clock sources for the timer 1 general-purpose counter. These sources are:

- The system clock with no prescale (system clock)
- The system clock when the external input T1EVT is high (pulse accumulation)
- An external source synchronized with the system clock (event input)
- No system clock source (no clock input)
- One of four taps from the 8-bit prescaler, which provides the system clock divided by 4, 16, 64, or 256

The combinations are shown below:

T1 INPUT SELECT2	T1 INPUT SELECT1	T1 INPUT SELECT0	Counter Clock Source
0	0	0	System clock
0	0	1	Pulse accumulation
0	1	0	Event input
0	1	1	No clock input
1	0	0	System clock/4
1	0	1	System clock/16
1	1	0	System clock/64
1	1	1	System clock/256

Bit 3 **Reserved.** Read data is indeterminate.

Bits 4–6 **WD INPUT SELECT0–2.** Watchdog Input Select 0–2.

Standard watchdog and simple counter: These three bits select one of eight possible clock sources. Once the WD OVRFL RST ENA bit is set, the values of these three bits can be changed only after a reset; a write to this bit has no effect when the WD OVRFL RST ENA bit is set.

Hard watchdog: The WD INPUT SELECT0 and WD INPUT SELECT1 bits are used to select one of the four possible clock sources. The clock sources come from one of the four taps from the 8-bit prescaler, which provides the system clock divided by 4, 16, 64, or 256. Note that the WD INPUT SELECT2 bit is functionally interpreted as 1.

The combinations are shown below:

WD INPUT SELECT2	WD INPUT SELECT1	WD INPUT SELECT0	Counter Clock Source
0	0	0	system clock†
0	0	1	pulse accumulation†
0	1	0	event input†
0	1	1	no clock input†
1	0	0	system clock/4
1	0	1	system clock/16
1	1	0	system clock/64
1	1	1	system clock/256

† These options are not available for the hard watchdog

Bit 7 **WD OVRFL TAP SEL.** Watchdog Overflow Tap Select.

This bit determines whether the watchdog counter operates as a 15-bit or a 16-bit counter in the standard watchdog, hard watchdog, and simple counter options. The default is the full 16 bits of the counter. If a shorter watchdog counter overflow rate is needed, then the most significant bit of the counter can be forced to remain at 1. This, in effect, changes the watchdog counter to a 15-bit counter with an overflow period half that of a 16-bit counter. This tap select feature, combined with the clock prescaler, allows watchdog overflow rates from 2^{15} to 2^{24} system clock cycles. Once the WD RST ENA bit is set, this bit can be changed only after a reset (for the nonwatchdog mode of the standard watchdog and simple counter). In the hard watchdog, this bit can be changed at any time.

0 = 16-bit watchdog counter overflow.
1 = 15-bit watchdog counter overflow.

7.9.2 Timer 1 Control Register 2 (T1CTL2)

The T1CTL2 register controls the timer 1 and watchdog overflow interrupts and contains the timer 1 software reset bit.

Timer 1 Control Register 2 (T1CTL2)
[Memory Address – 104Ah]

Bit #	7	6	5	4	3	2	1	0
P04A	WD OVRFL RST ENA	WD OVRFL INT ENA	WD OVRFL INT FLAG	T1 OVRFL INT ENA	T1 OVRFL INT FLAG	—	—	T1 SW RESET
	RS–0	RW–0	RC–0	RW–0	RC–0			S–0

R = Read, S = Set only, W = Write, C = Clear only, *–n* = Value of the bit after the register is reset

Bit 0 **T1 SW RESET.** Timer 1 Software Reset.

This bit is always read as a zero; however, when a one is written to this bit, the counter resets to 0000h on the next system clock cycle.

Bits 1, 2 **Reserved.** Read values are indeterminate.

Bit 3 **T1 OVRFL INT FLAG.** Timer 1 Overflow Interrupt Flag.

This bit indicates the status of the timer 1 overflow interrupt.

0 = General-purpose overflow interrupt inactive.
1 = General-purpose overflow interrupt pending.

Bit 4 **T1 OVRFL INT ENA.** Timer 1 Overflow Interrupt Enable.

This bit controls the timer 1 overflow interrupting capability.

0 = Disables interrupt.
1 = Enables interrupt.

Bit 5 **WD OVRFL INT FLAG.** Watchdog Overflow Interrupt Flag.

Note:

This bit operates differently for TMS370Cxxx devices. Refer to Section A.2, page A-3.

This bit is set if the last reset is initiated by the watchdog counter. Setting this bit will not prevent watchdog resets. This bit is cleared by writing a zero to it or by any system reset that is not initiated by the watchdog counter.

0 = Watchdog interrupt inactive.
1 = Watchdog counter has overflowed or the incorrect value is written to the watchdog reset key register.

Bit 6 **WD OVRFL INT ENA.** Watchdog Overflow Interrupt Enable.

This bit controls the watchdog overflow interrupting capability.

0 = Disables watchdog interrupt.
1 = Enables watchdog interrupt.

Bit 7 **WD OVRFL RST ENA.** Watchdog Overflow Reset Enable.

Note:

This bit operates differently for TMS370Cxxx devices. Refer to Section A.2, page A-3.

Standard watchdog: This bit controls the ability of a watchdog timer to generate a reset. The watchdog timer is a simple counter pulse accumulator when cleared. Once set, this bit can be cleared only by any system reset and locks the values of other WD bits so that they can be changed only after reset.

0 = Watchdog counter does *not* initiate a reset upon overflow.
1 = Watchdog counter *does* initiate a reset upon overflow.

Simple counter: This bit protects the WD INPUT SELECT and WD OVRFL TAP SEL bits. Once set, subsequent writes to these control bits are ignored; they can be changed only after reset.

0 = Other WD bits are not protected.
1 = Locks the value of other WD bits.

Hard watchdog: This bit is ignored.

Note:

Be careful using the AND, OR, XOR, CMPBIT, SBIT0, OR SBIT1 instructions to modify this register. The read/modify/write nature of these instructions can inadvertently clear an interrupt flag that was set between the read and the write cycles. If the state of the interrupt enable bits is known, the MOV #n1,Pn2 instruction can be used. If the state of the interrupt enable bits is not known, a sequence similar to the example shown below should be used.

```
;clearing the T1 OVRFL INT FLAG
      MOV   P04A,A
      OR    #028H,A
      AND   #0F7H,A
      MOV   A,P04A
```

7.9.3 Timer 1 Control Register 3 (T1CTL3)

The T1CTL3 register controls the edge-detect and compare interrupts. The six active bits in this register serve different functions for each mode, as shown below:

Timer 1 Control Register 3 (T1CTL3)
[Memory Address – 104Bh]

Mode: Dual Compare

Bit #	7	6	5	4	3	2	1	0
P04B	T1EDGE INT FLAG	T1C2 INT FLAG	T1C1 INT FLAG	—	—	T1EDGE INT ENA	T1C2 INT ENA	T1C1 INT ENA
	RC–0	RC–0	RC–0			RW–0	RW–0	RW–0

Mode: Compare/Capture

Bit #	7	6	5	4	3	2	1	0
P04B	T1EDGE INT FLAG	—	T1C1 INT FLAG	—	—	T1EDGE INT ENA	—	T1C1 INT ENA
	RC–0		RC–0			RW–0		RW–0

R = Read, W = Write, C = Clear only, *–n* = Value of the bit after the register is reset

Bit 0 **T1C1 INT ENA.** Timer 1 Compare 1 Interrupt Enable.

This bit determines whether or not the compare register flag can generate an interrupt.

0 = Disables interrupt.
1 = Enables interrupt.

Bit 1 **T1C2 INT ENA.** Timer 1 Compare 2 Interrupt Enable.

Dual compare mode only: This bit determines whether or not the capture/compare register flag can generate an interrupt.

0 = Disables interrupt.
1 = Enables interrupt.

Capture/compare mode: Reserved. Read data is indeterminate.

Bit 2 **T1EDGE INT ENA.** Timer 1 Edge Interrupt Enable.

This bit determines whether or not the active edge input to the T1IC/CR pin generates an interrupt. The T1EDGE DET ENA bit (T1CTL4.0) must be set before an edge can be detected.

0 = Disables interrupt.
1 = Enables interrupt.

Bit 3, 4 **Reserved.** Read data is indeterminate.

Bit 5 **T1C1 INT FLAG.** Timer 1 Compare 1 Interrupt Flag.

This bit is set when the compare register first matches the counter value.

0 = Interrupt inactive.
1 = Interrupt pending.

Bit 6 **T1C2 INT FLAG.** Timer 1 Compare 2 Interrupt Flag.

Dual compare mode: This bit is set when the capture/compare register first matches the counter value.

0 = Interrupt inactive.
1 = Interrupt pending.

Capture/compare mode: Reserved. Read data is indeterminate.

Bit 7 **T1EDGE INT FLAG.** Timer 1 Edge Interrupt Flag.

This bit indicates when an external pulse transition of the correct polarity is detected on the timer 1 input capture/counter reset (T1IC/CR) pin. This bit also indicates an input capture in the capture/compare mode.

0 = No transition.
1 = Transition detected.

Note:

Be careful using the AND, OR, XOR, CMPBIT, SBIT0, or SBIT1 instructions to modify this register. The read/modify/write nature of these instructions can inadvertently clear an interrupt flag that was set between the read and the write cycles. If the state of the interrupt enable bits is known, the MOV #n1,Pn2 instruction can be used. If the state of the interrupt enable bits is not known, a sequence similar to the example shown below should be used.

```
;Clearing the T1C1 INT FLAG
      MOV    P04B,A
      OR     #0E0h,A
      AND    #0DFh,A
      MOV    A,P04B
```

7.9.4 Timer 1 Control Register 4 (T1CTL4)

The T1CTL4 register controls the mode of operation and various functions of the timer 1 input and output pins. The bits in this register serve different functions, depending on the mode.

Timer 1 Control Register 4 (T1CTL4)
[Memory Address – 104Ch]

Mode: Dual Compare

Bit #	7	6	5	4	3	2	1	0
P04C	T1 MODE=0	T1C1 OUT ENA	T1C2 OUT ENA	T1C1 RST ENA	T1CR OUT ENA	T1EDGE POLARITY	T1CR RST ENA	T1EDGE DET ENA
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

Mode: Compare/Capture

Bit #	7	6	5	4	3	2	1	0
P04C	T1 MODE=1	T1C1 OUT ENA	—	T1C1 RST ENA	—	T1EDGE POLARITY	—	T1EDGE DET ENA
	RW–0	RW–0		RW–0		RW–0		RW–0

R = Read, W = Write, *–n* = Value of the bit after the register is reset

Bit 0 **T1EDGE DET ENA.** Timer 1 Edge Detect Enable.

Dual compare mode: This bit enables the edge detection circuit to sense the next level transition on the timer 1 T1IC/CR pin. This bit is cleared after the selected transition is detected and during reset.

0 = Disables edge detection.
1 = Enables edge detection.

Capture/compare mode: This bit enables the input capture circuit to capture the current counter value upon the next level transition on the counter reset/input capture pin, as determined by the T1EDGE POLARITY bit. This bit remains unchanged after the selected transition is detected.

0 = Disables input capture.
1 = Enables input capture.

Bit 1 **T1CR RST ENA.** Timer 1 External Reset Enable.

Dual compare mode: This bit determines whether or not an external signal can reset the counter.

0 = Disables external reset of the counter.
1 = Enables external reset of the counter on the next valid edge detect.

Capture/compare mode: Reserved. Read data is indeterminate.

Bit 2 **T1EDGE POLARITY.** Timer 1 Edge Polarity.

This bit determines the transition direction on the T1IC/CR pin to trigger a capture or counter reset, depending on the counter mode selected.

0 = Trigger on a high-to-low transition.
1 = Trigger on a low-to-high transition.

Bit 3 **T1CR OUT ENA.** Timer 1 External Edge Output Enable.

Dual compare mode: This bit determines whether or not the input signal on the T1IC/CR pin can toggle the output signal on the T1PWM pin.

0 = Disables pulse to toggle output.
1 = Enables pulse to toggle output.

Capture/compare mode: Reserved. Read data is indeterminate.

Bit 4 **T1C1 RST ENA.** Timer 1 Compare 1 Reset Enable.

When this bit is set and compare register 1 is equal to the counter, the counter will reset on the next counter increment.

0 = Disables counter reset upon compare equal.
1 = Enables counter reset upon compare equal.

Bit 5 **T1C2 OUT ENA.** Timer 1 Output-Compare Output Enable 2.

Dual Compare Mode: When this bit is set and compare register 2 is equal to the counter, the T1PWM pin toggles (when configured as a PWM pin).

0 = Disables pulse to toggle output.
1 = Enables pulse to toggle output.

Capture/compare mode: Reserved. Read data is indeterminate.

Bit 6 **T1C1 OUT ENA.** Timer 1 Output-Compare Output Enable 1.

When this bit is set and the compare register 1 is equal to the counter, the T1PWM pin toggles (when configured as a PWM pin).

0 = Disables pulse to toggle output.
1 = Enables pulse to toggle output.

Bit 7 **T1 MODE.** Timer 1 Mode Select.

This bit selects the general-purpose counter mode.

0 = Dual compare mode.
1 = Capture/compare mode.

7.9.5 Timer 1 Port Control Registers (T1PC1 and T1PC2)

Port control registers (PCRs) T1PC1 and T1PC2 are organized to allow all functions for a pin to be programmed in one write cycle. Each module pin is controlled by a nibble in one of the PCRs.

7.9.5.1 Timer 1 Port Control Register 1 (T1PC1)

The T1PC1 register controls the I/O functions of the timer 1 module, T1EVT pin.

Timer 1 Port Control Register 1 (T1PC1)
[Memory Address – 104Dh]

Bit #	7	6	5	4	3	2	1	0
P04D	—	—	—	—	T1EVT DATA IN	T1EVT DATA OUT	T1EVT FUNCTION	T1EVT DATA DIR
					R–0	RW–0	RW–0	RW–0

R = Read, W = Write, *–n* = Value of the bit after the register is reset

Bit 0 **T1EVT DATA DIR.** Timer 1 Event-Pin Data Direction.

This bit selects the T1EVT pin as an input or output if the T1EVT FUNCTION bit = 0.

0 = Enables T1EVT pin as data input.
1 = Enables T1EVT pin as data output.

Bit 1 **T1EVT FUNCTION.** T1EVT Pin Function Select.

This bit determines the function of the T1EVT pin.

0 = T1EVT is a general-purpose digital I/O pin.
1 = T1EVT is the event-input pin.

Bit 2 **T1EVT DATA OUT.** T1EVT Pin Data Out.

This bit contains the data to be output on the T1EVT pin if the following conditions are met:

a. TIEVT DATA DIR = 1.
b. T1EVT FUNCTION = 0.

Bit 3 **T1EVT DATA IN.** T1EVT Pin Data in.

This bit contains the data present on the T1EVT pin. A write operation to this bit has no effect.

Bits 4–7 **Reserved.** Read data is indeterminate.

7.9.5.2 Timer 1 Port Control Register 2 (T1PC2)

The T1PC2 register controls the I/O functions of the T1IC/CR and T1PWM pins.

Timer 1 Port Control Register 2 (T1PC2)
[Memory Address – 104Eh]

Bit #	7	6	5	4	3	2	1	0
P04E	T1PWM DATA IN	T1PWM DATA OUT	T1PWM FUNCTION	T1PWM DATA DIR	T1IC/CR DATA IN	T1IC/CR DATA OUT	T1IC/CR FUNCTION	T1IC/CR DATA DIR
	R–0	RW–0	RW–0	RW–0	R–0	RW–0	RW–0	RW–0

R = Read, W = Write, *–n* = Value of the bit after the register is reset

Bit 0 **T1IC/CR DATA DIR.** T1IC/CR Pin Data Direction.

This bit selects the T1IC/CR pin as an input or output if the T1IC/CR FUNCTION bit = 0.

0 = Enables T1IC/CR pin data input.
1 = Enables T1IC/CR pin data output.

Bit 1 **T1IC/CR FUNCTION.** T1IC/CR Pin Function Select.

This bit determines the function of the T1IC/CR pin.

0 = The T1IC/CR pin is a general-purpose digital I/O pin.
1 = The T1IC/CR pin is the input capture/counter reset pin.

Bit 2 **T1IC/CR DATA OUT.** T1IC/CR Pin Data Out.

This bit contains the data output on pin T1IC/CR if the following conditions are met:

a. T1IC/CR DATA DIR = 1.
b. T1IC/CR FUNCTION = 0.

Bit 3 **T1IC/CR DATA IN.** T1IC/CR Pin Data In.

This pin contains the data input on pin T1IC/CR. A write operation to this bit has no effect.

Bit 4 **T1PWM DATA DIR.** T1PWM Pin Data Direction.

This bit selects the T1PWM pin as an input or output if the T1PWM FUNCTION bit = 0.

0 = Enables T1PWM pin data input.
1 = Enables T1PWM pin data output.

Bit 5 **T1PWM FUNCTION.** T1PWM Pin Function Select.

This bit determines the function of the T1PWM pin.

0 = The T1PWM pin is a general-purpose digital I/O pin.
1 = The T1PWM pin is the PWM output.

Bit 6 **T1PWM DATA OUT.** T1PWM Pin Data Out.

This bit contains the data to be output on the T1PWM pin if the following conditions are met:

a. T1PWM DATA DIR = 1.
b. T1PWM FUNCTION = 0.

Bit 7 **T1PWM DATA IN.** T1PWM Pin Data In 1.

This bit contains the data input on pin T1PWM. A write operation to this bit has no effect.

Note:

See subsection 14.6.1, page 14-30, for examples of the PWM pin initialization.

7.9.6 Timer 1 Interrupt Priority Control Register (T1PRI)

The T1PRI register controls the level of the timer 1 interrupt. You can write to this register only in the privilege mode. During normal operation, this is a read-only register.

Timer 1 Interrupt Priority Control Register (T1PRI)
[Memory Address – 104Fh]

Bit #	7	6	5	4	3	2	1	0
P04F	T1 STEST	T1 PRIORITY	—	—	—	—	—	—
	RP–0	RP–0						

R = Read, P = Privilege write only, *–n* = Value of the bit after register is reset

Bits 0–5 **Reserved.** Read data is indeterminate.

Bit 6 **T1 PRIORITY.** Timer 1 Interrupt Priority Select.

This bit determines the level of the interrupt generated by timer 1.

0 = Interrupts are level 1 (high priority) requests.
1 = Interrupts are level 2 (low priority) requests.

Bit 7 **T1 STEST.** Timer 1 STEST.

This bit must be cleared (0) to ensure proper operation.

Chapter 8

Timer 2 Module

This chapter discusses the architecture and programming of the timer 2 module and covers the following topics:

Topic **Page**

8.1 Timer 2 Overview

The 16-bit general-purpose timer 2 module is composed of a 16-bit resettable counter, 16-bit compare register with associated compare logic, a 16-bit capture register, and a 16-bit register that functions as a capture register in one mode and as a compare register in the other mode. The timer 2 module adds an additional timer that provides event count, input capture, and compare functions. Timer 2 provides capabilities for:

System Requirements	Timer Resource
Real-time system control	Interval timers with interrupts
Input pulse-width measurement	Pulse accumulate or input capture functions
External event synchronization	Event count function
Timer output control	Compare function
PWM output control	PWM output function

8.1.1 Physical Description

The timer 2 module has the following features and is shown in Figure 8–1 on the following page:

- ❑ A 16-bit resettable counter
- ❑ A 16-bit compare register with associated compare logic
- ❑ A 16-bit capture register
- ❑ A 16-bit capture/compare register
- ❑ Selectable edge-detection circuitry
- ❑ Interrupts

  The timer 2 module has maskable interrupts for:

  - Two input captures
  - Two output compares
  - Counter overflow
  - External edge detect

- ❑ I/O Pins

  The timer 2 module has three I/O pins that can be dedicated as timer functions or used as general-purpose I/O pins. They are:

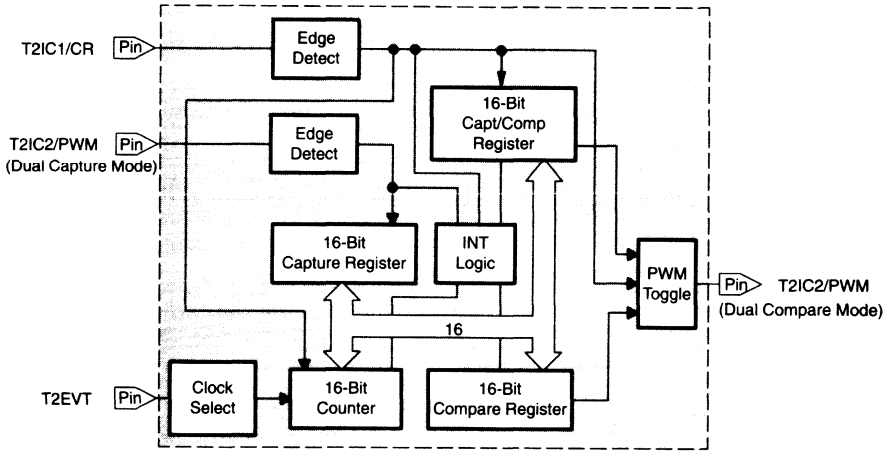
  - T2EVT, an input to the event counter or the external clock source
  - T2IC1/CR, an input to the counter reset, input capture, or PWM circuit
  - T2IC2/PWM, the pulse-width modulation output or a second input capture

The definitions of these pins are contained in the two port control registers located at addresses P06E and P06D of peripheral file frame 6. Table 8–1 defines the functions of the three timer 2 I/O pins for both operating modes.

Table 8–1. Timer 2 I/O Pin Definitions

Pin	Dual Compare Mode	Dual Capture Mode
T2IC1/CR	Counter reset input	Input capture 1 input
T2IC2/PWM	PWM output	Input capture 2 input
T2EVT	External event input or pulse accumulate input	External event input or pulse accumulate input

Figure 8–1. Timer 2 Block Diagram



8.1.2 Operating Modes

- **Dual compare mode.** The timer is configured to provide dual compare registers, external or software reset of the counter, internal or external clock source, and a programmable pulse-width modulated (PWM) output. The T2IC2/PWM pin can also be configured to toggle upon an external input edge. The external clock source can be selected for use as an event counter or pulse accumulator.

- **Dual capture mode.** The timer is configured to provide dual input capture registers and one compare register for use as a general-purpose timer. The compare register can provide periodic interrupts to the rest of the microcomputer. Each capture register can be configured to capture the current counter value upon either edge of an external input.

8.1.3 Control Registers

The timer 2 control registers are located at addresses 1060h to 106Fh, with locations 1068h and 1069h reserved. The functions of these locations are shown in Table 8–2.

Table 8–2. Timer 2 Memory Map

Peripheral File Location	Symbol	Name	Description
P060	T2CNTR	T2 Counter — MSbyte	16-bit resettable up counter.
P061		T2 Counter — LSbyte	
P062	T2C	Compare Register — MSbyte	16-bit compare register.
P063		Compare Register — LSbyte	
P064	T2CC	Capture/Compare Register — MSbyte	16-bit capture/compare register.
P065		Capture/Compare Register — LSbyte	
P066	T2IC	Capture Register — MSbyte	16-bit capture register.
P067		Capture Register — LSbyte	
P068		Reserved	
P069		Reserved	
P06A	T2CTL1	Timer 2 Control Register 1	Controls the clock input selection, counter overflow interrupts, and counter software reset.
P06B	T2CTL2	Timer 2 Control Register 2	Contains interrupt flags and controls the module's capability to issue interrupts.
P06C	T2CTL3	Timer 2 Control Register 3	Controls the mode of operation, outputs, active transition polarity, and counter reset.
P06D	T2PC1	Timer 2 Port Control Register 1	Assigns the I/O function of the T2EVT pin as either a general-purpose digital I/O or external event input of the module.
P06E	T2PC2	Timer 2 Port Control Register 2	Assigns the I/O functions of the T2IC1/CR and T2IC2/PWM pins as either general-purpose digital I/O pins or the input capture/counter reset and PWM output pins, respectively.
P06F	T2PRI	Timer 2 Interrupt Priority Control Register	Assigns the priority level of interrupts generated by the timer 2 module.

8.2 Timer 2 Components

The timer 2 module uses a 16-bit counter, a compare register, a capture register, and a capture/compare register to provide event, compare, and capture functions.

8.2.1 16-Bit Resettable Counter

The 16-bit free-running, read-only counter (T2CNTR) is clocked by the system clock, external event, or system clock while an external event is active (pulse accumulate).

- ❑ During initialization, the counter is loaded with 0000h and begins its count.
- ❑ If the counter is not reset before reaching FFFFh, the counter rolls over to 0000h and continues counting. Upon counter rollover, the T2 OVRFL INT FLAG bit (T2CTL1.3) is set, and a timer interrupt is generated if the T2OVRFL INT ENA bit (T2CTL1.4) is set.
- ❑ The counter can be reset to 0000h during counting by:
  - A 1 written to the T2 SW RESET bit (T2CTL1.0),
  - A compare equal condition from the dedicated T2 compare function,
  - System reset, or
  - An external pulse on the T2IC1/CR pin (dual compare mode only).

You can select which external transition on the T2IC1/CR pin, low-to-high or high-to-low, to reset the counter. To do this, use the T2EDGE1 POLARITY bit (T2CTL3.2).

Special circuitry prevents the contents of the T2CNTR register from changing in the middle of a 16-bit read operation. See the note in Section 8.8.

8.2.2 Compare Register

The compare register circuit consists of a 16-bit wide, read/write data register (T2C) and logic to compare the counter's current value with the value stored in the compare register.

When the counter value matches the compare register value, then the circuit:

- ❑ Sets the T2C1 INT FLAG bit (T2CTL2.5) to 1,
- ❑ Generates a timer 2 interrupt if the T2C1 INT ENA bit (T2CTL2.0) is set,
- ❑ Resets the counter if the T2C1 RST ENA bit (T2CTL3.4) is set, and
- ❑ Toggles the PWM output pin if the T2C1 OUT ENA bit (T2CTL3.6) is set (dual compare mode only).

Once T2C1 INT FLAG bit is set by a compare-equal condition and then cleared, it will not be set again if the same compare-equal condition still exists (that is, the same compare-equal condition can set the T2C1 INT FLAG bit only once). This flag causes various events to occur, depending on the mode of operation and on which enable bits are set.

Special circuitry prevents the T2C register from changing in the middle of a 16-bit read or write operation. See the note in Section 8.8.

The compare register value required for a specific timing application can be calculated using the following formula:

$$Compare\ Value = \frac{t}{SYSCLK} - 1$$

where:

t = desired timer Compare period (seconds)
SYSCLK = 4 / CLKIN (external clock frequency)

Table 8–3 provides some sample compare register values to achieve various desired timings with a 20-MHz crystal.

Table 8–3. Timer 2 Compare Values: (CLKIN = 20 MHz)

Time		T2 Compare Register		% Error (See Note)
Seconds	m Seconds	Decimal	Hex	
0.0005	0.5	2499	009C3h	0.0000
0.001	1	4999	01387h	0.0000
0.002	2	9999	0270Fh	0.0000
0.005	5	24999	061A7h	0.0000
0.010	10	49999	0C34Fh	0.0000
0.013	13	64999	0FDE7h	0.0000

Note: % error induced by the timer 2 formula. This error margin will vary, depending on the desired timer compare period and the minimum timer resolution (SYSCLK).

8.2.3 Capture Register (Dual Capture Mode Only)

The 16-bit capture register (T2IC) is a read-only data register. This register captures the counter values when an input capture pulse (pin T2IC2/PWM) is received. The capture register can be read at addresses P066 (MSbyte) and P067 (LSbyte) of peripheral file frame 6. Writes to this register are ignored, so the capture register retains the last counter value captured until another input capture pulse loads a new value in the register.

On receipt of a capture pulse, the circuit:

- ❑ Loads the value of the 16-bit counter into the capture register,
- ❑ Sets the T2EDGE2 INT FLAG bit (T2CTL2.6) to indicate that the capture register has latched the current counter value, and
- ❑ If the T2C2 INT ENA bit (T2CTL2.1) is set, generates an interrupt.

Special circuitry prevents the T2IC register from changing in the middle of a 16-bit read or write operation. See the note in Section 8.8.

8.2.4 Capture/Compare Register

The 16-bit capture/compare register (T2CC) can serve one of two functions, depending on the operating mode. The capture/compare register is located at addresses P064 (capture/compare register MSbyte) and P065 (capture/compare register LSbyte) in peripheral file frame 6.

Special circuitry prevents the T2CC register from changing in the middle of a 16-bit read or write operation. See the note in Section 8.8.

Dual compare mode

In the dual compare mode, the T2CC register becomes a read/write compare register. It functions exactly as the one described in subsection 8.2.2 except that T2CC **cannot** reset the counter.

When the 16-bit counter value matches the capture/compare register value, then the circuit:

- ❑ Sets the T2C2 INT FLAG bit (T2CTL2.6) to 1,
- ❑ Toggles the PWM output pin if the T2C2 OUT ENA bit (T2CTL3.5) is set, and
- ❑ Generates a timer 2 interrupt if the T2C2 INT ENA bit (T2CTL2.1) is set.

Dual capture mode

In the dual capture mode, the capture/compare register becomes a read-only capture register. When an external pulse appears on pin T2IC1/CR, the following events occur if the T2EDGE1 DET ENA bit (T2CTL3.0) is set:

- ❑ The current counter value is latched into the capture/compare register,
- ❑ The T2EDGE1 INT FLAG bit (T2CTL2.7) is set, and
- ❑ Generates an interrupt if the T2EDGE1 INT ENA bit (T2CTL2.2) is set.

8.3 Operating Modes

The timer 2 operating mode is determined by the T2 MODE bit (T2CTL3.7).

T2 MODE = 0 — dual compare mode
T2 MODE = 1 — dual capture mode

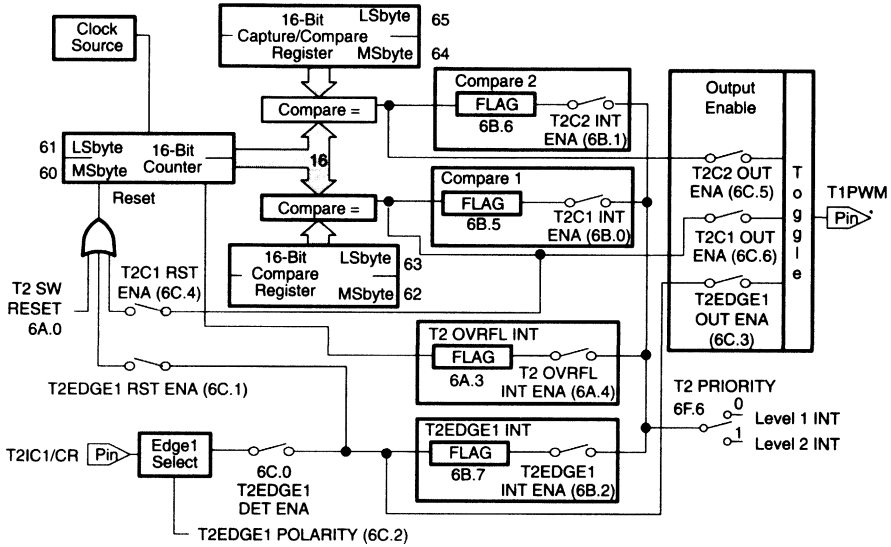
8.3.1 Dual Compare Mode

The dual compare mode provides the following:

- A 16-bit compare register (called compare 1)
- A 16-bit capture/compare register that acts as a compare register (called compare 2)
- A 16-bit external, resettable counter
- A timer output pin

These components allow timer 2 to act as an interval timer, a PWM output, simple output toggle, or many other timer functions. In the dual compare mode, the operation of the timer 2 module is identical to that of the T1 module, with the exception of the clock sources. The dual compare mode is shown in Figure 8–2.

Figure 8–2. Dual Compare Mode



Note: The bit locations in this figure are shown in following format: *xx.n*. The *xx* is the hexadecimal address of the peripheral file register that contains the bit, and *n* is the bit number (7 = MSB, 0 = LSB). Register locations are designated by *xx*, the hexadecimal address of the peripheral file.

8

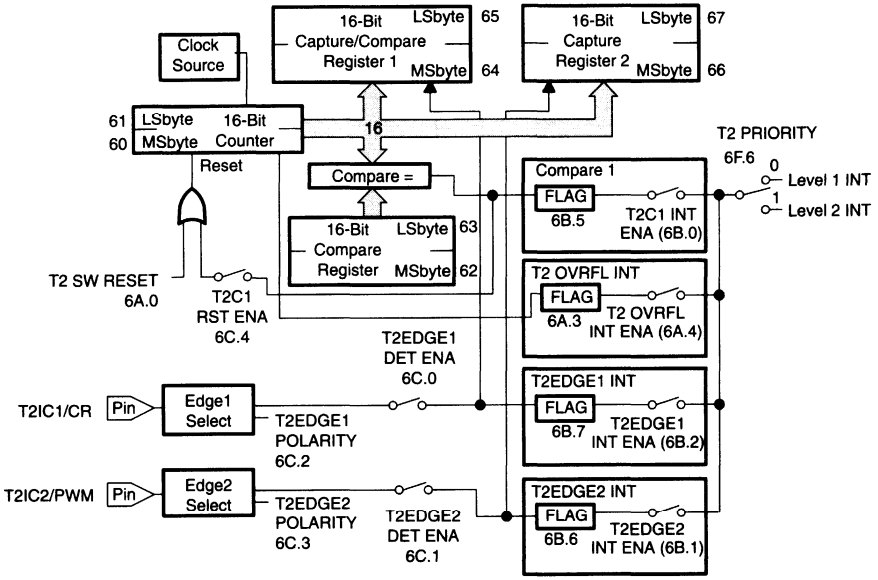
8.3.2 Dual Capture Mode

In the dual capture mode, timer 2 provides:

- A 16-bit compare register for use as a programmable interval timer
- A 16-bit capture register for external input time and pulse width measurement
- A 16-bit capture/compare register that acts as a capture register for external input timing and pulse width measurement

The dual capture mode is shown in Figure 8–3.

Figure 8–3. Dual Capture Mode



Note: The bit locations in this figure are shown in following format: *xx.n*. The *xx* is the hexadecimal address of the peripheral file register that contains the bit, and *n* is the bit number (7 = MSB, 0 = LSB). Register locations are designated by *xx*, the hexadecimal address of the peripheral file.

Each capture input pin (T2IC1/CR and T2IC2/PWM) has an input edge detect function enabled by the associated DET ENA control bit, with the associated POLARITY bit selecting the active input transition.

On the occurrence of a valid input on the T2IC1/CR or T2IC2/PWM pin, the current counter value is loaded into the 16-bit capture/compare register or 16-bit input capture register, respectively. In addition, the respective input capture INT FLAG bit is set, and a timer interrupt is generated if the respective INT ENA bit is set.

8.4 Edge-Detection Circuitry

This edge detection circuitry senses an active pulse transition on the input pins and provides appropriate output transitions to the rest of the module.

Dual Compare Mode

In this mode, the edge detection circuitry is connected to the module's T2IC1/CR pin. The program must set the T2EDGE 1 DET ENA bit (T2CTL3.0) to re-enable the timer 2 module after each edge detection.

When the timer 2 module detects an active transition (while enabled), then the module:

- ❑ Clears the T2EDGE1 DET ENA bit,
- ❑ Sets the T2EDGE1 INT FLAG bit (T2CTL2.7),
- ❑ Resets the counter if T2EDGE1 RST ENA bit (T2CTL3.1) is set, and
- ❑ Toggles the output flip-flop if the T2EDGE1 OUT ENA bit (T2CTL3.3) is set.

In the dual compare mode, the T2EDGE1 POLARITY bit (T2CTL3.2) determines whether the active transition is low-to-high or high-to-low.

Dual Capture Mode

In this mode, the edge detection circuitry is connected to both the T2IC1/CR pin and the T2IC2/PWM pin.

When the edge 1 detect circuit detects an active edge transition on the T2IC1/CR pin, then the timer 2 module:

- ❑ Loads the capture/compare register with the current counter value and
- ❑ Sets the T2EDGE1 INT FLAG bit (T2CTL2.7).

When the edge 2 detect circuit detects an active edge transition on the T2IC2/PWM pin, then the module:

- ❑ Loads the capture register with the current counter value and
- ❑ Sets the T2EDGE2 INT FLAG bit (T2CTL2.6).

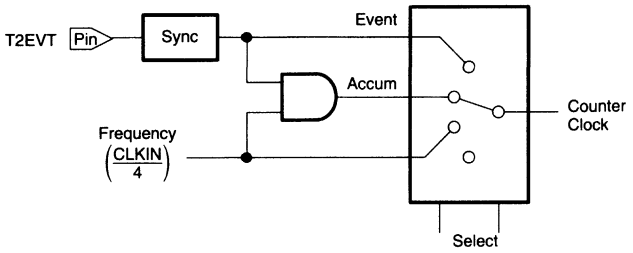
The T2EDGE1 POLARITY bit (T2CTL3.2) and the T2EDGE2 POLARITY bit (T2CTL3.3) determine the transition (rising or falling) to be detected.

8.5 Clock Sources

The timer 2 clock sources are shown Figure 8–4 and can be any of the following:

- System clock
- No clock (the counter is stopped)
- External clock synchronized to the system clock (event counter)
- System clock while external input is high (pulse accumulation)

Figure 8–4. Timer 2 Clock Sources



The T2 INPUT SELECT0 bit (T2CTL1.1) and the T2 INPUT SELECT1 bit (T2CTL1.2) select one of four clock sources (refer to subsection 8.8.1).

The maximum counter duration with an internal clock is based on the internal system clock time (SYSCLK) as follows:

$$\text{Maximum Counter Duration} = 2^{16} \times \text{SYSCLK}$$
$$\text{Counter Resolution} = \text{SYSCLK}$$

where: $\text{SYSCLK} = 4/\text{CLKIN}$

The external event frequency input to the module cannot exceed CLKIN/8. All external event inputs are synchronized with the system clock.

When the timer is using the system clock input, the 16-bit timer generates an overflow rate of 13.1 ms with 200-ns resolution (CLKIN = 20 MHz).

8.5.1 Event Counter Mode

When you use the event counter clock source, the 16-bit counter is programmable as a 16-bit event counter. An external low-to-high transition on the T2EVT pin provides the clock for the internal timer. The T2EVT external clock frequency cannot exceed the system clock frequency divided by 2.

8.5.2 Pulse Accumulator Mode

When you use the pulse accumulator clock source, the 16-bit counter is programmable as a 16-bit pulse accumulator. An external input on the T2EVT pin is used to gate the internal system clock to the internal timers. While T2EVT input is logic one (high), the timer is clocked at the system clock rate and counts system clock pulses until the T2EVT pin returns to logic zero.

8.6 Interrupts

Interrupts can be enabled to occur upon an input capture, output compare equal, counter overflow, and/or an external edge detect.

In dual compare mode, four separate events can generate an interrupt. These events are:

- ❑ Compare equal for the dedicated compare register if the T2C1 INT ENA bit (T2CTL2.0) is set,
- ❑ Compare equal for the capture/compare register if the T2CC2 INT ENA bit (T2CTL2.1) is set,
- ❑ Counter overflow if the T2 OVERFL INT ENA bit (T2CTL1.4) is set, or
- ❑ External edge detect if the T2EDGE1 DET ENA and T2EDGE1 INT ENA bits are set (T2CTL3.0 and T2CTL2.2, respectively).

In dual capture mode, four separate events can generate an interrupt. These events are:

- ❑ Compare equal for the dedicated compare register if the T2C1 INT ENA bit (T2CTL2.0) is set,
- ❑ Counter overflow if the T2 OVERFL INT ENA bit (T2CTL1.4) is set,
- ❑ External edge 1 detect if the T2EDGE1 DET ENA and T2EDGE1 INT ENA bits are set (T2CTL3.0 and T2CTL2.2), or
- ❑ External edge 2 detect if the T2EDGE2 DET ENA and T2EDGE2 INT ENA bits are set (T2CTL3.1 and T2CTL2.1).

Note:

All set and enabled interrupt flags must be cleared before the processor exits the T2 interrupt routine. If the flags are not reset, the processor will enter the T2 interrupt routine again instead of continuing the mainstream program. If the flag bits are never cleared, the program will lock.

8.7 Low-Power Modes

The timer 2 module supports low-power (powerdown) modes that aid in reducing power consumption during periods of inactivity. These modes are the halt and the standby modes. In both the halt and standby modes, no clocks or external inputs are recognized.

If the PWRDWN/IDLE bit (SCCR2.6) is set, the low-power modes are entered when an IDLE instruction is executed by the CPU. During the low-power mode, the timer 2 module holds the pre-idle status of all storage elements. All external pins are held constant, regardless of the pin function; inputs remain inputs, output low levels remain low, and output high levels remain high. When the idle state is exited, the I/O timer module continues from where it entered the idle state.

8.8 Timer 2 Control Registers

Six registers control the timer 2 module operating mode selection, interrupt enable, status flags, and output configuration. These registers are shown in Table 8–4. The bits that are shown in shaded boxes are privilege mode bits; that is, they can be written to only in the privilege mode.

Note:

Special circuitry prevents 16-bit registers from changing in the middle of a 16-bit read or write operation. When you read a 16-bit register, read the least significant byte (LSbyte) first to lock in the value, and then read the most significant byte (MSbyte). When you write to a 16-bit register, write the MSbyte first and then write the LSbyte. The register value does not change between reading and writing the bytes when they are done in this order. While you access a 16-bit register, do not read or write from a second 16-bit register within this module; if you do, the correct value for the first register's MSbyte will not be correct. The 16-bit read/write operation actually occurs when you access the LSbyte.

Read: **LSbyte then MSbyte**
Write: **MSbyte then LSbyte**

Table 8–4. Peripheral File Frame 6: Timer 2 Control Registers

Designation	ADDR	PF	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T2CNTR	1060h	P060	Bit 15	T2 Counter MSbyte						Bit 8
T2CNTR	1061h	P061	Bit 7	T2 Counter LSbyte						Bit 0
T2C	1062h	P062	Bit 15	Compare Register MSbyte						Bit 8
T2C	1063h	P063	Bit 7	Compare Register LSbyte						Bit 0
T2CC	1064h	P064	Bit 15	Capture/Compare Register MSbyte						Bit 8
T2CC	1065h	P065	Bit 7	Capture/Compare Register LSbyte						Bit 0
T2IC	1066h	P066	Bit 15	Capture Register 2 MSbyte						Bit 8
T2IC	1067h	P067	Bit 7	Capture Register 2 LSbyte						Bit 0
T2CTL1	106Ah	P06A	—	—	—	T2 OVRFL INT ENA (RW–0)	T2 OVRFL INT FLAG (RC–0)	T2 INPUT SELECT1 (RW–0)	T2 INPUT SELECT0 (RW–0)	T2 SW RESET (S–0)
T2CTL2	106Bh	P06B	Dual Compare Mode							
			T2EDGE1 INT FLAG (RC–0)	T2C2 INT FLAG (RC–0)	T2C1 INT FLAG (RC–0)	—	—	T2EDGE1 INT ENA (RW–0)	T2C2 INT ENA (RW–0)	T2C1 INT ENA (RW–0)
			Dual Capture Mode							
			T2EDGE1 INT FLAG (RC–0)	T2EDGE2 INT FLAG (RC–0)	T2C1 INT FLAG (RC–0)	—	—	T2EDGE1 INT ENA (RW–0)	T2EDGE2 INT ENA (RW–0)	T2C1 INT ENA (RW–0)
T2CTL3	106Ch	P06C	Dual Compare Mode							
			T2 MODE= 0 (RW–0)	T2C1 OUT ENA (RW–0)	T2C2 OUT ENA (RW–0)	T2C1 RST ENA (RW–0)	T2EDGE1 OUT ENA (RW–0)	T2EDGE1 POLARITY (RW–0)	T2EDGE1 RST ENA (RW–0)	T2EDGE1 DET ENA (RW–0)
			Dual Capture Mode							
			T2 MODE= 1 (RW–0)	—	—	T2C1 RST ENA (RW–0)	T2EDGE2 POLARITY (RW–0)	T2EDGE1 POLARITY (RW–0)	T2EDGE2 DET ENA (RW–0)	T2EDGE1 DET ENA (RW–0)
T2PC1	106Dh	P06D	—	—	—	—	T2EVT DATA IN (RW–0)	T2EVT DATA OUT (RW–0)	T2EVT FUNCTION (RW–0)	T2EVT DATA DIR (RW–0)
T2PC2	106Eh	P06E	T2IC2/PWM DATA IN (R–0)	T2IC2/PWM DATA OUT (RW–0)	T2IC2/PWM FUNCTION (RW–0)	T2IC2/PWM DATA DIR (RW–0)	T2IC1/CR DATA IN (R–0)	T2IC1/CR DATA OUT (RW–0)	T2IC1/CR FUNCTION (RW–0)	T2IC1/CR DATA DIR (RW–0)
T2PRI	106Fh	P06F	**T2 STEST (RP–0)**	**T2 PRIORITY (RP–0)**	—	—	—	—	—	—

8.8.1 Timer 2 Control Register 1 (T2CTL1)

The T2CTL1 register controls the clock input selection, counter overflow interrupts, and counter software reset.

Timer 2 Control Register 1 (T2CTL1)
[Memory Address – 106Ah]

Bit #	7	6	5	4	3	2	1	0
P06A	—	—	—	T2 OVRFL INT ENA	T2 OVRFL INT FLAG	T2 INPUT SELECT1	T2 INPUT SELECT0	T2 SW RESET
				RW–0	RC–0	RW–0	RW–0	S–0

R = Read, W = Write, S = Set only, C = Clear only, –*n* = Value of the bit after the register is reset

Bit 0 **T2 SW RESET.** Timer 2 Software Reset.

When a 1 is written to this bit, the counter will reset to 0000h on the next system clock cycle; however, this bit is always read as a zero.

Bits 1, 2 **T2 INPUT SELECT0–1.** Timer 2 Input Select.

These two bits select one of four clock sources as an input to the counter. The four options are:

- System clock with no prescale,
- System clock when external input is high (pulse accumulation),
- External source synchronized with system clock (event input), or
- No clock.

The combinations are shown below:

T2 INPUT SELECT2	T2 INPUT SELECT1	Counter Clock Source
0	0	system clock
0	1	pulse accumulation
1	0	event input
1	1	no clock input

Bit 3 **T2 OVRFL INT FLAG.** Timer 2 Overflow Interrupt Flag.

This bit is the timer 2 counter overflow bit.

0 = Overflow interrupt inactive.
1 = Overflow interrupt pending.

Bit 4 **T2 OVRFL INT ENA.** Timer 2 Overflow Interrupt Enable.

This bit controls the Timer 2 overflow interrupting capability.

0 = Disables Interrupt.
1 = Enables Interrupt from overflow.

Bits 5, 6, 7 **Reserved.** Read data is indeterminate.

8.8.2 Timer 2 Control Register 2 (T2CTL2)

The T2CTL2 register contains interrupt flags and controls the capability of the module to issue interrupts.

Timer 1 Control Register 2 (T1CTL2)
[Memory Address – 106Bh]

Mode: Dual Compare

Bit #	7	6	5	4	3	2	1	0
P06B	T2EDGE INT FLAG	T2C2 INT FLAG	T2C1 INT FLAG	—	—	T2EDGE1 INT ENA	T2C2 INT ENA	T2C1 INT ENA
	RC–0	RC–0	RC–0			RW–0	RW–0	RW–0

Mode: Dual Capture

Bit #	7	6	5	4	3	2	1	0
P06B	T2EDGE1 INT FLAG	T2EDGE2 INT FLAG	T2C1 INT FLAG	—	—	T2EDGE1 INT ENA	T2EDGE2 INT ENA	T2C1 INT ENA
	RC–0	RC–0	RC–0			RW–0	RW–0	RW–0

R = Read, W = Write, C = Clear only, *–n* = Value of the bit after the register is reset

Bit 0 **T2C1 INT ENA.** Timer 2 Compare 1 Interrupt Enable.

This bit controls the interrupting capability of the compare 1 register.

0 = Disables interrupt.
1 = Enables interrupt from compare 1 register.

Bit 1 *Dual Compare Mode:*

T2C2 INT ENA. Timer 2 Output Compare 2 Interrupt Enable.

This bit controls the interrupting capability of the compare 2 register.

0 = Disables interrupt.
1 = Enables interrupt from compare 2 register.

Dual Capture Mode:

T2EDGE2 INT ENA. Timer 2 External Edge 2 Interrupt Enable.

This bit determines whether or not the active edge input to the T2IC2/PWM pin generates an interrupt.

0 = Disables interrupt.
1 = Enables interrupt.

Bit 2 **T2EDGE1 INT ENA.** Timer 2 External Edge 1 Interrupt Enable.

This bit determines whether or not the active edge input to the T2IC1/CR pin generates an interrupt.

0 = Disables interrupt.
1 = Enables interrupt.

Bits 3, 4 **Reserved.** Read data is indeterminate.

Bit 5 **T2C1 INT FLAG.** Timer 2 Output Compare 1 Interrupt Flag.

This bit is set when the output compare register first matches the counter value.

0 = Interrupt inactive.
1 = Interrupt pending from compare 1.

Bit 6 *Dual Compare Mode:*

T2C2 INT FLAG. Timer 2 Output Compare 2 Interrupt Flag.

This bit is set when the capture/compare register first matches the counter value.

0 = Interrupt inactive.
1 = Interrupt pending from compare 2.

Dual Capture Mode:

T2EDGE2 INT FLAG. Timer 2 Edge 2 Interrupt Flag.

This bit is set when the appropriate edge is detected on T2IC2/PWM and indicates that the capture register was loaded.

0 = Interrupt inactive.
1 = Interrupt pending from edge 2 detect.

Bit 7 **T2EDGE1 INT FLAG.** Timer 2 External Edge 1 Interrupt Flag.

This bit is set when the appropriate edge is detected on the T2IC1/CR pin.

0 = Interrupt inactive.
1 = Interrupt pending from edge 1 detect circuitry.

8

Note:

Be careful using the AND, OR, XOR, CMPBIT, SBIT0, or SBIT1 instruction to modify this register. The read/modify/write nature of these instructions can inadvertently clear an interrupt flag that was set between the read and the write cycles. If the state of the interrupt enable bits is known, the MOV #n1,Pn2 instruction can be used. If the state of the interrupt enable bits is not known, a sequence similar to the example shown below should be used.

```
;Clearing the T2C1 INT FLAG
      MOV    P06B,A
      OR     #0E0h,A
      AND    #0DFh,A
      MOV    A,P06B
```

8.8.3 Timer 2 Control Register 3 (T2CTL3)

The T2CTL3 register controls the timer 2 module mode of operation, outputs, active transition polarity, and counter reset.

Timer 1 Control Register 3 (T1CTL3)
[Memory Address – 106Ch]

Mode: Dual Compare

Bit #	7	6	5	4	3	2	1	0
P06C	T2 MODE=0	T2C1 OUT ENA	T2C2 OUT ENA	T2C1 RST ENA	T2EDGE1 OUT ENA	T2EDGE1 POLARITY	T2EDGE1 RST ENA	T2EDGE1 DET ENA
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

Mode: Dual Capture

Bit #	7	6	5	4	3	2	1	0
P06C	T2 MODE=1	—	—	T2C1 RST ENA	T2EDGE2 POLARITY	T2EDGE1 POLARITY	T2EDGE2 DET ENA	T2EDGE1 DET ENA
	RW–0			RW–0	RW–0	RW–0	RW–0	RW–0

R = Read, W = Write, –*n* = Value of the bit after the register is reset

Bit 0 *Dual Compare Mode:*

T2EDGE1 DET ENA. Timer 2 Edge 1 Detect Enable.

This bit enables the edge detection circuit to sense the next active level transition on the T2IC1/CR pin. This bit is cleared after the selected transition is detected and during reset.

0 = Disables edge 1 detect.
1 = Enables edge 1 detect.

Dual Capture Mode:

T2EDGE1 DET ENA. Timer 2 Edge 1 Detect Enable.

This bit enables the edge detection circuit to sense the next active level transition on the T2IC1/CR pin. This bit remains unchanged after the selected transition is detected and during reset.

0 = Disables input capture.
1 = Enables input capture.

Bit 1 *Dual Compare Mode:*

T2EDGE1 RST ENA. Timer 2 Edge 1 Detect Reset Enable.

This bit controls whether or not an external signal can reset the counter.

0 = Disables external reset of the counter.
1 = Enables external reset of the counter.

Dual Capture Mode:

T2EDGE2 DET ENA. Timer 2 External Edge 2 Detect Enable.

This bit enables the edge detection circuit to sense the next active level transition on the T2IC2/PWM pin. This bit remains unchanged after the selected transition is detected and during reset.

0 = Disables edge detect.
1 = Enables edge detect.

Bit 2 **T2EDGE1 POLARITY.** Timer 2 Edge 1 Polarity Select.

This bit controls which transition level on the T2IC1/CR pin is active.

0 = Trigger on high-to-low transition.
1 = Trigger on low-to-high transition.

Bit 3 *Dual Compare Mode:*

T2EDGE1 OUT ENA. Timer 2 Edge 1 Detect Output Enable.

This bit controls whether or not the pulse indicating an external edge detect toggles the module's output pin.

0 = Disables pulse to toggle output.
1 – Enables pulse to toggle output.

Dual Capture Mode:

T2EDGE2 POLARITY. Timer 2 Edge 2 Polarity Select.

This bit controls which transition level on the T2IC2/PWM pin is active.

0 = Trigger on high-to-low transition.
1 = Trigger on low-to-high transition.

Bit 4 **T2C1 RST ENA.** Timer 2 Output Compare 1 Reset Enable.

This bit controls whether or not the compare equal pulse from the compare register resets the counter on the next counter increment.

0 = Disables reset upon compare equal.
1 = Enables reset upon compare equal.

Bit 5 *Dual Compare Mode:*

T2C2 OUT ENA. Timer 2 Output Compare 2 Enable.

This bit controls whether or not the output compare equal pulse from the capture/compare register toggles the T2IC2/PWM output pin.

0 = Disables pulse to toggle output.
1 = Enables pulse to toggle output.

Dual Capture Mode:

Reserved. Read data is indeterminate.

Bit 6 *Dual Compare Mode:*

T2C1 OUT ENA. Timer 2 Output Compare 1 Enable.

This bit controls whether or not the compare equal pulse from the compare register toggles T2IC2/PWM pin.

0 = Disables pulse from toggling output.
1 = Enables pulse to toggle output.

Dual Capture Mode:

Reserved. Read data is indeterminate.

Bit 7 **T2 MODE.** Timer 2 Mode Select.

This bit selects the operating mode for the counter.

0 = Dual compare mode.
1 = Dual capture mode.

8.8.4 Timer 2 Port Control Registers (T2PC1 and T2PC2)

The port control registers (PCRs) control the functions of the I/O pins. Each module pin is controlled by a nibble in one of the PCRs.

8.8.4.1 Timer 2 Port Control Register1 (T2PC1)

The T2PC1 register assigns the I/O function of the T2EVT pin as either a general-purpose digital I/O or external event input of the module.

Timer 2 Port Control Register 1 (T2PC1)
[Memory Address – 106Dh]

Bit #	7	6	5	4	3	2	1	0
P06D	—	—	—	—	T2EVT DATA IN	T2EVT DATA OUT	T2EVT FUNCTION	T2EVT DATA DIR
					RW–0	RW–0	RW–0	RW–0

R = Read, W = Write, –*n* = Value of the bit after the register is reset

Bit 0 **T2EVT DATA DIR.** Timer 2 Event Pin Data Direction.

This bit determines the data direction on the T2EVT pin if the T2EVT FUNCTION bit = 0.

0 = T2EVT is configured as input.
1 = T2EVT is configured as output.

Bit 1 **T2EVT FUNCTION.** Timer 2 Event Pin Function Select.

This bit selects the function of the T2EVT pin.

0 = T2EVT is a general-purpose digital I/O pin.
1 = T2EVT is the event input pin.

Bit 2 **T2EVT DATA OUT.** Timer 2 Event Pin Data Out.

This bit contains the data to be output on the T2EVT pin if the following conditions are met:

a. T2EVT DATA DIR = 1
b. T2EVT FUNCTION = 0

Bit 3 **T2EVT DATA IN.** Timer 2 Event Pin Data In.

This bit contains the data to be input from the T2EVT pin. A write to this bit has no effect.

Bits 4, 5, 6, 7 **Reserved.** Read data is indeterminate.

8.8.4.2 Timer 2 Port Control Register 2 (T2PC2)

The T2PC2 register assigns the I/O functions of the T2IC1/CR and T2IC2/PWM pins as either general-purpose digital I/O pins or the input capture/counter reset and PWM output pins, respectively.

Timer 2 Port Control Register 2 (T2PC2)
[Memory Address – 106Eh]

Bit #	7	6	5	4	3	2	1	0
P06E	T2IC2/ PWM DATA IN	T2IC2/ PWM DATA OUT	T2IC2/ PWM FUNCTION	T2IC2/ PWM DATA DIR	T2IC1/CR DATA IN	T2IC1/CR DATA OUT	T2IC1/CR FUNCTION	T2IC1/CR DATA DIR
	R–0	RW–0	RW–0	RW–0	R–0	RW–0	RW–0	RW–0

R = Read, W = Write, –*n* = Value of the bit after the register is reset

Bit 0 **T2IC1/CR DATA DIR.** Timer 2 IC1/CR Data Direction.

This bit determines the direction of data on the T2IC1/CR pin if the T2IC1/CR FUNCTION bit = 0.

0 = T2IC1/CR is an input.
1 = T2IC1/CR is an output.

Bit 1 **T2IC1/CR FUNCTION.** Timer 2 IC1/CR Function Select.

This bit determines the function of the T2IC1/CR pin.

0 = T2IC1/CR is a general-purpose digital I/O pin.
1 = T2IC1/CR is the input capture/counter reset pin.

Bit 2 **T2IC1/CR DATA OUT.** Timer 2 IC1/CR Data Out.

This bit contains the data output on the T2IC1/CR pin if the following conditions are true:

a. T2IC1/CR DATA DIR = 1
b. T2IC1/CR FUNCTION = 0

Bit 3 **T2IC1/CR DATA IN.** Timer 2 IC1/CR Data In.

This bit contains the data input on the T2IC1/CR pin. A write to this bit has no effect.

Bit 4 **T2IC2/PWM DATA DIR.** Timer 2 IC2/PWM Data Direction.

This bit determines the direction of data on the T2IC2/PWM pin if the T2IC2/PWM FUNCTION bit = 0.

0 = T2IC1/PWM is an input
1 = T2IC2/PWM is an output.

Bit 5 **T2IC2/PWM FUNCTION.** Timer 2 IC2/PWM Function Select.

This bit determines the function of the T2IC2/PWM pin.

0 = T2IC2/PWM is a general-purpose digital I/O pin.
1 = T2IC2/PWM is the input capture/PWM output pin.

Bit 6 **T2IC2/PWM DATA OUT.** Timer 2 IC2/PWM Data Out.

This bit contains the data output on the T2IC2/PWM pin if the following conditions are true:

a. T2IC2/PWM DATA DIR = 1
b. T2IC2/PWM FUNCTION = 0

Bit 7 **T2IC2/PWM DATA IN.** Timer 2 IC2/PWM Data In.

This bit contains the data input on the T2IC2/PWM pin. A write to this bit has no effect.

Note:

See Section 14.6.1 for examples of PWM pin initialization.

8.8.5 Timer 2 Interrupt Priority Control Register (T2PRI)

The T2PRI register assigns the priority level of interrupts generated by the timer 2 module. You can write to this register only in the privilege mode. During normal operation, this is a read-only register.

8

Timer 2 Priority Control Register (T2PRI)
[Memory Address – 106Fh]

Bit #	7	6	5	4	3	2	1	0
P06F	T2 STEST	T2 PRIORITY	—	—	—	—	—	—
	RP–0	RP–0						

R = Read, P = Privilege write only, *–n* = Value of the bit after the register is reset

Bits 0–5 **Reserved.** Read data is indeterminate.

Bit 6 **T2 PRIORITY.** Timer 2 Interrupt Priority Select.

This bit determines the level of timer 2 interrupts.

0 = Interrupts are level 1 (high priority) requests.
1 = Interrupts are level 2 (low priority) requests.

Bit 7 **T2 STEST.** Timer 2 STEST.

This bit must be cleared to ensure proper operation.

Chapter 9

Serial Communications Interface (SCI) Module

This chapter discusses the architecture and programming of the serial communications interface module and covers the following topics:

Topic **Page**

9.1 SCI Overview

The serial communications interface (SCI) module is a programmable I/O port that facilitates digital communications between the TMS370 device and other asynchronous peripherals and uses the standard NRZ (nonreturn to zero) format. The SCI transmits and receives serial data, one bit at a time, at a programmable bit rate. Both the SCI receiver and transmitter are double-buffered and have their own separate enable and interrupt bits. They can be operated independently or simultaneously in the full duplex mode.

9.1.1 Key Features

The SCI has the following key features:

- Two communications formats:
  - Asynchronous
  - Isosynchronous
- Programmable bit rates to over 65,000 different speeds through a 16-bit baud select register
  - Asynchronous:
    - Range at 20 MHz—3 BPS to 156 KBPS
    - Number of bit rates—64K
  - Isosynchronous:
    - Range at 20 MHz—39 BPS to 2.5 MBPS
    - Number of bit rates—64K
- Programmable data word length from 1 to 8 bits
- Programmable stop bits of either 1 or 2 bits in length
- Error detection flags that ensure data integrity:
  - Parity error
  - Overrun error
  - Framing error
  - Break detect
- Two wake-up multiprocessor modes that can be used with either communications format:
  - Idle line wake-up
  - Address bit wake-up
- Full duplex operation

- ❑ Separate transmitter and receiver interrupts for polled or interrupt-driven operation
- ❑ Double-buffered receive and transmit functions
- ❑ Separate enable bits for the transmitter and receiver
- ❑ NRZ (nonreturn to zero) format

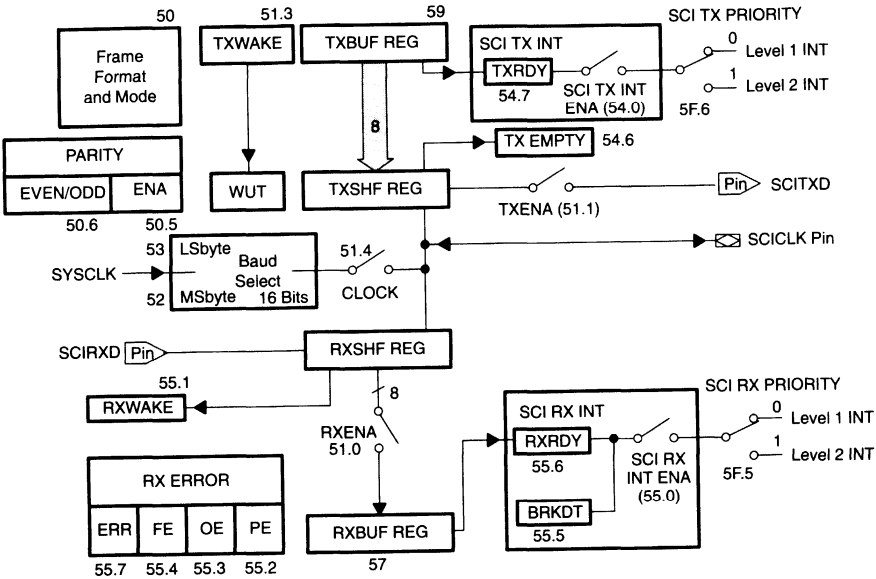
9.1.2 Physical Description

The major elements of the full-duplex SCI are shown in Figure 9–1 and include:

- ❑ A transmitter (SCITX)
  - ■ TXBUF — the transmitter buffer register that contains data, written by the CPU, to be transmitted
  - ■ TXSHF — the transmitter shift register that is loaded from TXBUF and shifts data onto the SCITXD pin, one bit at a time
- ❑ A receiver (SCIRX)
  - ■ RXSHF — the receiver shift register that shifts data in from the SCIRXD pin, one bit at a time
  - ■ RXBUF — the receiver buffer register that contains data that is to be read by the CPU and that is received from remote processor and loaded from RXSHF
- ❑ A programmable baud generator
- ❑ Memory-mapped control and status registers

The SCI receiver and transmitter can operate independently and simultaneously. A third port line, SCICLK, is available for the optional synchronizing clock line in the isosynchronous mode.

Figure 9–1. SCI Block Diagram



9.1.3 Communications Modes and Multiprocessing Modes

The SCI offers the following universal asynchronous receiver/transmitter (UART) communications modes for interfacing with many popular peripherals:

- Asynchronous mode (discussed in subsection 9.4.1) requires two lines to interface with many standard devices such as terminals and printers that use RS-232-C formats.
- Isosynchronous mode (discussed in subsection 9.4.2) permits high transmission rates and requires a synchronizing clock signal between the receiver and transmitter.

These modes can be programmed to contain:

- 1 start bit,
- 1 to 8 data bits,
- An even/odd parity bit or no parity bit, and
- 1 or 2 stop bits.

The SCI also has two multiprocessor modes: the idle line multiprocessor mode (see subsection 9.3.1) and the address bit multiprocessor mode (see subsection 9.3.2). These modes allow efficient data transfer between multiple processors and can be used with either the isosynchronous or standard asynchronous formats.

9.1.4 Control Registers

The SCI control registers are located at addresses 1050h to 105Fh and occupy peripheral file frame 5. The function of each location is shown in Table 9–1.

Table 9–1. SCI Memory Map

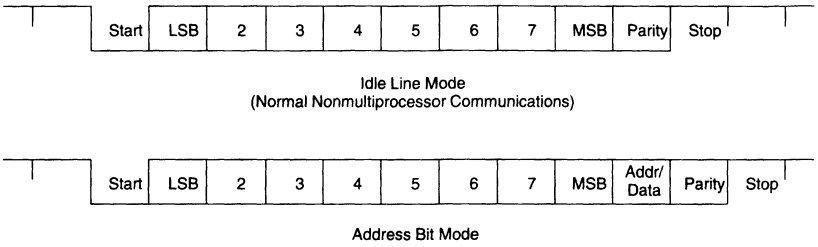
Peripheral File Location	Symbol	Name	Description
P050	SCICCR	SCI Communication Control Register	Defines the character format, protocol, and communications mode used by the SCI.
P051	SCICTL	SCI Control Register	Controls the RX/TX enable, TXWAKE and SLEEP functions, internal clock enable, and the SCI software reset.
P052	BAUD MSB	Baud Select MSbyte Register	Stores the data required to generate the bit rate.
P053	BAUD LSB	Baud Select LSbyte Register	
P054	TXCTL	SCI Transmitter Interrupt Control and Status Register	Contains the transmitter interrupt enable, the transmitter ready flag, and the transmitter empty flag.
P055	RXCTL	SCI Receiver Interrupt Control and Status Register	Contains one interrupt enable bit and seven receiver status flags.
P056		Reserved	
P057	RXBUF	SCI Receiver Data Buffer	Contains the current data from the receiver shift register.
P058		Reserved	
P059	TXBUF	SCI Transmit Data Buffer	Stores data bits to be transmitted by the SCITX.
P05A–P05C		Reserved	
P05D	SCIPC1	SCI Port Control Register 1	Controls the SCICLK pin functions.
P05E	SCIPC2	SCI Port Control Register 2	Controls the SCIRXD and SCITXD pin functions.
P05F	SCIPRI	SCI Interrupt Priority Control Register	Contains the receiver and transmitter interrupt priority select bits.

9.2 Programmable Data Format

SCI data, both receive and transmit, is in NRZ (nonreturn to zero) format. The NRZ data format is illustrated in Figure 9–2 and consists of:

- 1 start bit
- 1 to 8 data bits
- An even/odd parity bit (optional)
- 1 or 2 stop bits
- An extra bit to distinguish addresses from data (address bit mode only).

Figure 9–2. SCI Data Formats



To program the data format, use the SCICCR register (described in subsection 9.8.1). The bits that you use to program the data format are shown in Table 9–2:

9 *Table 9–2. Programming the Data Format Using SCICCR*

Bit Name	Designation	Function
SCI CHAR0–2	SCICCR.0–2	Select the character (data) length (1 to 8 bits). Refer to the bit listings on page 9-20 for additional information.
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.

9.3 Multiprocessor Communications

The multiprocessor communication format allows one processor to efficiently send blocks of data to other processors on the same serial link. You can have 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 SLEEP bit (SCICTL.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, 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 the 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:

- The **address bit mode** adds an extra 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 speeds, the program is not fast enough to avoid a 10-bit idle in the transmission stream.
- The **idle line mode** 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 in handling blocks that contain more than 10 bytes of data.

You can select the multiprocessor mode via the ADDRESS/IDLE WUP bit (SCICCR.3). Both modes use the TXWAKE flag bit (SCICTL.3) and the SLEEP flag bit (SCICTL.2) to control the SCITX and SCIRX features of these modes.

In both multiprocessor modes, the interrupt sequence is:

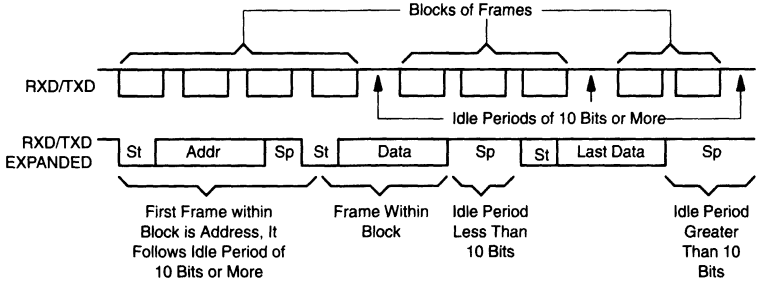
1) The SCI port wakes up (requests an interrupt) at the start of a block and reads the first frame that contains the destination address.
2) A software routine is entered through the interrupt and checks the incoming byte against its device address byte stored in memory.

3) If the block is addressed to the microcomputer, 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 SCI interrupts until the next block start.

9.3.1 Idle Line Multiprocessor Mode

In the idle line multiprocessor mode, blocks are separated by having a longer idle time between the blocks than between frames in the blocks. An idle time of 10 or more bits after a frame indicates the start of a new block. The idle line multiprocessor communication format is shown in Figure 9–3.

Figure 9–3. Idle Line Multiprocessor Communication Format



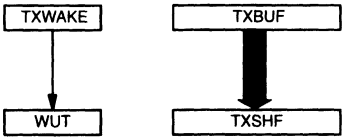
Note: In the figure, "St" = start and "Sp" = stop

There are two ways to send a block start signal.

- The first method is to deliberately leave an idle time of 10 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.
- In the second method, the SCI port uses the TXWAKE bit (SCICTL.3) to send an idle time of exactly 11 bits. Therefore, the serial communications line is not idle any longer than necessary.

Associated with the TXWAKE bit is the wake-up temporary or WUT flag bit. WUT is an internal flag, double buffered with TXWAKE. When TXSHF is loaded form TXBUF, WUT is loaded from TXWAKE, and the TXWAKE bit is cleared to 0. This arrangement is shown in Figure 9–4.

Figure 9–4. Double-Buffered WUT and TXSHF



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

1) Write a 1 to the TXWAKE bit.

2) Write a data word (don't care) to TXBUF. (The first data word written is suppressed while the block start signal is sent out, and ignored after that.)

   When TXSHF is free again, TXBUF's contents are shifted to TXSHF, the TXWAKE value is shifted to WUT, and then the TXWAKE bit is cleared. If TXWAKE bit 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 an address value to the TXBUF.

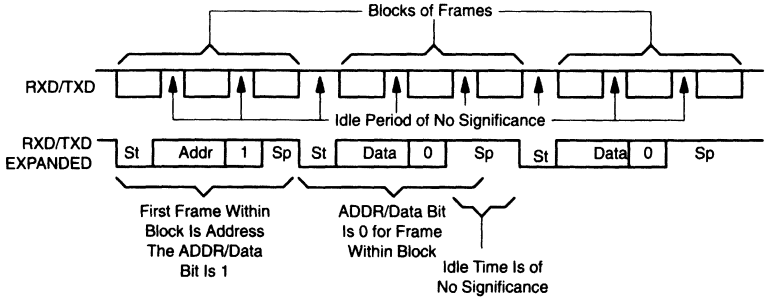
The receiver operates regardless of the SLEEP bit. The receiver does not set RXRDY, RXINT, or the error status bits until an address frame is detected.

9.3.2 Address Bit Multiprocessor Mode

In the address bit mode, 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.

The TXWAKE bit sets the address bit. In SCITX, when the TXBUF and TXWAKE are loaded into TXSHF and WUT, TXWAKE is reset to 0, and WUT is the value of the address bit of the current frame. Thus, to send an address, set the TXWAKE bit to a 1 and write the appropriate address value to the TXBUF. 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. Since TXSHF and WUT are both double-buffered, TXBUF and TXWAKE can be written to immediately after TXSHF and WUT are loaded. To transmit nonaddress frames in the block, leave the TXWAKE bit at 0.

Figure 9–5. Address Bit Multiprocessor Communication Format



Note: In the figure, "St" = start and "Sp" = stop

9.4 Communications Modes

The SCIRX/SCITX (receiver/transmitter) has two operating modes: asynchronous and isosynchronous. The ASYNC/ISOSYNC bit (SCICCR.4) determines the mode of operation. Either of these two modes can be used with either of the two forms of multiprocessor protocol: idle line and address bit.

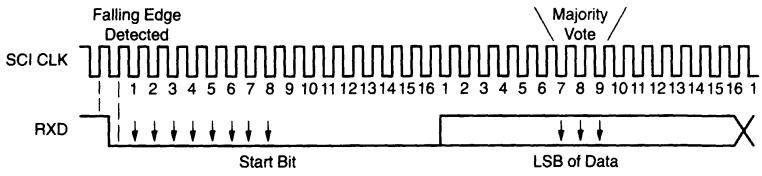
9.4.1 Asynchronous Communications Mode

The SCI asynchronous communication mode 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. There are 16 SCICLK periods per data bit.

The receiver begins operation on receipt of a valid start bit. A valid start bit consists of eight consecutive zero bits. 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 seventh, eighth, and ninth SCICLK period and are read on a majority (two out of three) basis. Figure 9–6 illustrates the asynchronous communication format, with a start bit showing how edges are found and where a majority vote is taken.

Figure 9–6. Asynchronous Communication Format



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. If the CLOCK bit (SCICTL.4) and SCICLK FUNCTION bit (SCIPC1.1) are set, then the serial clock is output continuously on the SCICLK pin.

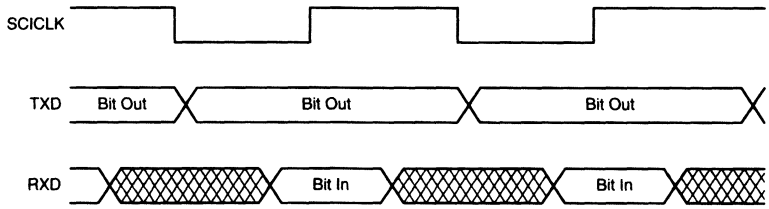
9.4.2 Isosynchronous Communications Mode

The SCI isosynchronous communication mode uses either two-line (one-way) or three-line (two-way) communications. The extra line (serial clock) in each case is required for data synchronization. In the isosynchronous mode, each bit of data requires only one serial clock pulse for transmission or reception. Thus, the data bit period equals the SCICLK period, and data bits are read on a single sample basis.

Since the receiver does not synchronize itself to data bits, the transmitter and receiver must be supplied with a common serial clock. If the internal serial clock is used, it must be output continuously on the SCICLK pin. The arrival of a valid start bit, which consists of a low on the RXD line at the time of a rising SCICLK edge, initiates receiver operation.

Figure 9–7 illustrates the isosynchronous communication format. A complete frame consists of a start bit, one to eight data bits, an optional even/odd parity bit, and one or two stop bits.

Figure 9–7. Isosynchronous Communication Format

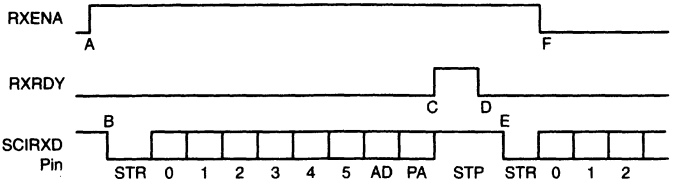


9.4.3 Receiver Signals in Communications Modes

Figure 9–8 illustrates receiver signal timing that assumes these conditions:

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

Figure 9–8. SCI RX Signals in Communications Modes



A) RX ENA goes high to enable the receiver.

B) Data arrives on the SCIRXD pin; start bit detected.

C) RXRDY goes high to signal that a new character has been received; data is shifted to RXBUF; an interrupt is requested.

D) The program reads the RXBUF register; RXRDY is automatically cleared.

E) The next byte of data arrives on the SCIRXD pin; start bit detected, then cleared.

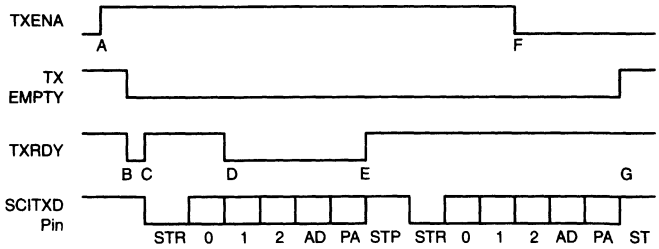
F) RX ENA goes low to disable the receiver; data continues to be assembled in the RXSHF register but is not transferred to the RXBUF register.

9.4.4 Transmitter Signals in Communications Modes

Figure 9–9 illustrates transmitter signal timing that assumes these conditions:

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

Figure 9–9. SCI TX Signals in Communications Modes



A) TX ENA goes high to enable the transmitter to send data.

B) Write to TXBUF; TX is no longer empty.

C) SCI transfers data to shift register; TX is ready for new character and requests an interrupt.

D) Program writes new character to TXBUF after TXRDY goes high (item C).

E) Finished transmitting first character; transfer new character to shift register.

F) TX ENA goes low to disable transmitter; SCI finishes transmitting current character.

G) Finished transmitting character; TX is empty and ready for new character.

9.5 Port Interrupts

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

- If the SCI RX INT ENA bit (RXCTL.0) 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 the RXSHF register to the RXBUF register. This action sets the RXRDY flag (RXCTL.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 (RXCTL.5) and initiates an interrupt.
- If the SCI TX INT ENA bit (TXCTL.0) is set, the transmitter interrupt is asserted whenever the data in the TXBUF register is transferred to the TXSHF register, indicating that the CPU can write to the TXBUF; this action sets the TXRDY flag bit (TXCTL.7) and initiates an interrupt.

SCI interrupts can be programmed onto different priority levels by the SCI RX PRIORITY (SCIPRI.5) and SCI TX 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; this reduces the possibility of receiver overrun.

9.6 Clock Sources

The SCI port can be driven by an internal or external baud generator. The CLOCK bit (SCICTL.4) configures the SCI clock source as either an input or an output:

- If an external clock source is selected (CLOCK = 0) and the SCICLK FUNCTION bit (SCIPC1.1) is set, the SCICLK pin functions as the high-impedance serial clock input pin.
- If an internal clock source is selected (CLOCK = 1), the SCICLK pin can be used as a general-purpose I/O pin or as the serial clock output pin. If the serial clock output is selected (SCICLK FUNCTION = 0), a 50-percent duty cycle clock signal is output on the SCICLK pin that makes it a serial clock output pin.

The SCI receives data on rising clock edges and transmits data on falling clock edges.

The internally generated serial clock is determined by the TMS370 CLKIN frequency and the baud select registers. The SCI uses the 16-bit value of the baud select registers to select one of 64K different serial clock rates for the communication modes in the following manner:

- Asynchronous Baud = CLKIN / [(BAUD REG + 1) × 128]

  BAUD REG = [CLKIN / (Asynchronous Baud × 128)] – 1
- Isosynchronous Baud = CLKIN / [(BAUD REG + 1)] × 8]

  BAUD REG = [CLKIN / (Isosynchronous Baud × 8)] – 1
- SCICLK frequency = CLKIN / [(BAUD REG + 1)] × 8]

  BAUD REG = [CLKIN / (SCICLK frequency × 8)] – 1

where

BAUD REG = The 16-bit value in the baud select registers.

Refer to Table 9–3.

Table 9–3. Asynchronous Baud Register Values for Common SCI Bit Rates

	Crystal Oscillator Frequency (MHz)							
	2.4576		7.3728		19.6608		20.0	
Baud	Baud Reg	% Error	Baud Reg	% Error	Baud Reg	% Error	Baud Reg	% Error
75	255	0.00	767	0.00	2047	0.00	2082	0.02
300	63	0.00	191	0.00	511	0.00	520	-0.03
600	31	0.00	95	0.00	255	0.00	259	0.16
1200	15	0.00	47	0.00	127	0.00	129	0.16
2400	7	0.00	23	0.00	63	0.00	64	0.16
4800	3	0.00	11	0.00	31	0.00	32	-1.38
9600	1	0.00	5	0.00	15	0.00	15	1.73
19200	0	0.00	2	0.00	7	0.00	7	1.73
38400	-	-	-	-	3	-	3	1.73
156000	-	-	-	-	-	-	0	0.16

Baud Reg = 16-bit baud register value

Note:

When the device is using an externally generated SCICLK in isosynchronous mode, the maximum speed at which the SCICLK can run is limited to CLKIN/40. This is necessary so that the internal clocks of the SCI have time to synchronize with the external clock. For this reason, it is recommended to use the TMS370 to drive the master serial clock in a system where maximum throughput is a major concern.

You can determine the current logic level on the SCICLK pin by reading the SCICLK DATA IN bit (SCIPC1.3).

9.7 Initialization Examples

This section contains two examples that initialize the serial port. In each example, the data is moved to and from the buffers in the interrupt routines.

1) The first example shows a typical RS-232 application that connects to a terminal.

2) The second example illustrates the address bit mode in a multiprocessor application.

In both examples, assume that the register mnemonics have been equated (EQU) with the corresponding peripheral-file location. For more examples using the TMS370 SCI, consult *Using the TMS370 SPI and SCI Modules Application Report.*

9.7.1 RS-232-C Example

This example initializes the transmitter and receiver to accept data at 9600 baud with a format of 8 data bits, 1 stop bit, and even parity.

```
B9600    .EQU   15                ;Value for counter for 9600 baud
                                  ;value = (CLKIN/128/baud) - 1 =
                                  ;(20 MHz/128/9600) - 1 = 15.27 ~ 15
                                  ;1.8 percent error
         AND    #01Fh, SCICTL     ;Make sure that SCI SW RESET bit is
                                  ;clear before writing to the SCI
                                  ;configuration registers
         MOV    #000h,SCIPRI      ;Set TX and RX to high priority
         MOV    #005h,SCIPC1      ;Set SCLK for general-purpose output
         MOV    #022h,SCIPC2      ;Set pins for RXD and TXD functions
         MOV    #Hi B9600,BAUDMSB ;Set bit rate for 9600 (MSbyte)
         MOV    #Lo B9600,BAUDLSB ;Set bit rate for 9600 (LSbyte)
         MOV    #077h,SCICCR      ;1 stop bit, even parity,
                                  ;and enable 8 data bits/char
         MOV    #033h,SCICTL      ;Enable Rx, Tx, clock is internal
         MOV    #001h,TXCTL       ;Enable TX interrupt
         MOV    #001h,RXCTL       ;Enable RX interrupt
         EINT                     ;Let the interrupts begin
         MOV    #00,TXBUF         ;Start transmitter by sending null
                                  ;character
```

9.7.2 RS-232-C Multiprocessor Mode Example

This example initializes the transmitter and receiver to accept data at 9600 baud with a format of 8 data bits, 1 stop bit, and even parity. It uses the address bit wake-up mode to implement the multiprocessor protocol.

```
B9600     .EQU    15                ;Value for counter for 9600 baud
                                    ;value = (CLKIN/128/baud) - 1 =
                                    ;(20 MHz/128/9600) - 1 = 15.27 ~ 15
                                    ;1.8 percent error
          MOV     #000h,SCIPRI      ;Set TX and RX to high priority
          MOV     #005h,SCIPC1      ;Set SCLK for general-purpose output
          MOV     #022h,SCIPC2      ;Set pins for RXD and TXD functions
          MOV     #Hi B9600,BAUDMSB ;Set bit rate for 9600 (MSbyte)
          MOV     #Lo B9600,BAUDLSB ;Set bit rate for 9600 (LSbyte)
          MOV     #07Fh,SCICCR      ;1 stop bit, even parity,
                                    ;and enable 8 data bits/char
          MOV     #037h,SCICTL      ;Enable Rx, Tx; RX to sleep,
                                    ;clock is internal
          MOV     #001h,TXCTL       ;Enable TX interrupt
          MOV     #001h,RXCTL       ;Enable RX interrupt
          EINT                      ;Let the interrupts begin
                                    ;
                                    ;MAIN ROUTINES
                                    ;

SENDADD   OR      #8,SCICTL         ;Main line routine; set TXWAKE
                                    ;wake bit
          MOV     ADDR,TXBUF        ;Transmit address stored in ADDR
          RTS

                                    ;INTERRUPT ROUTINES
                                    ;
                                    ;The locations of the SCI transmitter and
                                    ;receiver routines, SENDATA and GETDATA,
                                    ;need to be stored in the interrupt vec-
                                    tor
                                    ;table at locations 70F0h and 7FF2h,
                                    ;respectively.
                                    ;SCI TRANSMITTER INTERRUPT ROUTINE

SENDDATA  PUSH    A                 ;Address has already been sent by
                                    ;the SENDADD
          MOV     OUTDATA,TXBUF     ;Output character that is
                                    ;stored in DATA
          .                         ;
          .                         ;Other transmitter code
          .                         ;
          POP     A                 ;Restore and exit
          RTI
```

```
                                    ;SCI RECEIVER INTERRUPT ROUTINE

GETDATA   PUSH    A                 ;Receive a new character
          BTJZ    #2,RXCTL,ISDATA   ;Is this address or data byte?
          MOV     RXBUF,A           ;Get new character and clear
                                    ;interrupt flag
          CMP     #MYADDR,A         ;Is this my address or
                                    ;another processor's address
          JNE     RXEXIT            ;Exit if another's; still
                                    ;in sleep mode
          AND     #0FBh,SCICTL      ;If my address get out of sleep mode
          JMP     RXEXIT            ;Exit and wait for data
                                    ;
ISDATA    MOV     RXBUF,INDATA      ;Put incoming data in register
          .                         ;
          .                         ;Other receiver code
          .                         ;

RXEXIT    POP     P                 ;Restore and exit
          RTI
```

9.8 SCI Control Registers

The SCI is controlled and accessed through registers in peripheral file frame 5. These registers are listed in Table 9–4 and described in the following subsections. The bits shown in shaded boxes in Table 9–4 are privilege mode bits; that is, they can only be written to in the privilege mode.

Table 9–4. Peripheral File Frame 5: SCI Control Registers

Designation	ADDR	PF	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SCICCR	1050h	P050	STOP BITS	EVEN/ODD PARITY	PARITY ENABLE	ASYNC/ ISOSYNC	ADDRESS/ IDLE WUP	SCI CHAR2	SCI CHAR1	SCI CHAR0
SCICTL	1051h	P051	—	—	SCI SW RESET	CLOCK	TXWAKE	SLEEP	TXENA	RXENA
BAUD MSB	1052h	P052	BAUDF (MSB)	BAUDE	BAUDD	BAUDC	BAUDB	BAUDA	BAUD9	BAUD8
BAUD LSB	1053h	P053	BAUD7	BAUD6	BAUD5	BAUD4	BAUD3	BAUD2	BAUD1	BAUD0 (LSB)
TXCTL	1054h	P054	TXRDY	TX EMPTY	—	—	—	—	—	SCI TX INT ENA
RXCTL	1055h	P055	RX ERROR	RXRDY	BRKDT	FE	OE	PE	RXWAKE	SCI RX INT ENA
	1056h	P056	Reserved							
RXBUF	1057h	P057	RXDT7	RXDT6	RXDT5	RXDT4	RXDT3	RXDT2	RXDT1	RXDT0
	1058h	P058	Reserved							
TXBUF	1059h	P059	TXDT7	TXDT6	TXDT5	TXDT4	TXDT3	TXDT2	TXDT1	TXDT0
	105Ah	P05A	Reserved							
	105Bh	P05B								
	105Ch	P05C								
SCIPC1	105Dh	P05D	—	—	—	—	SCICLK DATA IN	SCICLK DATA OUT	SCICLK FUNCTION	SCICLK DATA DIR
SCIPC2	105Eh	P05E	SCITXD DATA IN	SCITXD DATA OUT	SCITXD FUNCTION	SCITXD DATA DIR	SCIRXD DATA IN	SCIRXD DATA OUT	SCIRXD FUNCTION	SCIRXD DATA DIR
SCIPRI	105Fh	P05F	SCI STEST	SCITX PRIORITY	SCIRX PRIORITY	SCI ESPEN	—	—	—	—

9.8.1 SCI Communication Control Register (SCICCR)

The SCICCR register defines the character format, protocol, and communications modes used by the SCI.

SCI Communication Control Register (SCICCR)
[Memory Address – 1050h]

Bit #	7	6	5	4	3	2	1	0
P050	STOP BITS	EVEN/ODD PARITY	PARITY ENABLE	ASYNC/ ISOSYNC	ADDRESS/ IDLE WUP	SCI CHAR2	SCI CHAR1	SCI CHAR0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R = Read, W = Write, *–n* = Value of the bit after the register is reset

Bits 0–2 **SCI CHAR0–2.** SCI Character Length Control Bits 0–2.

These bits select the SCI character (data) bit length, from 1 to 8 bits. Characters of less than 8 bits are right-justified in RXBUF and TXBUF, and are padded with leading 0s in RXBUF. TXBUF need not be padded with leading zeros.

Table 9–5. Character Bit Length

SCI CHAR2	SCI CHAR1	SCI 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

Bit 3 **ADDRESS/IDLE WUP.** SCI Multiprocessor Mode Control Bit.

This bit selects the multiprocessor mode.

0 = Selects idle line mode.
1 = Selects address bit mode.

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. Multiprocessor communication is different from the other communications modes because it uses TXWAKE and SLEEP functions.

Bit 4 **ASYNC/ISOSYNC.** SCI communications Mode Control Bit.

This bit determines the SCI communications mode.

0 = Selects isosynchronous mode. In this mode, the bit period is equal to the SCICLK period; bits are read on a single-sample basis.
1 = Selects asynchronous mode. In this mode, the bit period is 16 times the SCICLK period; bits are read on a two-out-of-three majority basis.

Bit 5 **PARITY ENABLE.** SCI Parity Enable.

This bit enables or disables the parity function. When parity is enabled during the address bit multiprocessor mode, the address bit is included in the parity calculation.

0 = Disables parity. No parity bit is generated during transmission or expected during reception.
1 = Enables parity.

Bit 6 **EVEN/ODD PARITY.** SCI Parity Enable.

If the PARITY ENABLE bit is set, this bit selects odd or even parity (odd or even number of bits in both transmitted and received characters).

0 = Sets odd parity.
1 = Sets even parity.

Bit 7 **STOP BITS.** SCI Number of Stop Bits.

This bit determines the number of stop bits transmitted. The receiver checks for one stop bit only.

0 = One stop bit.
1 = Two stop bits.

9.8.2 SCI Control Register (SCICTL)

The SCICTL register controls the RX/TX enable, TXWAKE and SLEEP functions, internal clock enable, and the SCI software reset.

SCI Control Register (SCICTL)
[Memory Address – 1051h]

Bit #	7	6	5	4	3	2	1	0
P051	—	—	SCI SW RESET	CLOCK	TXWAKE	SLEEP	TXENA	RXENA
			RW–0	RW–0	RS–0	RW–0	RW–0	RW–0

R = Read, W = Write, S = Set only, *–n* = Value of the bit after the register is reset

Bit 0 **RXENA.** SCI Receive Enable.

When this bit is set, received characters are transferred into RXBUF, and the RXRDY flag is set. When cleared, this bit prevents received characters from being transferred into the receiver buffer (RXBUF), and no receiver interrupts are generated. However, the receiver shift register continues to assemble characters. As a result, if RXENA is set during the reception of a character, the complete character is transferred into RXBUF.

0 = Disables SCI receiver.
1 = Enables SCI receiver.

Bit 1 **TXENA.** SCI Transmit Enable.

Data transmission through the SCITXD pin occurs only when this bit is set. If this bit is reset, the transmission is not halted until all the data previously written to TXBUF has been sent.

0 = Disables SCI transmitter.
1 = Enables SCI transmitter.

Bit 2. **SLEEP.** SCI Sleep.

This bit controls the receive features of the multiprocessor communication modes. You must clear this bit to bring the SCI out of sleep mode.

0 = Disables sleep mode.
1 = Enables sleep mode.

Bit 3 **TXWAKE.** SCI Transmitter Wake-up.

The TXWAKE bit controls the transmit features of the multiprocessor communication modes. This bit is cleared only by system reset. The SCI hardware clears this bit, once it has been transferred to wake-up temporary (WUT).

Bit 4 **CLOCK.** SCI Internal Clock Enable.

This bit determines the source of the SCICLK. Clearing this bit selects an external SCICLK, which is input on the high-impedance SCICLK line and bypasses the baud generator.

- For isosynchronous transactions, one bit is transmitted or received per SCICLK period.
- For asynchronous transactions, one bit is transmitted or received per 16 SCICLK periods.

The maximum frequency for the externally sourced SCICLK is CLKIN/16. Setting this bit selects an internal SCICLK, derived from the baud generator. This signal can be output on the SCICLK line.

0 = External SCICLK.
1 = Internal SCICLK.

Bit 5 **SCI SW RESET.** SCI Software Reset (Active Low).

Writing a 0 to this bit initializes the SCI state machines and operation flags to the reset condition. The CLOCK bit retains its state prior to the assertion of SCI SW RESET. If SCICLK is configured as an output, the SCICLK resets (low level). All affected logic is held in the reset state until a 1 is written to the SCI SW RESET bit. Thus, after a system reset, you must re-enable the SCI by writing a 1 to this bit. This bit must be cleared after a receiver break detect.

SCI SW RESET affects the operating flags of the SCI. This bit does not affect the configuration bits, nor does it put in the reset values. The flags listed in Table 9–6 are set to the values shown when SCI SW RESET is cleared. The operating flags are frozen until the SCI SW RESET bit is set again.

Table 9–6. Flags Affected by SCI SW RESET

SCI Flag	Designation	Value After SCI SW RESET
TXRDY	TXCTL.7	1
TXEMPTY	TXCTL.6	1
RXWAKE	RXCTL.1	0
PE	RXCTL.2	0
OE	RXCTL.3	0
FE	RXCTL.4	0
BRKDT	RXCTL.5	0
RXRDY	RXCTL.6	0
RX ERROR	RXCTL.7	0

Note:

The SCI SW RESET bit must be cleared before the SCI configuration registers can be set up or altered. The application program should set up all configuration registers before it sets the SCI SW RESET bit.

Bits 6, 7 **Reserved.** Read data is indeterminate.

9.8.3 Baud Select Registers (BAUD MSB and BAUD LSB)

The BAUD MSB and BAUD LSB registers store the data required to generate the bit rate. The SCI uses the combined 16-bit value, BAUD REG, of the baud select registers to set the SCI clock frequency as follows:

SCICLK frequency = CLKIN / [(BAUD REG + 1) × 8]

where

BAUD REG = The 16-bit value in the baud select registers.

For example, if the CLKIN frequency is 20 MHz, the maximum internal SCICLK frequency would be [20 MHz / 8] or 2.5 MHz.

- For asynchronous mode communication, data is transmitted and received at the rate of one bit for each 16 SCICLK periods.
- For isosynchronous mode communication, data is transmitted and received at the rate of one bit for each SCICLK period.

The asynchronous and isosynchronous bit rates are calculated as follows:

Asynchronous Baud = CLKIN / [(BAUD REG + 1) × 128]

Isosynchronous Baud = CLKIN / [(BAUD REG + 1) × 8]

Baud Select Register (BAUD MSB)
[Memory Address – 1052h]

Bit #	7	6	5	4	3	2	1	0
P052	BAUDF (MSB)	BAUDE	BAUDD	BAUDC	BAUDB	BAUDA	BAUD9	BAUD8
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

Baud Select Register (BAUD LSB)
[Memory Address – 1053h]

Bit #	7	6	5	4	3	2	1	0
P053	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

R = Read, W = Write, –*n* = Value of the bit after the register is reset

9.8.4 SCI Transmitter Interrupt Control and Status Register (TXCTL)

The TXCTL register contains the transmitter interrupt enable bit, the transmitter ready flag, and the transmitter empty flag. The status flags are updated each time a compete character is transmitted.

SCI Transmitter Interrupt Control and Status Register (TXCTL)
[Memory Address – 1054h]

Bit #	7	6	5	4	3	2	1	0
P054	TXRDY	TX EMPTY	—	—	—	—	—	SCI TX INT ENA
	R–1	R–1						RW–0

R = Read, W = Write, *–n* = Value of the bit after the register is reset

Bit 0 **SCI TX INT ENA.** SCI Transmitter Ready Interrupt Enable.

This bit controls the ability of the TXRDY bit to request an interrupt but does not prevent the TXRDY bit from being set. The SCI TX INT ENA bit is set to 0 by a system reset.

0 = Disables SCI TXRDY interrupt.
1 = Enables SCI TXRDY interrupt.

Bits 1–5 **Reserved.** Read data is indeterminate.

Bit 6 **TX EMPTY.** SCI Transmitter Empty.

This bit indicates the status of the transmitter-shift register and the TXBUF register. TX EMPTY is set to 1 by an SCI SW RESET or by a system reset.

0 = The CPU has written data to the TXBUF register; the data has not been completely transmitted.
1 = TXBUF and TXSHF registers are empty.

Bit 7 **TXRDY.** SCI Transmitter Ready.

The TXRDY bit is set by the transmitter to indicate that TXBUF is ready to receive another character. The bit is automatically cleared when a character is loaded into TXBUF. This flag asserts a transmitter interrupt if the interrupt enable bit SCI TX INT ENA (TXCTL.0) is set. TXRDY is a read-only flag. It is set to 1 by an SCI SW RESET or by a system reset.

0 = TXBUF is full.
1 = TXBUF is ready to receive a character.

9.8.5 SCI Receiver Interrupt Control and Status Register (RXCTL)

The RXCTL register contains one interrupt enable bit and seven receiver status flags (two of which can generate interrupt requests). The status flags are updated each time a complete character is transferred to the RXBUF. They are cleared each time RXBUF is read.

SCI Receiver Interrupt Control and Status Register (RXCTL)
[Memory Address – 1055h]

Bit #	7	6	5	4	3	2	1	0
P055	RX ERROR	RXRDY	BRKDT	FE	OE	PE	RXWAKE	SCI RX INT ENA
	R–0	R–0	R–0	R–0	R–0	R–0	R–0	RW–0

R = Read, W = Write, –*n* = Value of the bit after the register is reset

Bit 0 **SCI RX INT ENA.** SCI Receiver Interrupt Enable.

The SCI RX INT ENA bit controls the ability of the RXRDY and the BRKDT bits to request an interrupt but does not prevent these flags from being set.

0 = Disables RXRDY/BRKDT interrupt.
1 = Enables RXRDY/BRKDT interrupt.

Bit 1 **RXWAKE.** Receiver Wake-Up Detect.

The SCI sets this bit when a receiver wake-up condition is detected. In the address bit multiprocessor mode, RXWAKE reflects the value of the address bit for the character contained in RXBUF. In the idle line multiprocessor mode, RXWAKE is set if an idle SCIRXD line is detected. RXWAKE is a read-only flag. It is cleared by transfer of the first byte after the address byte to RXBUF, by reading the address character in RXBUF, by an SCI SW RESET, or by a system reset.

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 parity checker includes the address bit in the calculation. If parity generation and detection are not enabled, the PE flag is disabled and read as 0. The PE bit is reset by an SCI SW RESET, by a system reset, or by reading RXBUF.

0 = No parity error or parity is disabled.
1 = Parity error detected.

Bit 3. **OE.** SCI Overrun Error Flag.

The SCI sets this bit when a character is transferred into RXBUF before the previous character has been read out. The previous character is overwritten and lost. The OE flag is reset by an SCI SW RESET, by a system reset, or by reading RXBUF.

0 = No Overrun error detected.
1 = Overrun error detected.

Bit 4 **FE.** SCI Framing Error Flag.

The SCI sets this bit when it doesn't find a stop bit that it expects. 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 an SCI SW RESET, by a system reset, or by reading RXBUF.

0 = No framing error detected.
1 = Framing error detected.

Bit 5 **BRKDT.** SCI Break Detect Flag.

The SCI sets this bit when a break condition occurs. A break condition occurs when the SCIRXD line remains continuously low for at least 10 bits, beginning after a missing first stop bit. The occurrence of a break causes a receiver interrupt to be generated if the SCI RX 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 SCI 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 by reset through toggling the SCI SW RESET bit or by a system reset.

Bit 6 **RXRDY.** SCI Receiver Ready.

The receiver sets this bit to indicate that RXBUF is ready with a new character and clears the bit when the character is read. A receiver interrupt is generated if the SCI RX INT ENA bit is a 1. RXRDY is reset by an SCI SW RESET or by a system 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. It is a logical OR of the parity, overrun, framing error, and break detect flags. The bit can be used for fast error condition checking during the interrupt service routine because a negative value of the status register indicates that an error condition has occurred. This error flag cannot be cleared directly but is cleared if no individual error flags are set. This bit is cleared by an SCI SW RESET, by a system reset, or by reading RXBUF.

9.8.6 SCI Receiver Data Buffer Register (RXBUF)

The RXBUF register contains current data from the receiver shift register. RXBUF is cleared by a system reset.

SCI Receiver Data Buffer Register (RXBUF)
[Memory Address – 1057h]

Bit #	7	6	5	4	3	2	1	0
P057	RXDT7	RXDT6	RXDT5	RXDT4	RXDT3	RXDT2	RXDT1	RXDT0
	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0

R = Read, –*n* = Value of the bit after the register is reset

9.8.7 SCI Transmitter Data Buffer Register (TXBUF)

The TXBUF register is a read/write register that stores data bits to be transmitted by SCITX. Data written to TXBUF must be right-justified because the left-most bits are ignored for characters less than eight bits long.

SCI Transmit Data Buffer Register (TXBUF)
[Memory Address – 1059h]

Bit #	7	6	5	4	3	2	1	0
P059	TXDT7	TXDT6	TXDT5	TXDT4	TXDT3	TXDT2	TXDT1	TXDT0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R = Read, W = Write, –*n* = Value of the bit after the register is reset

9.8.8 SCI Port Control Register 1 (SCIPC1)

The SCIPC1 register controls the SCICLK pin functions.

SCI Port Control Register 1 (SCIPC1)
[Memory Address – 105Dh]

Bit #	7	6	5	4	3	2	1	0
P05D	—	—	—	—	SCICLK DATA IN	SCICLK DATA OUT	SCICLK FUNCTION	SCICLK DATA DIR
					R–0	RW–0	RW–0	RW–0

R = Read, W = Write, *–n* = Value of the bit after the register is reset

BIT 0 **SCICLK DATA DIR.** SCICLK Data Direction.

This bit determines the data direction on the SCICLK pin if SCICLK has been configured as a general-purpose I/O pin.

0 = SCICLK pin is a general-purpose input pin.
1 = SCICLK pin is a general-purpose output pin.

Bit 1 **SCICLK FUNCTION.**

This bit defines the function of the SCICLK pin.

0 = SCICLK pin is a general-purpose digital I/O pin.
1 = SCICLK pin is the SCI serial clock pin.

Bit 2 **SCICLK DATA OUT.**

This bit contains the data to be output on the SCICLK pin if the following conditions are met:

- ❑ SCICLK pin is configured as general-purpose I/O.
- ❑ SCICLK pin data direction is defined as output.

Bit 3 **SCICLK DATA IN.**

The SCICLK DATA IN bit contains the current value on the SCICLK pin.

Bits 4–7 **Reserved.** Read data is indeterminate.

9.8.9 SCI Port Control Register 2 (SCIPC2)

The SCIPC2 register controls the SCIRXD and SCITXD pin functions.

SCI Port Control Register 2 (SCIPC2)
[Memory Address – 105Eh]

Bit #	7	6	5	4	3	2	1	0
P05E	SCITXD DATA IN	SCITXD DATA OUT	SCITXD FUNCTION	SCITXD DATA DIR	SCIRXD DATA IN	SCIRXD DATA OUT	SCIRXD FUNCTION	SCIRXD DATA DIR
	R–0	RW–0	RW–0	RW–0	R–0	RW–0	RW–0	RW–0

R = Read, W = Write, *–n* = Value of the bit after the register is reset

Bit 0 **SCIRXD DATA DIR.** SCIRXD Data Direction.

This bit determines the data direction on the SCIRXD pin if SCIRXD has been defined as a general-purpose I/O pin.

0 = SCIRXD pin is a general-purpose input pin.
1 = SCIRXD pin is a general-purpose output pin.

Bit 1 **SCIRXD FUNCTION.**

This bit defines the function of the SCIRXD pin.

0 = SCIRXD pin is a general-purpose digital I/O pin.
1 = SCIRXD pin is the SCI receiver pin.

Bit 2 **SCIRXD DATA OUT.**

This bit contains the data to be output on the SCIRXD pin if the following conditions are met:

- SCIRXD pin has been defined as a general-purpose I/O pin.
- SCIRXD pin data direction has been defined as output.

Bit 3 **SCIRXD DATA IN.**

This bit contains the current value on the SCIRXD pin.

Bit 4 **SCITXD DATA DIR.** SCITXD Data Direction.

This bit determines the data direction on the SCITXD pin if SCITXD has been defined as a general-purpose I/O pin.

0 = SCITXD pin is a general-purpose input pin.
1 = SCITXD pin is a general-purpose output pin.

Bit 5 **SCITXD FUNCTION.**

This bit defines the function of the SCITXD pin.

0 = SCITXD pin is a general-purpose digital I/O pin.
1 = SCITXD pin is the SCI transmit pin.

Bit 6 **SCITXD DATA OUT.**

This bit contains the data to be output on the SCITXD pin if the following conditions are met:

- ❑ SCITXD pin has been defined as a general-purpose I/O pin.
- ❑ SCITXD pin data direction has been defined as output.

Bit 7 **SCITXD DATA IN.**

This bit contains the current value on the SCITXD pin.

9.8.10 SCI Priority Control Register (SCIPRI)

The SCIPRI register contains the receiver and transmitter interrupt priority select bits. This register is read-only during normal operation but can be written to in the privilege mode.

SCI Priority Control Register (SCIPRI)
[Memory Address – 105Fh]

Bit #	7	6	5	4	3	2	1	0
P05F	SCI STEST	SCITX PRIORITY	SCIRX PRIORITY	SCI ESPEN	—	—	—	—
	RP–0	RP–0	RP–0	RP–0				

R = Read, W = Privilege write only, –*n* = Value of the bit after the register is reset

Bits 0–3 **Reserved.** Read data is indeterminate.

Bit 4 **SCI ESPEN.** SCI Emulator Suspend Enable.

This bit has no effect except when you are using the XDS emulator to debug a program. Then, this bit determines how the SCI operates when the program is suspended by an action such as a hardware or software breakpoint.

0 = When the emulator is suspended, the SCI continues to work until the current transmit or receive sequence is complete.
1 = When the emulator is suspended, the SCI state machine is frozen so that the state of the SCI can be examined at the point that the emulator was suspended.

Bit 5 **SCI RX PRIORITY.** SCI Receiver Interrupt Priority Select.

This bit assigns the interrupt priority level of the SCI receiver interrupts.

0 = Receiver interrupts are level 1 (high-priority) requests.
1 = Receiver interrupts are level 2 (low-priority) requests.

Bit 6 **SCI TX PRIORITY.** SCI Transmitter Interrupt Priority Select.

This bit assigns the interrupt priority level of the SCI transmitter interrupts.

0 = Transmitter interrupts are level 1 (high-priority) requests.
1 = Transmitter interrupts are level 2 (low-priority) requests.

Bit 7 **SCI STEST.** SCI STEST.

This bit must be cleared to ensure proper operation.

Chapter 10

Serial Peripheral Interface (SPI) Module

This chapter discusses the architecture and programming of the serial peripheral interface module and covers the following topics:

Topic **Page**

10.1 SPI Overview

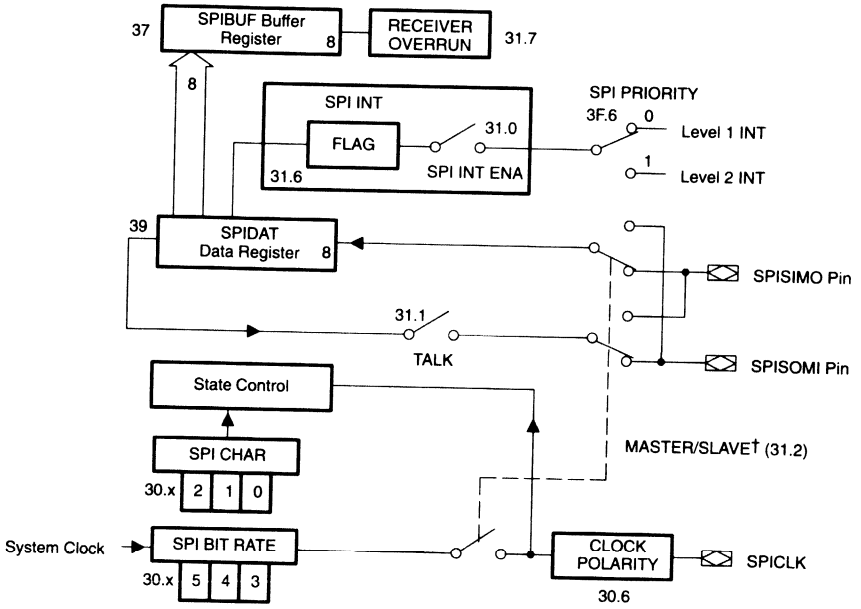
The SPI module is a high-speed synchronous serial I/O port that allows a serial bit stream of programmed length (one to eight bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI is normally used for communications between the microcontroller and external peripherals or another microcontroller. Typical applications include external I/O or peripheral expansion via devices such as shift registers, display drivers, and A/D converters. Multiprocessor communications are also supported by the master/slave operation of the SPI.

10.1.1 Physical Description

The SPI module, as shown in Figure 10–1, consists of:

- Three I/O pins:
  - SPISIMO—SPI slave in, master out
  - SPISOMI—SPI slave out, master in
  - SPICLK—SPI clock
- SPIBUF—the buffer register that contains the data received from the network that is ready for the CPU to read
- SPIDAT—the data shift register that serves as the transmit/receive shift register
- State control logic
- Memory-mapped control and status registers

Figure 10–1. SPI Block Diagram†



† The diagram is shown in slave mode.

10

10.1.2 Control Registers

The SPI control registers are located at addresses 1030h to 103Fh and occupy peripheral file frame 3. The function of each location is shown in Table 10–1.

Table 10–1. SPI Memory Map

Peripheral File Location	Symbol	Name	Description
P030	SPICCR	SPI Configuration Control Register	Controls the set-up of the SPI for operation.
P031	SPICTL	SPI Operation Control Register	Controls data transmission, the SPI's ability to generate interrupts, and the operating mode (slave or master).
P032–P036		Reserved	
P037	SPIBUF	Serial Input Buffer	Contains the data received from the network that is ready for the CPU to read.
P038		Reserved	
P039	SPIDAT	Serial Data Register	Serves as the transmit/receive shift register.
P03A–P03C		Reserved	
P03D	SPIPC1	SPI Port Control Register 1	Controls the SPICLK pin functions.
P03E	SPIPC2	SPI Port Control Register 2	Controls the SPISOMI and SPISIMO pin functions.
P03F	SPIPRI	SPI Interrupt Priority Control Register	Selects the interrupt priority level of the SPI interrupt.

10.2 Communications Between the Master and the Slave

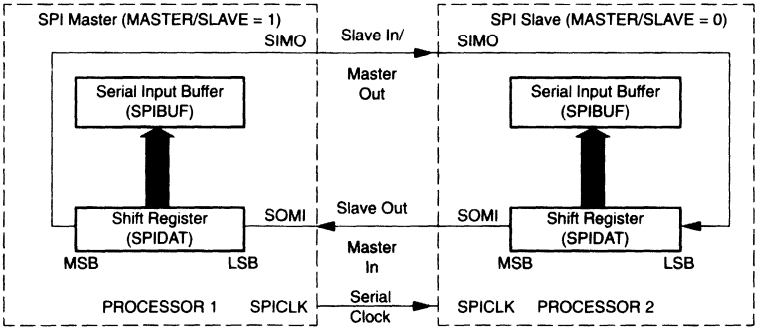
Figure 10–2 shows a typical connection of the SPI for communications between two microcontrollers: the master and the slave. The master initiates data transfer by sending the SPICLK signal. For both the slave and the master, data is shifted out of the shift registers on one edge of the clock and latched into the shift register on the opposite clock edge. As a result, both controllers send and receive data at the same time. The application software determines whether the data is meaningful or dummy data.

There are three possible cases 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. However, the software protocol determines how the master detects when the slave is ready to broadcast data.

Figure 10–2. SPI Master/Slave Connection



10.3 Operating Modes

The MASTER/SLAVE bit (SPICTL.2) selects the operating mode and the source of SPICLK. The SPI module can operate as a master or a slave.

10.3.1 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 on the first SPICLK edge and latched from the SPISOMI pin on the opposite edge of SPICLK.

The SPI BIT RATE0–2 bits of the SPICCR register determine the bit transfer rate for the network, both transmit and receive. Eight data transfer rates can be selected by these control bits as shown in Table 10–2 on page 10-10.

Data written to the SPIDAT register initiates data transmission on the SPISIMO pin, MSB first. Simultaneously, received data is shifted in the SPISOMI pin into the SPIDAT register. When the selected number of bits have been transmitted, the data is transferred to the SPIBUF (double-buffered receiver) for reading by the CPU to permit new transactions to take place. Data is shifted into the SPI MSB first. It is stored right-justified in SPIBUF.

To initiate a character transaction when the SPI is operating as a master, data must be written to the SPIDAT. When the specified number of data bits have been shifted through the SPIDAT register, the following events occur:

- ❑ The SPI INT FLAG bit (SPICTL.6) is set,
- ❑ The SPIDAT register contents transfer to the SPIBUF register, and
- ❑ If the SPI INT ENA bit (SPICTL.1) is set to 1, an interrupt is asserted.

Writing to the SPIDAT register before transmission is complete corrupts the current transmission.

10.3.2 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 CLKIN frequency divided by 32.

Data written to the SPIDAT register is transmitted to the network when the SPICLK is received from the network master. To receive data, the SPI waits for the network master to send SPICLK and then shifts the data on the SPISIMO pin into the SPIDAT register. If data is to be transmitted by the slave simultaneously, it must be written to the SPIDAT register before the beginning of SPICLK.

When the TALK bit (SPICTL.1) is cleared, data transmission is disabled, and the output line is put into a 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.

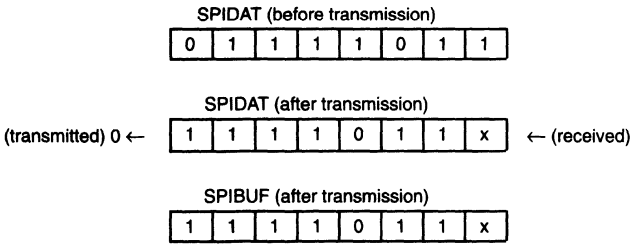
10.4 Data Format

Three character-length bits (SPICCR.2–0) specify the number of bits in the data character (1–8 bits). 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.

For characters with fewer than 8 bits:

- Data must be written to the SPIDAT register left-justified.
- Data must be read back from the SPIBUF register right-justified.
- The SPIBUF register contains the most recently received character, right-justified, plus any bits that are left over from previous transmission(s) and that have been shifted to the MSB position.

For example:
If the character length = 1 bit, and
the value written into SPIDAT = 07Bh,
then:



Note: x = 1 if SOMI is held high; x = 0 if SOMI is held low

10.5 Interrupts

The interrupt for the SPI is controlled by bits in two registers:

- The SPI INT ENA bit (SPICTL.0), when set, allows assertion of an interrupt request when an interrupt condition occurs.
- The SPI PRIORITY bit (SPIPRI.6) determines whether SPI interrupts are level 1 or level 2 priority requests.

When a complete character has been shifted into or out of the SPIBUF register, the SPI interrupt flag (SPI INT FLAG) of the SCICTL register is set, and an interrupt is generated if enabled by SPI INT ENA bit (SPICTL.0). The interrupt flag remains set until it is cleared by one of the following four events:

- The CPU reads the SPI receiver buffer (SPIBUF),
- The CPU enters the halt or standby mode with an IDLE instruction,
- Software sets the SPI SW RESET bit (SPICCR.7), or
- A system reset occurs.

An interrupt request must be explicitly cleared by one of the four methods listed above to avoid generating another interrupt. An interrupt request can be temporarily disabled by clearing the SPI INT ENA bit. However, unless the SPI INT FLAG itself is cleared, the interrupt request will be reasserted when the enable bit (SPI INT ENA) is again set to 1.

The priority level of the SPI interrupt is specified by the SPI PRIORITY bit (SPIPRI.6).

- If SPI PRIORITY = 0, a level 1 priority interrupt is generated.
- If SPI PRIORITY = 1, a level 2 priority interrupt is generated.

When the SPI INT FLAG bit is set, a character has been placed into the SPIBUF register 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 the SPIBUF, and the RECEIVER OVERRUN bit (SPICTL.7) is set. This indicates that the last character of data has been overwritten with new data before the previous character could be read.

10.6 Clock Sources

The CLOCK POLARITY bit (SPICCR.6) selects the active edge of the clock, either rising or falling.

- In the slave mode, the SPI clock is received from an external source and can be no greater than the CLKIN frequency divided by 32.
- In the master mode, the SPI clock is generated by the SPI and is output on the SPICLK pin.

The SPI BIT RATE0–2 bits (SPICCR.5–3) determine the bit transfer rate for sending and receiving the data. This transfer rate is defined by:

SPI BAUD RATE = CLKIN / (8×2^b)

where b=bit rate in SPICCR.5–3 (range 0–7).

Table 10–2 shows the bit rates for common crystal frequencies versus the SPI bit rate values.

Table 10–2. Common SPI Bit Rates

Crystal/Oscillator Frequency (MHz)						
SPI Value	**Divide by**	**20 MHz**	**12 MHz**	**10 MHz**	**5 MHz**	**2 MHz**
0	8	2500	1500	1250	625	250
1	16	1250	750	625	312.5	125
2	32	625	375	312.5	156.25	62.5
3	64	312.5	187.5	156.25	78.125	31.25
4	128	156.25	93.75	78.125	39.0625	15.625
5	256	78.125	46.875	39.0625	19.53125	7.8125
6	512	39.0625	23.4375	19.53125	9.765625	3.90625
7	1024	19.53125	11.71875	9.765625	4.882813	1.953125

Note: Bit rates are in KBPS.

10.7 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 SPICLK.
- ❑ Character length is assumed to be 1 bit.
- ❑ The SPI interrupts are disabled.
- ❑ Data in the SPI data register is set to 00h.

To change this SPI configuration:

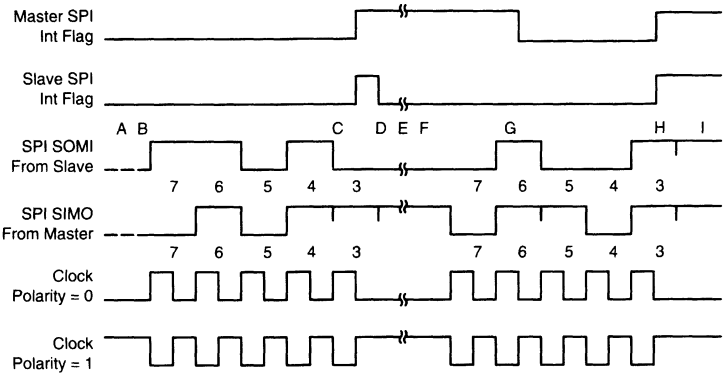
- ❑ Set the SPI SW RESET bit (SPICCR.7) to 1,
- ❑ Make any changes that you want in the configuration described above, and
- ❑ Clear the SPI SW RESET bit.

This prevents unwanted and unforeseen events from occurring during or as a result of mode change.

10.8 SPI Example

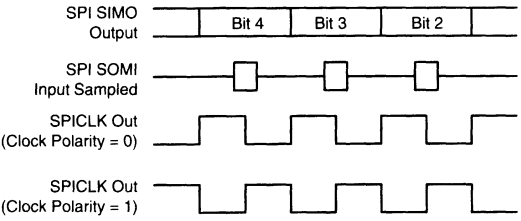
The following timing diagrams illustrate an SPI data transfer between two TMS370 devices using a character length of five bits. The lettered notes following the first diagram are keyed to the letter labels in the diagram.

5 Bits per Character



A. Slave writes 0D0h to SPIDAT and waits for the master to shift out the data.
B. Master writes 058h to SPIDAT, which starts the transmission procedure.
C. First byte is finished and sets the interrupt flags.
D. Slave reads 0Bh from its SPIBUF register (right justified).
E. Slave writes 04Ch to SPIDAT and waits for the master to shift out the data.
F. Master writes 06Ch to SPIDAT, which starts the transmission procedure.
G. Master reads 01Ah from the SPIBUF register (right justified).
H. Second byte is finished and sets the interrupt flags.
I. Master receives 09h and the slave receives a 0Dh (right justified).

Signals Connecting to Master Processor



10.9 SPI Control Registers

The SPI is controlled and accessed through registers in peripheral file frame 3. These registers are listed in Table 10–3 and described in the following subsections. The bits shown in shaded boxes in Table 10–3 are privilege mode bits; that is, they can be written to only in the privilege mode.

Table 10–3. Peripheral File Frame 3: SPI Control Registers

Designation	ADDR	PF	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SPICCR	1030h	P030	SPI SW RESET	CLOCK POLARITY	SPI BIT RATE2	SPI BIT RATE1	SPI BIT RATE0	SPI CHAR2	SPI CHAR1	SPI CHAR0
SPICTL	1031h	P031	RECEIVER OVERRUN	SPI INT FLAG	—	—	—	MASTER/ SLAVE	TALK	SPI INT ENA
	1032h to 1036h	P032 to P036	Reserved							
SPIBUF	1037h	P037	RCVD7	RCVD6	RCVD5	RCVD4	RCVD3	RCVD2	RCVD1	RCVD0
	1038h	P038	Reserved							
SPIDAT	1039h	P039	SDAT7	SDAT6	SDAT5	SDAT4	SDAT3	SDAT2	SDAT1	SDAT0
	103Ah to 103Ch	P03A to P03C	Reserved							
SPIPC1	103Dh	P03D	—	—	—	—	SPICLK DATA IN	SPICLK DATA OUT	SPICLK FUNCTION	SPICLK DATA DIR
SPIPC2	103Eh	P03E	SPISIMO DATA IN	SPISIMO DATA OUT	SPISIMO FUNCTION	SPISIMO DATA DIR	SPISOMI DATA IN	SPISOMI DATA OUT	SPISOMI FUNCTION	SPISOMI DATA DIR
SPIPRI	103Fh	P03F	SPI STEST	SPI PRIORITY	SPI ESPEN	—	—	—	—	—

10.9.1 SPI Configuration Control Register (SPICCR)

The SPICCR register controls the set-up of the SPI for operation.

SPI Configuration Control Register (SPICCR)
[Memory Address – 1030h]

Bit #	7	6	5	4	3	2	1	0
P030	SPI SW RESET	CLOCK POLARITY	SPI BIT RATE2	SPI BIT RATE1	SPI BIT RATE0	SPI CHAR2	SPI CHAR1	SPI CHAR0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R = Read, W = Write, *–n* = Value of the bit after the register is reset

Bits 0–2 **SPI CHAR0–2**. Character Length Control Bits 0–2.

These three bits determine the number of bits to be shifted in or out as a single character during one shift sequence. The value of these bits is represented in the following table.

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

Bits 3–5 **SPI BIT RATE0–2.** SPI Bit Rate Control Bits 0–2.

These bits determine the bit transfer rate if the SPI is the network master. There are eight data transfer rates (each a function of the system clock) that can be selected. The system clock is divided by an 8-bit, free-running prescaler from which eight taps are available for use as the shift clock. One data bit is shifted per SPICLK cycle.

SPI BIT RATE2†	SPI BIT RATE1†	SPI BIT RATE0†	SPI Clock Frequency
0	0	0	CLKIN/8
0	0	1	CLKIN/16
0	1	0	CLKIN/32
0	1	1	CLKIN/64
1	0	0	CLKIN/128
1	0	1	CLKIN/256
1	1	0	CLKIN/512
1	1	1	CLKIN/1024

† If the SPI is a network slave, then the module receives a clock on the SPICLK pin from the network master, and these bits have no effect on SPICLK. The frequency of the input clock should be no greater than the CLKIN frequency divided by 32.

Bit 6 **CLOCK POLARITY**. Shift Clock Polarity.

The CLOCK POLARITY bit controls the polarity of the SPICLK signal.

0 = The inactive level is low; data is output by the rising edge of SPICLK; input data is latched by the falling edge of SPICLK.

1 = The inactive level is high; data is output by the falling edge of SPICLK; input data is latched by the rising edge of SPICLK.

Bit 7 **SPI SW RESET**. SPI Software Reset.

- Writing a 1 to this bit initializes the SPI circuitry and operating flags to the reset condition. Specifically, the RECEIVER OVERRUN and SPI INT FLAG flags are cleared. The SPI configuration remains unchanged. If the module is operating as a master, the SPICLK output level returns to its inactive level.
- When a 0 is written to SPI SW RESET, the SPI is ready to transmit or receive the next character. A character written to the transmitter when SPI SW RESET is a 1 will not be shifted out when SPI SW RESET bit is cleared. A new character must be written to the serial data register. To change any configuration bits, use this bit (see Section 10.7).

10.9.2 SPI Operation Control Register (SPICTL)

The SPI operation control register controls data transmission, the SPI's ability to generate interrupts, and the operating mode (slave or master).

SPI Operation Control Register (SPICTL)
[Memory Address – 1031h]

Bit #	7	6	5	4	3	2	1	0
P031	RECEIVER OVERRUN	SPI INT FLAG	—	—	—	MASTER/ SLAVE	TALK	SPI INT ENA
	R–0	R–0				RW–0	RW–0	RW–0

R = Read, W = Write, *–n* = Value of the bit after the register is reset

Bit 0 **SPI INT ENA**. SPI Interrupt Enable.

This bit controls the SPI's ability to generate an interrupt. The SPI INT FLAG is unaffected by this bit.

0 = Disables interrupt.
1 = Enables interrupt.

Bit 1 **TALK**. Master/Slave Transmit Enable.

This bit disables data transmission (master or slave) by placing the serial data output in a high-impedance state. TALK is cleared (disabled) by a system reset.

0 = Disables transmission; if not programmed as a general-purpose I/O pin, the SPI serial output is in a high-impedance state.
1 = Enables transmission.

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 slave.

0 = SPI configured as a slave.
1 = SPI configured as a master.

Bits 3–5 **Reserved.** Read data is indeterminate.

Bit 6 **SPI INT FLAG**. Serial Peripheral Interrupt 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. A character received is placed in the receiver buffer at the time the SPI INT FLAG bit is set. The SPI INT FLAG is cleared when the receiver buffer is read. It is also cleared by an SPI software reset (SPI SW RESET) or by a system reset.

Bit 7 **RECEIVER OVERRUN.**

This bit is a read-only flag that the SPI hardware sets when a receive or transmit operation completes before the previous character has been read from the receive buffer. It indicates that the last received character has been overwritten and therefore has been lost. RECEIVER OVERRUN is cleared when the receiver buffer is read. It is also cleared by SPI SW RESET or by a system reset.

10.9.3 Serial Input Buffer (SPIBUF)

The SPIBUF register contains the data received from the network ready for the CPU to read.

Serial Input Buffer (SPIBUF)
[Memory Address – 1037h]

Bit #	7	6	5	4	3	2	1	0
P037	RCVD7 (MSB)	RCVD6	RCVD5	RCVD4	RCVD3	RCVD2	RCVD1	RCVD0 (LSB)
	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0

R = Read, *–n* = Value of the bit after the register is reset

Once the serial data register has received the complete character, the character is then transferred to the SPIBUF register, where it can be read. The SPI INT FLAG bit (SPICTL.6) is set to indicate that the data is available when the received character is transferred. Since data is shifted into the SPI most significant bit first, it is stored right-justified in the SPIBUF.

10.9.4 Serial Data Register (SPIDAT)

The SPIDAT register is the transmit/receive shift register. Data written to the 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.

Writing to the SPIDAT performs two functions:

- ❑ It provides data to be output on the serial output pin if the TALK bit is set.
- ❑ It initiates a transaction when the SPI is operating as a master.

A receiver sequence is initiated when dummy data is written to the register. Since the data is not hardware justified for characters that are shorter than eight bits, transmit data must be written in left-justified form and received data read in right-justified form.

Serial Data Register (SPIDAT)
[Memory Address – 1039h]

Bit #	7	6	5	4	3	2	1	0
P039	SDAT7	SDAT6	SDAT5	SDAT4	SDAT3	SDAT2	SDAT1	SDAT0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R = Read, W = Write, *–n* = Value of the bit after the register is reset

10.9.5 SPI Port Control Registers (SPIPC1 and SPIPC2)

Two port control registers (SPIPC1 and SPIPC2) allow you to control all of the functions for a SPI port pin in one write cycle. Each module pin is controlled by a nibble in one of the SPIPCs.

10.9.5.1 SPI Port Control Register 1 (SPIPC1)

This register controls the SPICLK pin.

Port Control Register 1 (SPIPC1)
[Memory Address – 103Dh]

Bit #	7	6	5	4	3	2	1	0
P03D	—	—	—	—	SPICLK DATA IN	SPICLK DATA OUT	SPICLK FUNCTION	SPICLK DATA DIR
					R–0	RW–0	RW–0	RW–0

R = Read, W = Write, *–n* = Value of the bit after the register is reset

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 I/O pin.

0 = SPICLK pin is a general-purpose input pin.
1 = SPICLK pin is a general-purpose output pin.

Bit 1 **SPICLK FUNCTION**. SPICLK Pin Function Select.

This bit defines the function of the SPICLK pin.

0 = SPICLK pin is a general-purpose digital I/O pin.
1 = SPICLK pin contains the SPI clock.

Bit 2 **SPICLK DATA OUT**. SPICLK 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 I/O pin.
- ❑ SPICLK pin data direction has been defined as output.

Bit 3 **SPICLK DATA IN**. SPICLK Pin Port Data In.

This bit contains the current value on the SPICLK pin, regardless of the mode. A write to this bit has no effect.

Bits 4–7 **Reserved.** Read data is indeterminate.

Note:

The SPICLK pin always functions as the SPICLK input pin in the slave mode (i.e., SPICTL.2=0), even if SPICLK FUNCTION = 0.

10.9.5.2 SPI Port Control Register 2 (SPIPC2)

The SPIPC2 register controls the SPISOMI and SPISIMO pin functions.

Port Control Register 2 (SPIPC2)
[Memory Address – 103Eh]

Bit #	7	6	5	4	3	2	1	0
P03E	SPISIMO DATA IN	SPISIMO DATA OUT	SPISIMO FUNCTION	SPISIMO DATA DIR	SPISOMI DATA IN	SPISOMI DATA OUT	SPISOMI FUNCTION	SPISOMI DATA DIR
	R–0	RW–0	RW–0	RW–0	R–0	RW–0	RW–0	RW–0

R = Read, W = Write, –*n* = Value of the bit after the register is reset

Bit 0 **SPISOMI DATA DIR**. SPISOMI Data Direction.

This bit determines the data direction on the SPISOMI pin if SPISOMI has been defined as a general-purpose I/O pin.

0 = SPISOMI pin is a general-purpose input pin.
1 = SPISOMI pin is a general-purpose output pin.

Bit 1 **SPISOMI FUNCTION**. SPISOMI Pin Function Select.

This bit defines the function of the SPISOMI pin. When SPISOMI is an input and SPISOMI FUNCTION and SPISOMI DATA DIR are disabled, SPICLK still clocks the internal circuitry.

0 = SPISOMI pin is a general-purpose digital I/O pin.
1 = SPISOMI pin contains the SPI data.

Bit 2 **SPISOMI DATA OUT**. SPISOMI Pin Data Out.

This bit contains the data to be output on the SPISOMI pin if the following conditions are met:

- SPISOMI pin has been defined as a general-purpose I/O pin.
- SPISOMI pin data direction has been defined as output.

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 4 **SPISIMO DATA DIR**. SPISIMO Data Direction.

This bit determines the data direction on the SPISIMO pin if SPISIMO has been defined as a general-purpose I/O pin.

0 = SPISIMO pin is a general-purpose input pin.
1 = SPISIMO pin is a general-purpose output pin.

Bit 5 **SPISIMO FUNCTION**. SPISIMO Pin Function Select.

This bit defines the function of the SPISIMO pin.

0 = SPISIMO pin is a general-purpose digital I/O pin.
1 = SPISIMO pin contains the SPI data.

Bit 6 **SPISIMO DATA OUT**. SPISIMO Pin Data Out.

This bit contains the data to be output on the SPISIMO pin if the following conditions are met:

- SPISIMO pin has been defined as a general-purpose I/O pin.
- SPISIMO pin data direction has been defined as output.

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.

10.9.6 SPI Interrupt Priority Control Register (SPIPRI)

The SPIPRI register selects the interrupt priority level of the SPI interrupt. The register is read only during normal operation but can be written to in the privilege mode.

SPI Interrupt Priority Control Register (SPIPRI)
[Memory Address – 103Fh]

Bit #	7	6	5	4	3	2	1	0
P03F	SPI STEST	SPI PRIORITY	SPI ESPEN	—	—	—	—	—
	RP–0	RP–0	RP–0					

R = Read, P = Privilege write only, –*n* = Value of the bit after the register is reset

Bits 0–4 **Reserved.** Read data is indeterminate.

Bit 5 **SPI ESPEN**. Emulator Suspend Enable.

This bit has no effect, except when you are using the XDS emulator to debug a program; then, 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 the state of the SPI is frozen so that it can be examined at the point that the emulator was suspended.

Bit 6 **SPI PRIORITY**. Interrupt Priority Select.

0 = Interrupts are level 1 (high-priority) requests.
1 = Interrupts are level 2 (low-priority) requests.

Bit 7 **SPI STEST.** SPI STEST.

This bit must be cleared to ensure proper operation.

Chapter 11

Analog-to-Digital Converter Module

This chapter discusses the architecture and programming of the analog-to-digital converter module and covers the following topics:

Topic **Page**

11.1 Analog-to-Digital Converter (A/D) Overview

The analog-to-digital (A/D) converter module is an 8-bit, successive approximation converter with internal sample-and-hold circuitry. The module has eight multiplexed analog input channels that allow the processor to convert the voltage levels from up to eight different sources.

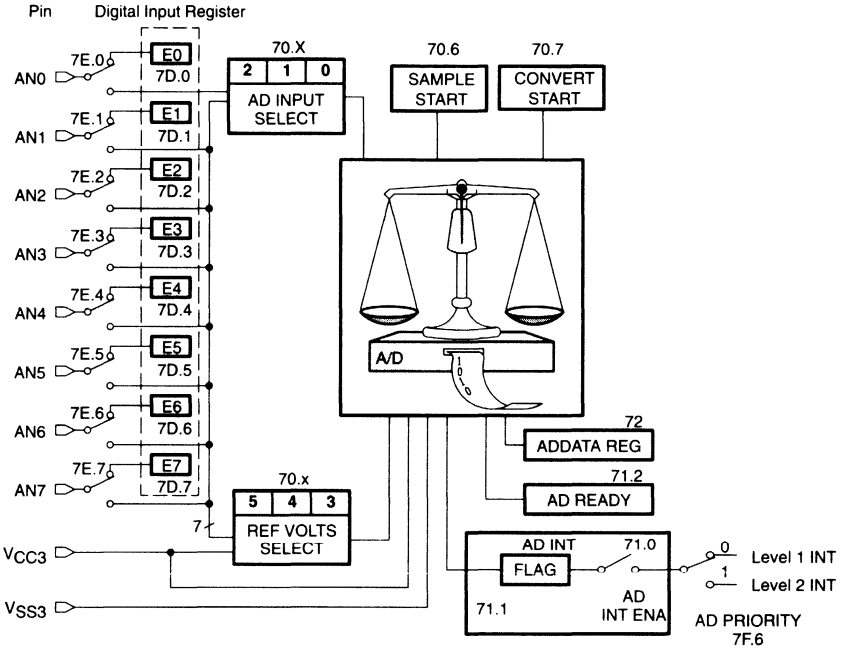
11.1.1 Physical Description

The A/D module, shown in Figure 11–1, consists of:

- Eight analog input channels (AN0–AN7), any of which can be software-configured as digital inputs (E0–E7) if not needed as analog channels
- An A/D input selector (INPUT)
- A $+V_{REF}$ input selector ($+V_{REF}$)
- The analog-to-digital converter (A/D)
- The ADDATA register, which contains the digital value of a completed conversion
- A/D module control registers

The input channels can be routed through either the channel selector or the positive voltage selector. The A/D converter then processes these signals and puts the result in the ADDATA register. The A/D interrupt circuit informs the rest of the system when a conversion is complete.

Figure 11–1. Analog-to-Digital Converter Block Diagram



11.1.2 Control Registers

The A/D control registers are located at addresses 1070h to 107Fh and occupy peripheral file frame 7, as shown in Table 11–1.

Table 11–1. A/D Memory Map

Peripheral File Location	Symbol	Name	Description
P070	ADCTL	Analog Control Register	Controls the input selection, reference voltage selection, sample start, and conversion start.
P071	ADSTAT	Analog Status and Interrupt Register	Indicates the converter and interrupt status.
P072	ADDATA	Analog Conversion Data Register	Contains the digital result of the last A/D conversion.
P073–P07C		Reserved	
P07D	ADIN	Analog Port E Data Input Register	Contains digital input data when one or more of the AN0–AN7 pins are used as digital ports.
P07E	ADENA	Analog Port E Input Enable Register	Controls the function of the AN0–AN7 pins.
P07F	ADPRI	Analog Interrupt Priority Register	Selects the interrupt priority level of the A/D interrupt.

11.2 A/D Operation

The following subsections describe the functions and options of the A/D module.

11.2.1 Input/Output Pins

The A/D module uses ten pins to connect itself to the external world: AN0–AN7, V_{CC3}, and V_{SS3}. These pins are described below:

- Eight (AN0–AN7) of the ten pins are analog channels and can be individually configured as general-purpose input pins when not used as analog inputs.

  Seven (AN1–AN7) of the eight analog channels are also available as the positive input voltage reference. This feature allows a weighted measurement or ratio of one channel to another.

- The analog voltage supply pins, V_{CC3} and V_{SS3}, isolate the A/D module from digital switching noise that can be present on the other power supply pins. This isolation provides a more accurate conversion.

  To further reduce noise and produce a more accurate conversion, you should run the power to the V_{CC3} and V_{SS3} pins on separate conductors from the other power lines. Additionally, the power conductors to the V_{CC3} and V_{SS3} should be as short as possible, and the two lines should be properly decoupled. Use other standard noise-reduction techniques to help provide a more accurate conversion.

Note that you can select the V_{REF} pin to be either V_{CC3} or one of the analog input channels AN1 to AN7. V_{CC3} must provide power to the A/D module even if it does not provide the voltage reference. A channel configured as the $+V_{REF}$ for one conversion can be changed to an analog input channel for the next conversion.

11.2.2 Sampling Time

The application program controls the length of the sample time, which provides the flexibility to optimize the conversion process for both high- and low-impedance sources. The program should wait 1 µs for each kilohm of source output impedance or a minimum of 1 µs for low-impedance sources.

11.2.3 A/D Conversion

The digital result of the conversion process is given in the following formula.

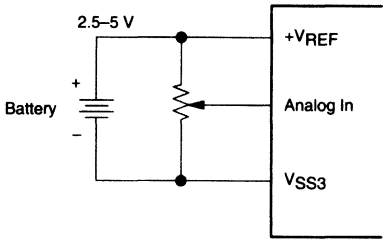
digital result = 255 × voltage of input / voltage of reference

The conversion process requires 164 cycles and results in a conversion time of 32.8 microseconds at 20 MHz. A maximum of 27,600 conversions per second are possible at 20 MHz, including setting up the conversion, sampling, converting, and saving the results.

In ratiometric conversions, the conversion value is a ratio of the V_{REF} source to the analog input. As V_{REF} is increased, the input voltage that is required in order to produce a certain conversion value changes; however, all conversion values keep the same relationship to V_{REF}. That is, one half of V_{REF} always results in the value 080h, regardless of the value of V_{REF} (assuming that V_{REF} is in the range of 2.5 to 5.5 volts above V_{SS3}).

Figure 11–2 shows an example of ratiometric conversion. In this example, the digital result of the conversion indicates the position of the potentiometer wiper, even if the battery loses voltage over time. The A/D conversion always gives the ratio of the resistor values on either side of the wiper, even if V_{REF} drops from 5.0 to 2.5 volts.

Figure 11–2. Ratiometric Conversion Example



11.2.4 Interrupts

The A/D module sets the AD INT FLAG bit (ADSTAT.1) at the end of the conversion process. If both the AD INT FLAG and the AD INT ENA bit (ADSTAT.0) are set, then the module generates an interrupt request. This interrupt request can be asserted on either the high priority level 1 or the lower priority level 2, depending on the AD PRIORITY bit (ADPRI.6).

The program must clear the AD INT FLAG bit, or else the same interrupt will cause the CPU to enter the interrupt routine again. If the AD INT ENA bit is cleared without clearing the flag, the interrupt is reasserted when the AD INT ENA bit is again set.

11.2.5 Programming Considerations

Follow these steps to obtain data from the A/D converter:

1) Write to the ADCTL register (described on page 11-10) to:
   a) Select the analog channel (ADCTL.2–0).
   b) Select the V_{REF} source (ADCTL.5–3).
   c) Set the SAMPLE START bit to 1 (ADCTL.6) to begin sampling.
2) Wait for the sample time to elapse. The program should wait 1 µs for each kilohm of source output impedance or a minimum of 1 µs for low impedance sources.
3) When the sample time completes, set the CONVERT START bit (ADCTL.7); leave the SAMPLE START bit (ADCTL.6) set.
4) Wait for either the interrupt flag to be set or the A/D interrupt to occur.
5) Read the conversion data register (ADDATA).
6) Clear the interrupt flag bit (ADSTAT.1).

Eighteen cycles after the program sets the CONVERT START bit, the A/D module clears both the SAMPLE START and CONVERT START bits to signify the end of the internal sampling phase. After these bits are cleared, the program can change the input channel without affecting the conversion process. The voltage reference source V_{REF} should remain constant throughout the conversion.

To stop a conversion in progress, set the SAMPLE START (ADCTL.6) bit to 1 anytime after the A/D clears this bit. The entire conversion process requires 164 system clock cycles after the program sets the CONVERT START bit (ADCTL.7).

11.3 A/D Example Program

This example program samples and converts data from all eight channels and stores the digital results into a table beginning at ATABLE. The routine stops interrupting the main program after it finishes all eight channels. If the main program wants more recent data, it needs to execute only the code at RESTART, and the A/D routine will again sample and convert all eight channels of data. The AD INT ENA bit (ADSTAT.0) is cleared by the A/D interrupt routine as a signal to the main program that all eight channels have been processed. The address of the label ATOD must be placed into the interrupt vector table located at 7FECh and 7FEDh.

```
ADCTL    .EQU  P070          ;A/D control register
ADSTAT   .EQU  P071          ;A/D status register
ADDATA   .EQU  P072          ;A/D conversion results
ADENA    .EQU  P07E          ;A/D input enable
         .REG  ADCHANL       ;keeps current channel number
         .REG  ATABLE,8      ;8 byte table that stores
                             ; channel data, LSB first
;
INIT     MOV   #0,ADENA      ;all channels to A/D inputs
                             ; (reset condition)
         CALL  RESTART       ;start taking data
;
;        MAIN PROGRAM GOES HERE
;        .
;        .
         CALL  RESTART       ;start taking more data
;        .
;        .
;        MORE MAIN PROGRAM
;
;        SUBROUTINE SECTION
RESTART  CLR   ADCHANL       ;initialize channel
         MOV   #001h,ADSTAT  ;enable interrupts, clear
                             ; any flag
         MOV   #040h,ADCTL   ;start sampling (approx. 2 µs
                             ; delay)
         MOV   #0C0h,ADCTL   ;start converting now; enter
                             ; main program
         RTS
;
;        INTERRUPT ROUTINE FOR ANALOG TO DIGITAL CONVERTER
ATOD     PUSH  A             ;save registers
         PUSH  B
         MOV   ADCHANL,B     ;get channel number
         MOV   ADDATA,A      ;get A/D conversion value
         MOV   A,ATABLE(B)   ;store in a table according to
                             ; channel number
         INC   B             ;point to next channel
         BTJZ  #8,B,GOCNVRT
;stop when all channels sampled
                             ;(bit3 =1)
         CLR   ADCHANL       ;reset the A/D channel
         MOV   #0,ADSTAT     ;turn off interrupt and
                             ; clear flag
         JMP   EXITA2D       ;all 8 channels taken, enable
                             ;set to 0 now
;
GOCNVRT  MOV   B,ADCHANL     ;store current A/D channel
         MOV   #01h,ADSTAT   ;clear interrupt flag
         OR    #040h,B
;set up sample bit in value
         MOV   B,ADCTL       ;start sampling channel data
         OR    #080h,ADCTL   ;start converting data
;
EXITA2D  POP   B             ;Restore data
         POP   A
         RTI
```

11.4 A/D Control Registers

The A/D module control registers occupy peripheral file frame 7, as shown in Table 11–2. The bits shown in shaded boxes in Table 11–2 are privilege mode bits; that is, they can be written only to in the privilege mode.

Table 11–2. Peripheral File Frame 7: A/D Converter Control Registers

Designation	ADDR	PF	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCTL	1070h	P070	CONVERT START	SAMPLE START	REF VOLT SELECT2	REF VOLT SELECT1	REF VOLT SELECT0	AD INPUT SELECT2	AD INPUT SELECT1	AD INPUT SELECT0
ADSTAT	1071h	P071	—	—	—	—	—	AD READY	AD INT FLAG	AD INT ENA
ADDATA	1072h	P072	A-to-D Conversion Data Register							
	1073h to 107Ch	P073 to P07C	Reserved							
ADIN	107Dh	P07D	Port E Data Input Register							
ADENA	107Eh	P07E	Port E Input Enable Register							
ADPRI	107Fh	P07F	AD STEST	AD PRIORITY	AD ESPEN	—	—	—	—	—

11.4.1 Analog Control Register (ADCTL)

The ADCTL register controls the input selection, reference voltage selection, sample start, and conversion start.

Analog Control Register (ADCTL)
[Memory Address – 1070h]

Bit #	7	6	5	4	3	2	1	0
P070	CONVERT START	SAMPLE START	REF VOLT SELECT2	REF VOLT SELECT1	REF VOLT SELECT0	AD INPUT SELECT2	AD INPUT SELECT1	AD INPUT SELECT0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R = Read, W = Write, –n = Value of the bit after the register is reset

Bits 0–2 **AD INPUT SELECT0–2**. Analog Input Channel Select Bits 0–2.

These bits select the channel used for conversion. Channels should be changed only after the A/D has cleared the SAMPLE START and CONVERT START bits. Changing the channel while either the SAMPLE START bit or the CONVERT START bit is 1 invalidates the conversion in progress.

AD INPUT SELECT2	AD INPUT SELECT1	AD INPUT SELECT0	Channel
0	0	0	AN0
0	0	1	AN1
0	1	0	AN2
0	1	1	AN3
1	0	0	AN4
1	0	1	AN5
1	1	0	AN6
1	1	1	AN7

Bits 3–5 **REF VOLT SELECT0–2**. Reference Voltage ($+V_{REF}$) Select Bits 0–2.

These bits select the channel the A/D uses for the positive voltage reference. The REF VOLT SELECT bits must not change during the entire conversion.

REF VOLT SELECT2	REF VOLT SELECT1	REF VOLT SELECT0	$+V_{REF}$ Source
0	0	0	V_{CC3}†
0	0	1	AN1
0	1	0	AN2
0	1	1	AN3
1	0	0	AN4
1	0	1	AN5
1	1	0	AN6
1	1	1	AN7

† Pin AN0 cannot be selected as positive voltage reference.

Bit 6 **SAMPLE START**. Sample Start.

Setting this bit stops any ongoing conversion and starts sampling the selected input channel to begin a new conversion. This bit is cleared by the A/D module 18 system-clock cycles after the program sets the CONVERT START bit. Entering halt or standby mode clears this bit and aborts any sampling in progress.

Bit 7 **CONVERT START.** Conversion Start.

Setting this bit starts the conversion. This bit is cleared by the A/D 18 system clock cycles after the program sets the CONVERT START bit. Entering halt or standby mode clears this bit and aborts any conversion in progress.

11.4.2 Analog Status and Interrupt Register (ADSTAT)

The ADSTAT register indicates the converter and interrupt status.

Analog Status and Interrupt Register (ADSTAT)
[Memory Address – 1071h]

Bit #	7	6	5	4	3	2	1	0
P071	—	—	—	—	—	AD READY	AD INT FLAG	AD INT ENA
						R–0	RC–0	RW–0

R = Read, W = Write, C = Clear only, –*n* = Value of the bit after the register is reset

Bit 0 **AD INT ENA**. A/D Interrupt Enable.

This bit controls the A/D module's ability to generate an interrupt.

0 = Disables A/D interrupt.
1 = Enables A/D interrupt.

Bit 1 **AD INT FLAG**. A/D Interrupt Flag.

The A/D module sets this bit at the end of an A/D conversion. If this bit is set while the AD INT ENA bit is set, an interrupt request is generated. Clearing this flag clears pending A/D interrupt requests. This bit is cleared by the system reset. Software cannot set this bit.

Bit 2 **AD READY**. A/D Converter Ready.

The A/D module sets this bit whenever a conversion is not in progress and the A/D is ready for a new conversion to start. Writing to this bit has no effect on its state.

0 = Conversion in process.
1 = Converter ready.

Bits 3–7 **Reserved.** Read data is indeterminate.

11.4.3 Analog Conversion Data Register (ADDATA)

11

The ADDATA register contains the digital result of the last A/D conversion.

Analog Conversion Data Register (ADDATA)
[Memory Address – 1072h]

Bit #	7	6	5	4	3	2	1	0
P072	DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0
	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0

R = Read, –*n* = Value of the bit after the register is reset

The analog-to-digital conversion data is loaded into this register at the end of a conversion and remains until replaced by another conversion.

11.4.4 Analog Port E Data Input Register (ADIN)

The ADIN register contains digital input data when one or more of the AN0 through AN7 pins are used as digital ports.

Analog Port E Data Input Register (ADIN)
[Memory Address – 107Dh]

Bit #	7	6	5	4	3	2	1	0
P07D	PORT E DATA AN 7	PORT E DATA AN 6	PORT E DATA AN 5	PORT E DATA AN 4	PORT E DATA AN 3	PORT E DATA AN 2	PORT E DATA AN 1	PORT E DATA AN 0
	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0

R = Read, –*n* = Value of the bit after the register is reset

The ADIN register shows the data present at the AN0–AN7 pins when they are configured for general-purpose input instead of for A/D channels. A bit is configured as a general-purpose input if the corresponding bit of the port enable register is a 1. Pins configured as A/D channels are read as 0s. Writing to this address has no effect.

11.4.5 Analog Port E Input Enable Register (ADENA)

The ADENA register controls the function of the AN0 through AN7 pins.

Analog Port E Data Input Enable Register (ADENA)
[Memory Address – 107Eh]

Bit #	7	6	5	4	3	2	1	0
P07E	PORT E INPUT ENA 7	PORT E INPUT ENA 6	PORT E INPUT ENA 5	PORT E INPUT ENA 4	PORT E INPUT ENA 3	PORT E INPUT ENA 2	PORT E INPUT ENA 1	PORT E INPUT ENA 0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R = Read, W = Write, –*n* = Value of the bit after the register is reset

The ADENA register individually configures the AN0–AN7 pins as either analog input channels or as general-purpose input pins.

0 = The pin becomes an analog input channel for the A/D converter. When the bit is 0, the corresponding bit in the ADIN register reads as a 0.

1 = Enables the pin as a general-purpose input pin; its digital value can be read from the corresponding bit in the port E data input register.

11.4.6 Analog Interrupt Priority Register (ADPRI)

The ADPRI register selects the interrupt priority level of the A/D interrupt.

Analog Interrupt Priority Register (ADPRI)
[Memory Address – 107Fh]

Bit #	7	6	5	4	3	2	1	0
P07F	AD STEST	AD PRIORITY	AD ESPEN	—	—	—	—	—
	RP–0	RP–0	RP–0					

R = Read, P = Privilege write only, *–n* = Value of the bit after the register is reset

Bits 0–4 **Reserved.** Read data is indeterminate.

Bit 5 **AD ESPEN**. Emulator Suspend Enable.

Normally, this bit has no effect. However, when you are using the XDS emulator to debug a program, this bit determines what happens to the A/D when the program is suspended by an action such as a hardware or software breakpoint.

0 = When the emulator is suspended, the A/D continues to work until the current conversion is complete.
1 = When the emulator is suspended, the A/D is frozen so that its state can be examined at the point that the emulator was suspended. The conversion data is indeterminate upon restart.

Bit 6 **AD PRIORITY**. A/D Interrupt Priority Select.

This bit selects the priority level of the A/D interrupt.

0 = A/D interrupt is a higher priority (level 1) request.
1 = A/D interrupt is a lower priority (level 2) request.

Bit 7 **AD STEST**. This bit must be cleared (0) to ensure proper operation.

Chapter 12
Programmable Acquisition and Control Timer (PACT)

This chapter discusses the architecture and programming of the programmable acquisition and control timer (PACT) module. Even if you have extensive experience with microcontroller timers, you should read this chapter to fully understand how to use the TMS370 PACT module. The chapter covers the following topics:

Topic **Page**

12.1 PACT Overview

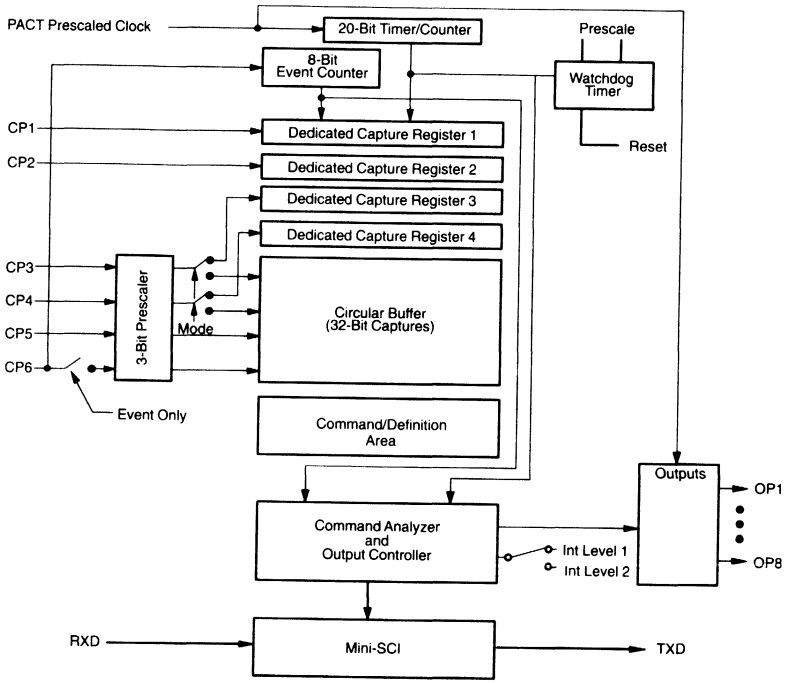
The PACT module acts as a timer coprocessor by gathering timing information on input signals and controlling output signals with little or no intervention by the CPU. The coprocessor nature of this module allows for levels of flexibility and power not found in traditional microcontroller timers.

12.1.1 Physical Description

The PACT module, shown in Figure 12–1, consists of:

- ❑ Input capture functions on up to six input pins, four of which (CP3–CP6) may have a programmable prescaler
- ❑ Timer-driven outputs on eight pins
- ❑ Configurable timer overflow rates for different functions
- ❑ One 8-bit event counter driven by CP6
- ❑ Timer capability of up to 20 bits
- ❑ Interaction between event counter and timer activity
- ❑ Register-based organization, allowing single-cycle accesses to parameters
- ❑ Eighteen independent interrupt vectors with two priority levels
- ❑ Integrated, configurable watchdog timer with selectable time-out period
- ❑ Mini-serial communications interface with independent set-up of bit rate (baud) for receive and transmit lines

Figure 12–1. PACT Block Diagram



12.1.2 Control Registers

The PACT control registers are located at addresses 1040h to 104Fh and occupy peripheral file frame 4. The function of each is shown in Table 12–1.

Table 12–1. PACT Peripheral Frame

Peripheral File Location	Symbol	Name	Description
P040	PACT SCR	Setup Control Register	Determines the time base for the PACT module, enables the command/definition area, and controls the default timer overflow.
P041	CDSTART	Command/Definition Area Start Register	Defines the starting address of the command/definition area and enables the interrupts for that area.
P042	CDEND	Command/Definition Area End Register	Defines the end address of the command/definition area.
P043	BUFPTR	Buffer Pointer Register	Defines the address of the buffer pointer.
P044		Reserved	
P045	SCICTLP	PACT-SCI Control Register	Controls the functions of the mini-SCI.
P046	RXBUFP	PACT-SCI RX Data Register	Contains the data received by the SCI.
P047	TXBUFP	PACT-SCI TX Data Register	Contains the data to be transmitted by the SCI.
P048	OPSTATE	Output Pins 1–8 State Register	Contains information about the current state of the output pins.
P049	CDFLAGS	Command/Definition Entry Flags Register	Contains information about the command/definition interrupts.
P04A	CPCTL1	Setup CP Control Register 1	Controls the functions of the CP1 and CP2 pins.
P04B	CPCTL2	Setup CP Control Register 2	Controls the functions of the CP3 and CP4 pins.
P04C	CPCTL3	Setup CP Control Register 3	Controls the functions of the CP5 and CP6 pins.
P04D	CPPRE	CP Input Control Register	Controls input and output functions.
P04E	WDRST	Watchdog Reset Key	Location that is written to when serving the watchdog.
P04F	PACTPRI	Global Function Control Register	Controls the watchdog time-out rate, the PACT interrupt priority levels, and the PACT operating mode.

The PACT module is controlled not only by the peripheral file but also by the defined areas of the dual-port RAM. The dual-port RAM is located at addresses 0080h and 00FFh on the memory map (see Section 12.3 for more information).

12.2 PACT Operation

The following subsections describe the functions and options of the PACT module.

12.2.1 Hardware Pins

The PACT module has 16 external hardware pins allocated to its functions. There are three groups of pins:

- **Input Pins.** The input or capture pins are CP1 to CP6. The function of these pins depends on the mode selected:

Mode A		Mode B	
CP1–2	Dedicated capture	CP1–4	Dedicated capture
CP3–6	Circular buffer capture	CP5–6	Circular buffer capture
CP6	Event pin	CP6	Event pin

- **Output pins.** There are eight output pins: OP1–OP8.

- **SCI receive/transmit pins.** The PACT module has two pins for the mini-SCI: RX (receive data) and TX (transmit data).

In the TMS370Cx3x devices, the input pins CP3, CP4, and CP5 are internally bonded with I/O pins D4, D6, and D7, giving you a software-governed choice of function. Output pins (OP1–8) are initialized to a logic low on reset.

12.2.2 Memory Organization

To use the PACT module, you must set up three distinct areas of memory:

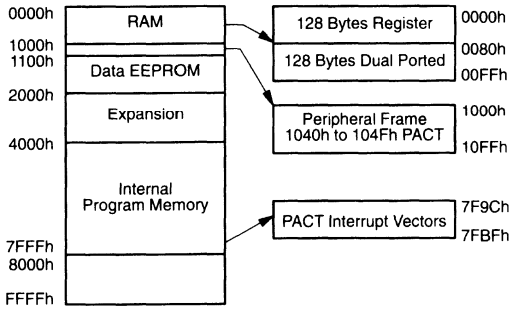
- **128 bytes of dual-port RAM** contain the capture registers, the circular buffer, and a command/definition area. Dual-port RAM is described in Section 12.3.

- **Peripheral file frame 4** contains the hardware registers used for initial set-up. These registers are described in Section 12.11.

- Three groups of **interrupt vectors** are available. To use interrupts, you must set up the interrupt vectors. Interrupts are described in Section 12.7.

The memory map in Figure 12–2 is a typical implementation of the PACT module.

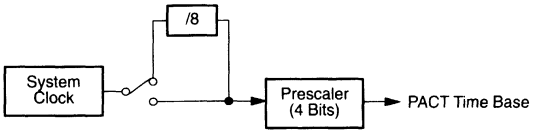
Figure 12–2. TMS370 Memory Map Highlighting PACT Areas



12.2.3 Time Base

The time base section of PACT is very similar to that used in traditional timers. The microcontroller system clock is routed to a prescaler that feeds a hardware counter. The prescale section consists of a 4-bit prescaler and an optional divide-by-8 circuit, as shown in Figure 12–3. The hardware counter is 20 bits wide.

Figure 12–3. Prescaler Circuit



The divide rate is the binary value of the 4-bit prescaler plus one except for the value zero which, by hardware, provides a divide rate of two. The five bits that control the prescaler are located in the PACTSCR register at address 1040h. Refer to subsection 12.11.1 for more information.

Divide Rate			Divide Rate		
PACT PRESCALE SELECT	FAST MODE SELECT		PACT PRESCALE SELECT	FAST MODE SELECT	
3 2 1 0	1	0	3 2 1 0	1	0
0 0 0 0	2	16	1 0 0 0	9	72
0 0 0 1	2	16	1 0 0 1	10	80
0 0 1 0	3	24	1 0 1 0	11	88
0 0 1 1	4	32	1 0 1 1	12	96
0 1 0 0	5	40	1 1 0 0	13	104
0 1 0 1	6	48	1 1 0 1	14	112
0 1 1 0	7	56	1 1 1 0	15	120
0 1 1 1	8	64	1 1 1 1	16	128

12.2.4 Command/Definition File Format

All entries in the command/definition area are 32 bits in length. Normally, commands are executed sequentially starting with entry 0, the entry next to the circular buffer. Each entry requires a specified number of time slots (see the command/definition descriptions in Section 12.6 for the number of time slots required for each command).

12.2.5 Available Time Slots

The number of time slots available to you, and hence the number of definitions/commands allowed, is governed by the crystal frequency and the resolution required by PACT functions. As a result, you must balance these factors to suit your application.

The PACT prescaler value determines the number of slots available. The prescaler gives the ratioed crystal frequency against resolution achievable by the PACT. Table 12–2 defines the maximum number of time slots available for all prescaler options.

Table 12–2. Number of Time Slots Available for Each Prescale Setting

Divide Rate	Time Slots	Divide Rate	Time Slots	Divide Rate	Time Slots
—	—	11	31	56	179
2	2	12	35	64	205
3	5	13	38	72	231
4	9	14	41	80	257
5	12	15	45	88	283
6	15	16	48	96	309
7	19	24	74	104	336
8	21	32	101	112	362
9	25	40	127	120	389
10	29	48	153	128	415

If your application requires a prescaler value that does not provide sufficient time slots, you can use the STEP command to cut in half the resolution of all commands later in the command/definition area. The STEP command affects the second entry after the one containing the command.

For example, if 10 entries exist in the command file and entry 2 contains a STEP instruction, then commands 0–3 run at full resolution, but 4–9 run at half resolution.

Note that this use of the STEP command affects *all* operations, including the clocking of virtual and offset timers. Repeated use of the STEP instruction within a command file is not necessary—the commands are run at either full or half speed (no slower).

Note:

When you use STEP, the end address of the command/definition area must be programmed as the next to last address that will be executed.

12.3 Dual-Port RAM

The PACT is a RAM-based module that occupies an area of the internal RAM. The size of the RAM is determined by the functions that you select, according to the end device.

It does not matter to the PACT module where the dual-port RAM is located in the memory map. By convention, it is located in the register file from 0080h to 00FFh. This gives the CPU maximum speed in accessing the registers in the register file when you use the register address mode.

The dual-port RAM contains the following major areas:

- **Command/definition area.** The length of the command/definition area (described in Section 12.6) is defined in the software. Four bits define the start, and five bits define the end of the command area (the two least significant bits (LSBs) are not defined, because this area must start and finish on a 32-bit boundary). The start address of the command area also serves to define the length of the circular buffer because the buffer resides between the last dedicated storage location and the start of the command area.

  All RAM at lower addresses than the end of the command area can be used for general purposes such as registers, stack, etc. However, since all of the RAM locations above the end are also within the register file, the data in these locations can be directly manipulated with normal register-based instructions.

- **Circular buffer.** The circular buffer is an area in memory that stores the value of a PACT timer when a capture request is made. As new values are captured, they are put into successive locations in the buffer. When the buffer is full, the oldest captures are replaced with newer captures.

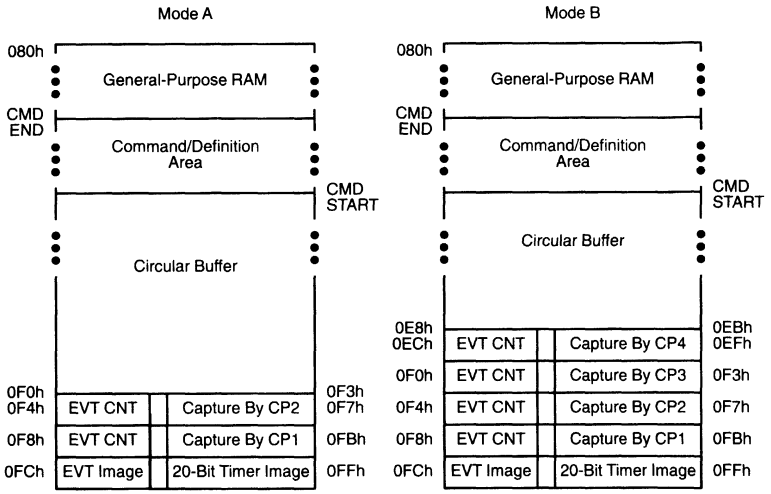
  The length of the circular buffer is defined in the software. The circular buffer is directly before the capture registers.

- **Capture registers.** One of the major differences between the PACT module and standard timers is the location of the 32-bit capture registers in dual-port RAM. These locations can be read or written by the CPU. They are automatically written to by PACT when the appropriate feature is enabled. The capture registers are directly before the timer and event counter images. For more information about the input capture pins, refer to Section 12.4.

The addresses 0FCh–0FFh of the dual-port memory contain an image of the 20-bit default timer and an image of the 8-bit event counter. Since they are images or copies of the actual hardware registers, they can be overwritten by the application software. However, they will be rewritten every time the PACT module receives another prescaled clock.

The dual-port RAM is 128 bytes long as shown in Figure 12–4.

Figure 12–4. Dual-Port RAM Organization

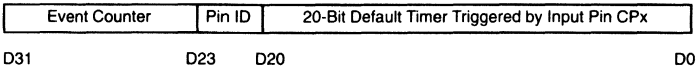


Mode A provides two, and Mode B provides four dedicated 32-bit storage locations, which follow a circular storage buffer. Modes A and B are described in Section 12.4.

The PACT module uses memory starting from the highest address going to lower addresses. For the TMS370Cx3x devices, the highest address of the PACT module's dual-port RAM is 00FFh. An image of the 20-bit default timer, a copy of the flag bits for capture pins 3 to 6, and an image of the 8-bit event counter are always held in the first 32-bit block. Thereafter, allocation depends on the mode selected.

12.4 Inputs

The PACT module has six input capture pins that cause data to be stored into fixed locations. Each location is defined by the pin that triggers the capture. When triggered directly from the pin, capture values are 32 bits long and consist of the 20-bit hardware timer, the 8-bit event counter, and four extra bits that identify the pin that caused the capture in the circular buffer:



The pin ID is set according to which input caused the capture. Only one of these bits will be set:

Bit	Transition on Pin	Result
D20 = 1	CP3	Has caused the capture
D21 = 1	CP4	Has caused the capture
D22 = 1	CP5	Has caused the capture
D23 = 1	CP6	Has caused the capture

Each input capture pin has a rising edge select bit, a falling edge select bit, and an interrupt enable bit, along with a fourth bit that acts as a flag to cause an interrupt. Table 12–3 lists these bits for each of the pins; the bits are described in Section 12.11.

Table 12–3. Bits That Control Functions on the Input Capture Pins

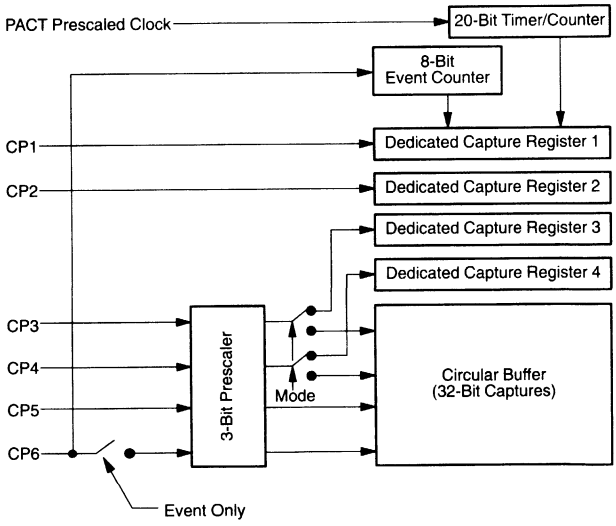
Pin	Rising Edge Select Bit	Falling Edge Select Bit	Interrupt Enable Bit	Interrupt Flag Bit
CP1	CP1 CAPT RISING EDGE (CPCTL1.1)	CP1 CAPT FALLING EDGE (CPCTL1.0)	CP1 INT ENA (CPCTL1.3)	CP1 INT FLAG (CPCTL1.2)
CP2	CP2 CAPT RISING EDGE (CPCTL1.5)	CP2 CAPT FALLING EDGE (CPCTL1.4)	CP2 INT ENA (CPCTL1.7)	CP2 INT FLAG (CPCTL1.6)
CP3	CP3 CAPT RISING EDGE (CPCTL2.1)	CP3 CAPT FALLING EDGE (CPCTL2.0)	CP3 INT ENA (CPCTL2.3)	CP3 INT FLAG (CPCTL2.2)
CP4	CP4 CAPT RISING EDGE (CPCTL2.5)	CP4 CAPT FALLING EDGE (CPCTL2.4)	CP4 INT ENA (CPCTL2.7)	CP4 INT FLAG (CPCTL2.6)
CP5	CP5 CAPT RISING EDGE (CPCTL3.1)	CP5 CAPT FALLING EDGE (CPCTL3.0)	CP5 INT ENA (CPCTL3.3)	CP5 INT FLAG (CPCTL3.2)
CP6	CP6 CAPT RISING EDGE (CPCTL3.5)	CP6 CAPT FALLING EDGE (CPCTL3.4)	CP6 INT ENA (CPCTL3.7)	CP6 INT FLAG (CPCTL3.6)

When you select the rising or falling edge or both, the capture function for that pin is enabled, and the timer value captured will be stored in the location that is determined by the mode of operation.

Two operating modes are available for the PACT module: mode A and mode B. You can select between these modes according to the capture functions that you need (refer to Figure 12–5):

- **Mode A** offers two dedicated capture locations (associated with pins CP1 and CP2) plus four other pins (CP3–6), each with a programmable prescaler to store 32-bit data in the circular buffer. The prescaler rate is the same for all of the four pins (CP3–6). Pin CP6 also clocks the 8-bit event counter.
- **Mode B** offers four dedicated capture locations (associated with pins CP1–4). Pins CP3–6 have a programmable prescaler. Pin CP5 can capture 32-bit data in the circular buffer when the software defined edge occurs. The remaining capture pin, CP6, clocks the 8-bit event counter and can capture 32-bit data in the circular buffer.

Figure 12–5. Input Capture Block Diagram



Captures can be set to occur on the falling, rising, or both edges of the input signal.

Capture pins CP3 through CP6 can be prescaled with a divide value from 1 to 8. Each of these four pins has its own edge counter, but the maximum count value (1–8) before an actual capture occurs must be the same for all four pins.

Since it takes several system clock periods for the CPU to read a 20-bit timer capture value, it is possible for an additional capture to occur while the original capture is being read. Your program can detect this situation by clearing the capture flag in the peripheral file before the read and then verifying that the flag has not been set again after the read is complete. If the flag was set again, the value read can be invalid and should be reread.

The buffer pointer (BUFPTR register) is available to tell the program when the last capture value was stored. You can also modify the buffer pointer under processor control.

Note:

The 16-bit captures in the circular buffer area are available when triggered by commands in the command/definition area. A 32-bit capture can overwrite the last 16-bit capture if the 16-bit capture is located at the two higher addresses (address bit0 = 1 and address bit1 = 1) of a 32-bit block.

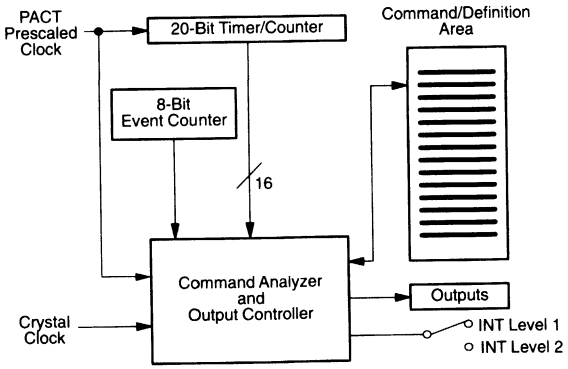
12.5 Control and Outputs

The control and outputs section of PACT is perhaps the most unique and most powerful part of this timer. Figure 12–6 shows the output control section of the PACT block diagram.

- The controller acts like a state machine and starts when it receives a rising edge from the PACT prescaled clock. The controller reads its commands (or state microcode) from the command definition part of the dual-port RAM.
- The 8-bit event counter and the 16 LSBs of the 20-bit default counter are also input into the controller for use in comparisons.
- The outputs from the controller set or clear the 8 output pins (OP1–8).
- The prescaled clock from the PACT time base is used only to start the controller.

The controller steps through its commands, using the system clock phases for synchronization. The controller must step through all of the commands in the command/definition area before the next rising edge of the prescaled clock. The next prescaled clock increments the 20-bit default counter and starts the whole process over.

Figure 12–6. Output Control Section



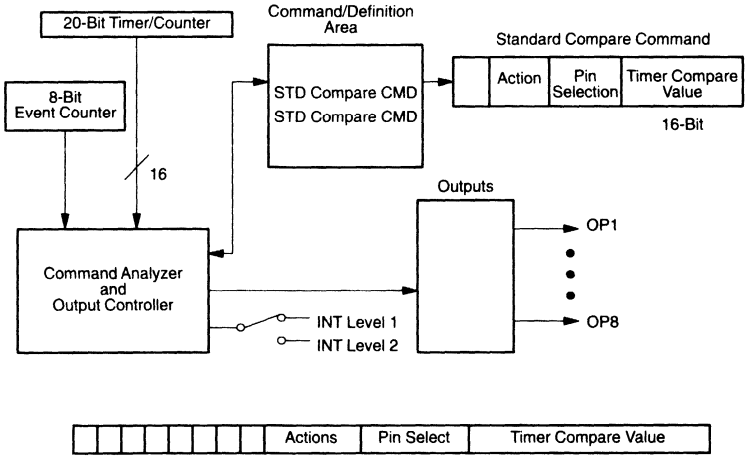
12.5.1 Standard Compare Command

To use the controller, you should understand the commands that it can execute. All of the commands or definitions in the command/definition area are 32 bits long. The simplest command is the Standard Compare command. The Standard Compare command sets or clears an output pin whenever the timer/counter is equal to a certain value. As shown in Figure 12–7, the Standard Compare command is made up of:

- A 16-bit compare value
- 3 bits to select one of the 8 output pins
- Bits to select what action to take
- Bits to distinguish this command from the others

For more information or actual bit definitions, refer to subsection 12.6.4.

Figure 12–7. Standard Compare Command



The Standard Compare command can:

- Set or clear the chosen output pin when the counter matches the compare value,
- Execute the opposite action (clear or set) when the 16 LSBs of the counter are equal to zero, or
- Generate an interrupt when the compare value is reached.

Therefore, with a Standard Compare command, you can make a pulse-width modulated (PWM) output of limited usefulness. Assume that you want a PWM output with an initial duty cycle of 75%. Using Standard Compare command:

- ❑ Set the timer compare value to 04000h (1/4 the overflow rate)
- ❑ Set the actions to cause an output pin to go high when the count is equal to the compare value and then low again when the 16 LSBs of the counter are zero.
- ❑ Vary the duty cycle by changing the 16-bit compare value.
- ❑ Invert the signal by selecting clear on compare equal, as opposed to set on compare equal.

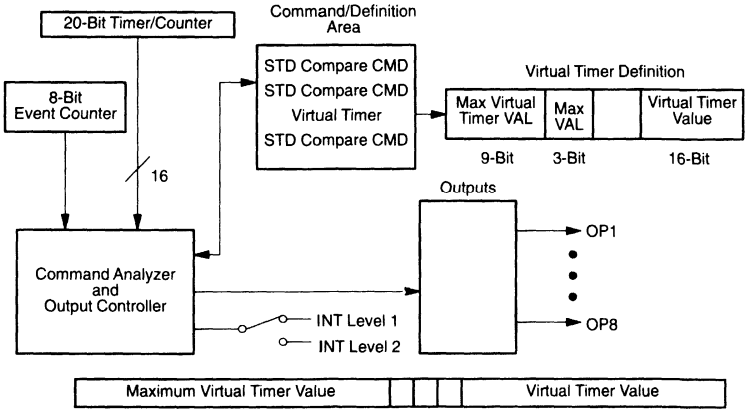
You cannot use this command to vary the period of the PWM.

12.5.2 Virtual Timers

You can vary the period of the PWM by using a virtual timer. Remember that the command/definition area is implemented in RAM. Figure 12–8 shows the virtual timer definition and its implementation. The virtual timer definition consists of

- ❑ 16 bits that are read, incremented, and rewritten on each tic of the PACT clock.
- ❑ 13 bits that define a maximum value. When the virtual timer reaches this maximum value, it is reset to zero.

Figure 12–8. Virtual Timer Implementation



For more information about the actual bit definitions, refer to subsection 12.6.1.

The command/definition area of Figure 12–8 shows two Standard Compare commands, a virtual timer definition, and a third Standard Compare command. Assume that you are using a microcontroller with a 200-ns (5-MHz) internal system clock and that you are prescaling the PACT clock with divide by five so that each PACT clock tic is one microsecond.

- ❑ The first two Standard Compare commands generate PWM signals of variable duty cycle with a period of 65,536 prescaled clock tics (65.536 ms).
- ❑ If you want the third PWM to have a period of one millisecond, then you set up the virtual timer with a maximum value of one thousand.
- ❑ When the controller sees the timer definition, it will increment the virtual timer and then use the virtual timer value for future comparisons.
- ❑ The third Standard Compare command generates a PWM of variable duty cycle with a period of one millisecond.

Combinations of Standard Compare commands and virtual timers can be used to create complex repeating waveforms. Multiple Standard Compare commands can be used on a single output pin to create multiple pulses of different duration.

You can use virtual timers to provide periodic interrupts to the processor.

12.5.3 Double Event Compare Command

Actions can also be taken as determined by comparisons to the 8-bit event counter. Since all commands are 32 bits wide, the Double Event Compare command actually defines two event compare values and the actions that can be performed based on each value. The actions that are allowed according to the event 1 compare value matching the event counter are:

- ❑ Set or reset the selected output pin (OP1–8),
- ❑ Interrupt, or
- ❑ Generate a 32-bit capture into the circular buffer.

The actions that are allowed according to the event 2 compare value matching the event counter are:

- ❑ Set or reset the selected output pin (OP1–8),
- ❑ Interrupt,
- ❑ Generate a 32-bit capture into the circular buffer, or
- ❑ Reset the 20-bit default timer.

Because of synchronization, these actions will occur two or three prescaled clock cycles after the input edge that incremented the event counter. A block diagram of the double event command is shown below. This diagram shows the information contained in the command. For more information or actual bit definitions, refer to subsection 12.6.5.

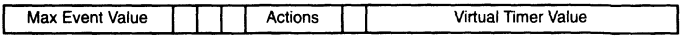
So far, you can manipulate output lines depending on time values or the number of external events. An additional virtual timer definition allows you to manipulate output lines according to a combination of the event counter and time.

12.5.4 Offset Timer Definition-Time From the Last Event

The offset timer definition-time from the last event creates a 16-bit virtual timer that is cleared on each occurrence of an event on pin CP6. This definition also sets an event counter maximum, so that the event counter is reset after reaching this maximum value. The offset timer definition can perform the following actions:

- ❑ Generate an interrupt when the maximum event count is reached,
- ❑ Store the 16-bit virtual timer in the circular buffer on each event,
- ❑ Store the 20-bit default timer and 8-bit event counter in the circular buffer when the maximum event count is reached, or
- ❑ Reset the 20-bit hardware default timer when the maximum event count is reached.

A block diagram of the offset timer definition is shown below. This diagram shows the information contained in the command. For more information or actual bit definitions, refer to subsection 12.6.2.



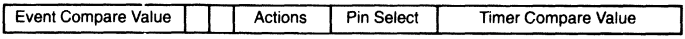
12.5.5 Conditional Compare Command

There is a special compare command that has a timer compare value and an event compare value. Both of these values must match for the defined action to take place. Usually, a series of these commands follows an offset-timer definition time from last the event and provides output pulses on different pins, based on the event count and an elapsed time from the event. The actions that can be generated by the Conditional Compare command are:

- Generate an interrupt when both conditions are met:
  - The event compare value equals the event counter.
  - The timer compare value equals the last defined timer.
- Set or clear one of seven output pins (OP1– OP7) when the following two conditions are met:
  - The event compare value equals the event counter.
  - The timer compare value equals the last defined timer.

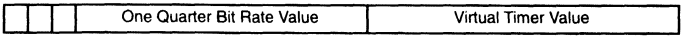
The same actions described above can be enabled on the event counter reaching the event compare value plus one, without regard to the timer compare value. This allows for the case where the next event occurs before the delay period specified by the timer compare value was reached.

A block diagram of the conditional compare command is shown below. This diagram shows the information contained in the command. For more information or actual bit definitions, refer to subsection 12.6.6.



12.5.6 Baud Rate Timer Definition

The last item that can be put into the command/definition area of the PACT module is a baud rate virtual timer. This virtual timer runs the serial communications port built into the PACT module. Set up the maximum timer value for one-quarter bit period of the desired bit rate. Separate timers can be defined for transmit and receive. For more information on the baud rate timer definition, see subsection 12.6.3. For more information about the SCI, see subsection 12.9.



12.6 Command/Definition Area

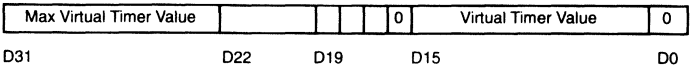
All commands/definitions are 32 bits long. They are stored in memory most significant byte (MSbyte) first. If byte 3 is stored at location N, then byte 0 would be at location N+3. The bits are referenced as D0–D31.

This section summarizes the available commands and the number of time slots required for each command.

Definitions	**Time Slots**
Virtual timer definition	2
Offset timer definition	2/3
Baud rate timer definition	2

Commands	**Time Slots**
Standard Compare command	1
Conditional Compare command	1
Double Event Compare command	1

12.6.1 Virtual Timer Definition



Requires two time slots

D0 This bit must be written as a 0 for this definition to be valid.

D1–D15 Virtual Timer Value

These are the 15 MSBs of a 16-bit virtual timer. This timer is resident at this location, so any write to this address by the CPU modifies the timer value. Because of hardware limitations, the LSB of the virtual timer cannot be read from or written to by the CPU, but it is used by PACT commands such as the Standard Compare command.

D16 This bit must be written as a 0 for this command to be valid.

D17 Interrupt on 0—Active = 1

Sets interrupt flag when the virtual timer (D1–15) overflows to zero.

D18 Enable—Active = 1

Enables the timer update. Used to stop or start the timer.

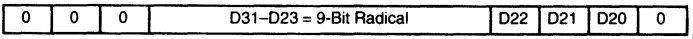
D19 Range Bit

Used in conjunction with D20–22 to define the maximum value (see the illustration below).

D20–22 Define a further three bits of the maximum value of the virtual timer. Either the MSBs (bits 13–15) of the maximum value if range bit=1, or the LSBs (bits 1–3) if range bit=0. The undefined bits of the maximum value for the virtual timer are set to 0.

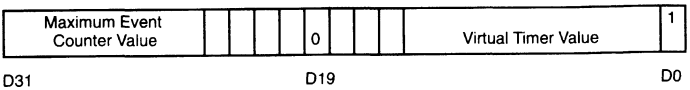
D23–31 Sets the radical of the maximum value of the virtual timer. Used with D20–22 to specify the maximum value of the virtual timer. The maximum virtual timer value is equal to the desired period minus 2. For example, if you want a timer with a period of 100 PACT prescaled clocks, set the maximum virtual timer value to 98. The virtual timer increments from 0 to 99 and is then reset to 0.

Maximum Value Format (D19=0)



Maximum Value Format (D19=1)

12.6.2 Offset Timer Definition-Time From Last Event



Requires two time slots if bit D21 = 0; requires three time slots if bit D21 = 1.

D0 This bit must be written as a 1 for this definition to be valid.

D1–D15 Virtual Timer Value

These are the 15 MSBs of a 16-bit virtual timer. This timer is resident at this location, so any write to this address by the CPU modifies the timer value. Because of hardware limitations, the LSB of the virtual timer cannot be read from or written to by the CPU, but it is used by the PACT commands such as the Standard Compare command.

D16 Step—Active = 1

Allows lower resolution on the following commands in this definition area (see subsection 12.2.5 for details on the use of this function).

D17 Interrupt on Maximum Event—Active = 1

Sets the interrupt flag when the event counter reaches the maximum value (D24–31)

D18 Enable—Active = 1

Enables the timer update. Used to stop and start the timer.

D19 This bit must be written as a 0 for this definition to be valid.

D20 Reset Default Timer—Active = 1

Clears the default timer when the event counter reaches the maximum value (D24–31).

D21 Virtual Capture—Active = 1

Stores the 16-bit virtual offset timer (defined by this definition) in the circular buffer on every event on CP6 before it is cleared.

D22 Default Capture—Active = 1

Captures 32-bit data into the circular buffer when the event counter reaches the maximum value (D24–31).

D23 Interrupt on Event—Active = 1

Sets the interrupt flag when an event on pin CP6 occurs.

D24–31 Event Maximum Value

Specifies a maximum for the event counter. Upon reaching this value, the event counter will be reset to zero by the next event on CP6.

12.6.3 Baud Rate Timer Definition

Max Virtual Timer Value					1	Virtual Timer Value	0
D31	D22	D19				D15	D0

Requires two time slots

D0 This bit must be written as a 0 for this definition to be valid.

D1–D15 Baud Rate Timer

These are the 15 MSBs of a 16-bit virtual timer used as the baud-rate generator. The timer is resident at this location, so any write to this address modifies its value.

D16 This bit must be written as a 1 for this definition to be valid.

D17 TXselect—Active = 1

Selects this timer definition to be used for the transmit baud-rate generator.

D18 RXselect—Active = 1

Selects this timer definition to be used for the receive baud-rate generator.

D19 Range Bit

Used in conjunction with D20–22 (see the illustration below).

D20–22 Define a further three bits of the maximum value of the virtual timer. Either the MSBs (bits 13–15) of the maximum value if range bit=1, or the LSBs (bits 1–3) if range bit=0. The undefined bits of the maximum value for the virtual timer are set to 0.

D23–31 Set the radical of the maximum value of the virtual timer. Used with D20–22 to specify the maximum value of the virtual timer. When the virtual timer reaches the defined value, the next prescaler clock cycle will cause the timer to be cleared. The maximum virtual timer value should be set to one quarter of the desired bit time minus 2. For more details, see Section 12.9.

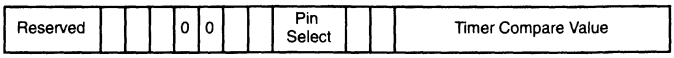
Maximum Value Format (D19=0)

0	0	0	D31–D23 = 9-Bit Radical	D22	D21	D20	0

Maximum Value Format (D19=1)

D22	D21	D20	D31–D23 = 9-Bit Radical	0	0	0	0

12.6.4 Standard Compare Command



Requires one time slot

D0–15 Timer Compare Value

These 16 bits provide timer-compare value. The timer to which this value is compared is either the last virtual timer defined above this command or, if no virtual timer has been defined, the default timer.

D16 Next Command Is a Definition—Active = 1

Indicates that the following entry in the command/definition area will be a definition.

D17 Interrupt on Compare—Active = 1

Sets the interrupt flag when the compare value is matched by the referred timer.

D18–20 Pin Selection

Select an output pin whose state will be modified when the compare value is matched. The pin number is the binary value of D20–18 (20=MSB,18=LSB) plus one.

D21 Compare Action—Set = 1, Clear = 0

Sets or resets the pin defined by D18–20 when the compare value is matched by the referred timer.

D22 Step—Active = 1

Allows lower resolution on following commands in this definition area (see subsection 12.2.5 for details on use of this function).

D23–24 These bits must be written as a 0 for this definition to be valid.

D25 Reset Action

Sets or resets the pin defined by D18–20 when the referred timer is reset to zero.

0 = No action when the referred timer is zero.
1 = When the referred timer is zero, execute the opposite action.

D26 Interrupt on Reset—Active = 1

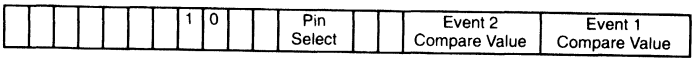
Sets the interrupt flag when the referred timer is reset to zero.

D27 Enable Pin—Active = 1

Enables output pin actions on this command.

D28–31 Reserved.

12.6.5 Double Event Compare Command



Requires one time slot.

D0–7 Event 1

Set an 8-bit value that, when matched by the 8-bit event counter, causes the action defined by D17, D21, and D29.

D8–15 Event 2

Set an 8-bit value that, when matched by the 8-bit event counter, causes the action defined by D25, 26, 28, and 30.

D16 Next Command Is a Definition—Active = 1

Indicates that the following entry in the command/definition area will be a definition.

D17 Interrupt on Compare 1—Active = 1

Sets the interrupt flag when the event 1 compare value is matched by the event counter.

D18–20 Pin Selection

Select an output pin whose state will be modified when the compare value is matched. The pin number is the binary value of D20–18 (20=MSB,18=LSB) + 1 (OP1–8).

D21 Compare Action 1—Set = 1, Clear = 0

Sets or resets the pin defined by pin selection when the event 1 compare value (D0–D7) is matched by the event counter.

D22 Step—Active = 1

Allows lower resolution on the following commands (see subsection 12.2.5 for details on use of this function).

D23 This bit must be written as a 0 for this command to be valid.

D24 This bit must be written as a 1 for this command to be valid.

D25 Compare Action 2—No Action = 0 Inverted Action = 1

Sets or resets the pin defined by pin selection when the event 2 compare value (D8–D15) is matched by the event counter.

D26 Interrupt on Compare 2—Active = 1

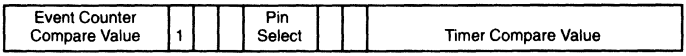
Sets the interrupt flag when event 2 occurs.

D27 Enable pin—Active = 1

Enables the output pin actions for this command.

D28	Event 2 Default Timer Reset—Active = 1 Resets the default timer when event 2 occurs.
D29	Event 1 Default Timer Capture—Active = 1 Stores a 32-bit data capture in the circular buffer when event 1 occurs.
D30	Event 2 Default Timer Capture—Active = 1 Stores a 32-bit data capture in the circular buffer when event 2 occurs.
D31	Reserved.

12.6.6 Conditional Compare Command



Requires one time slot.

D0–15 Timer Compare Value

These 16 bits provide a timer compare value. The timer to which this value is compared is either the last virtual timer defined above this command or, if no virtual timer has been defined, the default timer. This is called the referred timer. The value that you write *must* be greater than one.

D16 Next Command is a Definition Active = 1

Indicates that the following entry in the command/definition area will be a definition.

D17 Interrupt on Compare—Active = 1

Sets the interrupt flag when the timer compare value (D0–15) is matched by the referred timer, *and* the event compare value (D24–31) is matched by the event counter.

D18–20 Pin Selection

Select an output pin whose state will be modified when the compare value is matched. The pin number is the binary value of D20–18 (20=MSB,18=LSB) plus one, except the binary value 111, which disables any pin action. Therefore, OP8 is not available for this command.

D21 Compare Action—Set = 1, Clear = 0

Sets or resets the pin defined by pin selection when both compare values are matched by the referred timer and the event counter.

D22 Same Action—Active = 1

Indicates the same action as compare action when the event counter reaches the event compare value plus one. This allows an action on the next event if the next event occurs before the time value is reached. If Same Action = 0, then there will be no action on event compare + 1.

D23 This bit must be written as a 1 for this command to be valid.

D24–31 Event Compare Value

Sets an 8-bit value that is compared with the 8-bit event counter. The actions selected by this command will occur under either of the following conditions:

- The event compare value matches the value of the event counter, *and* the timer compare value matches the referred timer value, or
- The same action active bit is set, *and* the event counter matches the event compare value + 1.

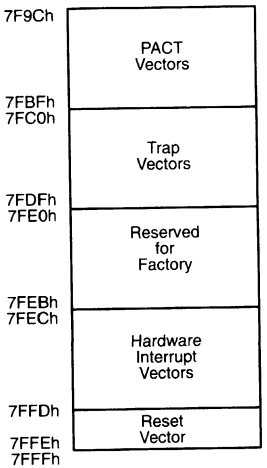
12.7 Interrupts

This section discusses interrupts that are specific to the PACT module. There are three groups of interrupt vectors.

- The first is associated with the events on a particular capture pin.
- The second is associated with the SCI interrupts, such as the receive buffer full.
- The third group of interrupts is associated with the absolute position of the command or definition within the RAM area.

The eighteen vectors available for PACT functions are located immediately after the trap vectors in the TMS370 address space. Refer to the memory map in Figure 12–9.

Figure 12–9. Interrupt Vector Memory Map



As is standard in the TMS370, two levels of priority (1 and 2) exist for each of the three groups of interrupts described above. These interrupt groups can be allocated to one of two interrupt levels:

- A priority that determines the order in which multiple interrupts within a level are serviced (see Table 12–4).
- An order for servicing groups on the same level (see Table 5–2, page 5-3).

Interrupts are enabled either in peripheral frame 4 or within the command/definition line. The service routine must clear the flag associated with the interrupt to prevent multiple servicing of the same interrupt.

Table 12–4. Interrupt Vector Sources

Module	Vector Address	Interrupt Source	Interrupt Flag	System Interrupt	Priority in Group†
PACT (Group 2)	7F9Ch, 7F9Dh	PACT SCI TXINT	PACT TXRDY	PTXINT	2
	7F9Eh, 7F9Fh	PACT SCI RXINT	PACT RXRDY	PRXINT	1
PACT (Group 3)	7FA0h, 7FA1h	PACT Cmd/Def Entry 0	CMD/DEF INT 0 FLAG	CDINT0	1
	7FA2h, 7FA3h	PACT Cmd/Def Entry 1	CMD/DEF INT 1 FLAG	CDINT1	2
	7FA4h, 7FA5h	PACT Cmd/Def Entry 2	CMD/DEF INT 2 FLAG	CDINT2	3
	7FA6h, 7FA7h	PACT Cmd/Def Entry 3	CMD/DEF INT 3 FLAG	CDINT3	4
	7FA8h, 7FA9h	PACT Cmd/Def Entry 4	CMD/DEF INT 4 FLAG	CDINT4	5
	7FAAh, 7FABh	PACT Cmd/Def Entry 5	CMD/DEF INT 5 FLAG	CDINT5	6
	7FACh, 7FADh	PACT Cmd/Def Entry 6	CMD/DEF INT 6 FLAG	CDINT6	7
	7FAEh, 7FAFh	PACT Cmd/Def Entry 7	CMD/DEF INT 7 FLAG	CDINT7	8
PACT (Group 1)	7FB0h, 7FB1h	PACT Circular Buffer (Half/ Full)	BUFFER HALF/FULL INT FLAG	BUFINT	1
	7FB2h, 7FB3h	PACT CP6 Edge	CP6 INT FLAG	CP6INT	2
	7FB4h, 7FB5h	PACT CP5 Edge	CP5 INT FLAG	CP5INT	3
	7FB6h, 7FB7h	PACT CP4 Edge	CP4 INT FLAG	CP4INT	4
	7FB8h, 7FB9h	PACT CP3 Edge	CP3 INT FLAG	CP3INT	5
	7FBAh, 7FBBh	PACT CP2 Edge	CP2 INT FLAG	CP2INT	6
	7FBCh, 7FBDh	PACT CP1 Edge	CP1 INT FLAG	CP1INT	7
	7FBEh, 7FBFh	PACT Default Timer Overflow	DEFTIM OVRFL INT FLAG	POVRFL INT	8

† 1 is the highest priority.

Notes:

1) CP1–6 interrupts are caused by the software edge(s) selected for that particular pin in peripheral frame 4. Also, the associated interrupt enable bit in frame 4 must be set for that particular pin. Interrupts set in a command/definition line relate to their position in that area.

2) The entry address is derived from bits 2, 3, and 4 of the address. If the command or definition is at address 006Ch, 006Dh, 006Eh, or 006Fh (32 bits), then 0110 11xx is the binary value of the address. The entry address comes from bits 4–3–2 = 011 = Entry 3. Thus, the vector associated with entry address 3 will be used when this command or definition causes an interrupt.

For command/definition areas that contain more than eight entries, the entry vectors become overlaid, and the program must determine the correct source. Entry 0 has the same vector as entry 8, so the command/definition at address 04Ch, 04Dh, 04Eh, or 04Fh has the same interrupt vector as the command/definition located at address 06Ch, 06Dh, 06Eh, or 06Fh: both are entry 3.

12.8 Watchdog Timer

At power-up, the watchdog timer is enabled with the shortest time-out period (bit 9 of default timer).

A watchdog-originated reset is generated when a software-selected bit of the default timer toggles. Three options determine the watchdog time-out period and a disable watchdog code. These options are specified according to how bits 0 and 1 of the global function control register (PACTPRI) are configured:

PACT WD PRE-SCALE SELECT 1 (PACTPRI.1)	PACT WD PRE-SCALE SELECT 0 (PACTPRI.0)	Options
0	0	Reset when bit 9 of default timer toggles
0	1	Reset when bit 15 of default timer toggles
1	0	Reset when bit 19 of default timer toggles
1	1	Disable watchdog

These bits are described in subsection 12.11.14. They are available only in privilege mode immediately after power-up.

Once a time-out period has been selected as shown above, the alternate key bytes, 55h (first) and AAh must be written to the WDRST register (peripheral frame 4, 104Eh) to avoid issuing a watchdog-originated reset. The only exception to this occurs when the default counter is cleared by the PACT module; in this case, a watchdog-originated reset will occur, unless the correct keyword (55h/AAh) has been written since the previous clear.

The watchdog timer is stopped in standby mode and halt mode.

12.9 Mini-Serial Communications Interface (SCI)

The mini-SCI works as a simplified full duplex UART by transmitting 8-bit words with a fixed format of one start and one stop bit.

- If parity transmission is required, the parity bit must be calculated by the CPU and placed in the transmit buffer as part of the 8-bit word. Parity reception is facilitated by the parity result bit. This bit allows the processor to check for parity errors by comparing the PACT PARITY bit (SCICTLP.5) against 0 or 1 for even or odd parity. Hence, there is no parity error bit to be checked by the processor.

- There is no overrun detection. The PACT SCI has a shift register and a buffer register. This gives the program a full data byte reception time to read the previous byte before it is overwritten.

During reception, the start bit is detected on the falling edge and then sampled again in the center of the bit to avoid false detection. All other bits are sampled once at their centers. If at least one stop bit is not detected when it is expected, the framing error flag (PACT FE bit, P045.3) is set. This bit will remain set until cleared by reset, by SCI software reset, or by a zero written to it.

The bit rate (baud) is determined by setting the maximum virtual timer value to one quarter of the desired bit time minus 2. For example, if the system clock period is 200 ns and the prescale value is 5, then the PACT resolution is 1 µs. If a baud of 9600 is desired, the maximum virtual timer value should be:

$$\text{Max Virtual Timer Value} = \frac{1}{(\text{Baud})\ (4)\ (\text{PACT Resolution})} - 2$$

$$= \frac{1}{(9600)\ (4)\ (10^{-6})} - 2$$

$$= 24$$

The software selectable bauds (RX and TX can be different) are set up as shown in subsection 12.6.3. Receive and transmit operations can be stopped or started by using the control bits within the baud rate definition command.

The data being received and transmitted is accessed in the same peripheral frame (4) as the control bits are. Received data is held at 1046h; transmitted data at 1047h.

12.10 PWM Example

The following three-part routine is an example of how to set up a pulse-width-modulated signal with the PACT module:

- The first part defines the bytes that will make up the command/definition area.
- The second part copies the command/definition area bytes from ROM to the dual-port RAM.
- The third part sets up the PACT peripheral file.

Refer to Example 12–1 on page 12-34 as you read the following subsections.

12.10.1 Defining the Command/Definition Area

The macro file PACT.H simplifies the task of setting up the command/definition area. See Appendix I of this manual. Since this file is subject to change as improvements are found, TI recommends that you download the latest version of this file from the microcontroller bulletin board.

To set up the PWM signal,

- A virtual timer definition must establish the timer period.
- A Standard Compare command follows to determine the period and the polarity of the signal.
- Since the PACT command/definition area cannot start with a definition, an additional Standard Compare command is inserted at the beginning with only the D16 bit set.
- Line 8 of the routine causes the bytes that will become the commands and definitions to be located in a separate section. In this example, this section starts at location 7800h.
- Line 10 is the dummy Standard Compare command.
- Line 11 is the virtual timer definition. The period is set to 1000 µs, and the virtual timer is enabled. Note that the macro takes care of subtracting two from the maximum count value as it creates the proper byte sequence.
- Line 12 is the Standard Compare command that sets the period to 800 µs or 80% and selects output pin 1. The default value is to set the pin high on compare equal and opp_act is selected to cause the pin to go low when the timer is reset. Notice how multiple actions are concatenated with the | operator in this command.

12.10.2 Copying the Command/Definition Area to Dual-Port RAM

Since the dual-port RAM must be initialized after power-up, the initial values for the command/definition area were defined in nonvolatile memory; they must be copied from the nonvolatile memory to the dual-port RAM. Since the PACT module works its way through memory from high addresses to lower addresses but the assembler works through memory from low addresses to higher addresses, this routine flips the table end for end as it is copied into the dual-port RAM. This makes the table easier to read.

Two variables must be defined before this routine is used.

- The first, cmd_st, is the start of the command/definition area (the largest address in that area). Its value is dependent on the mode used and on the size of the desired circular buffer. cmd_st is defined in line 16 of this routine.
- The second variable, len, is defined in line 13 as the number of bytes in the command/ definition area.

12.10.3 Initializing the PACT Peripheral Frame

The last part of the PWM routine sets up the PACT peripheral frame.

1) First, the mode is chosen in line 27.
2) Next, the command/definition start is chosen in line 28, and the end is chosen in line 29.
3) Finally, the timer resolution is set to one microsecond, and the command/ definition area is enabled in line 31. This line causes the PWM signal to start.

You should always verify that the PACT clock prescale value allows enough time slots for the entire command/definition area. In this example, four time slots are required—one for each Standard Compare command and two for the virtual timer definition. The prescale value of 5 provides 12 time slots, which is more than enough for this application.

Example 12–1. Performing a PWM

```
0001  ;This is an example program to do PWM using the PACT module
0002      .include "PACT.H"
0003
0004  ;MACRO DESCRIPTION
0005  ;stdcmp <compare value>,<pin>,<actions>
0006  ;virtmr <period>,<actions>,<initial timer value>
0007
0008      .sect "pact",7800h
0009  ;PACT instructions to do PWM
0010  table  stdcmp 0,0,nxt_def   ;dummy cmd, next line=def
0001  #   .byte  0,0,1,0
0011      virtmr    1000,enable       ;period = 1000 uSec
0001  #   .byte  0,0,52,31
0012      stdcmp    800,op1,opp_act|enable ;80% duty, pin 1
0001  #   .byte  32,3,0,10
0013  len.equ   $-table
0014
0015      .text  6000h
0016  cmd_st .equ   0EBh
0017  start  mov#7,P04F   ;disable the watchdog
0018  ;copy PACT commands/def. into ram
0019      mov    #len,b     ;length of cmd/def area
0020      movw   #(cmd_st-len+1),r3      ;R2:R3 points to area
0021  loop   mov    table-1(b),a
0022         mov    a,@r3
0023         inc    r3
0024         djnz   b,loop
0025
0026  ;set up the peripheral file
0027      mov    #07,p04f                ;set to mode B
0028      mov    #cmd_st,p041            ;cmd/def start at 0ebh
0029      mov    #(cmd_st-len+1),p042    ;cmd/def end = 0E0h
0030  ;set prescale to 5, 1 usec res, enable cmd/def area
0031   .. mov    #034h,p040
0032
0033  ;PWM running without processor intervention
0034      idle
0035      .end
```

12.11 PACT Control Registers

The PACT module is controlled and accessed through registers in peripheral frame 4. These registers are listed in Table 12–5 and described in the following subsections. The bits shown in shaded boxes in Table 12–5 are privilege mode bits; that is, they can be written to only in the privilege mode.

Note:

Be careful using the AND, OR, XOR, CMPBIT, SBIT0, or SBIT1 instruction to modify any registers that contain status flags. The read/modify/write nature of these instructions can inadvertently clear an interrupt flag that was set between the read and the write cycles. If the state of the nonflag bits is known, the MOV #n1,Pn2 instruction can be used. If the state of the nonflag bits is not known, a sequence similar to the example shown below should be used.

```
;Clearing an interrupt flag
      MOV    P04n,A
      OR     #flag_mask,A   ;set all flag bits to one
      AND    #desired_flag,A ;clear the desired flag bit
      MOV    A,P04n
```

Table 12–5. Peripheral File Frame 4: PACT Control Registers

Desig-nation	ADDR	PF	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PACTSCR	1040h	P040	DEFTIM OVRFL INT ENA	DEFTIM OVRFL INT FLAG	CMD/DEF AREA ENA	FAST MODE SELECT	PACT PRESCALE SELECT 3	PACT PRESCALE SELECT 2	PACT PRESCALE SELECT 1	PACT PRESCALE SELECT 0
CDSTART	1041h	P041	CMD/DEF AREA INT ENA	—	CMD/DEF AREA START BIT 5	CMD/DEF AREA START BIT 4	CMD/DEF AREA START BIT 3	CMD/DEF AREA START BIT 2	—	—
CDEND	1042h	P042	—	CMD/DEF AREA END BIT 6	CMD/DEF AREA END BIT 5	CMD/DEF AREA END BIT 4	CMD/DEF AREA END BIT 3	CMD/DEF AREA END BIT 2	—	—
BUFPTR	1043h	P043	1	1	BUFFER POINTER BIT 5	BUFFER POINTER BIT 4	BUFFER POINTER BIT 3	BUFFER POINTER BIT 2	BUFFER POINTER BIT 1	—
	1044h	P044	Reserved							
SCICTLP	1045h	P045	PACT RXRDY	PACT TXRDY	PACT PARITY	PACT FE	PACT SCI RX INT ENA	PACT SCI TX INT ENA	—	PACT SCI SW RESET
RXBUFP	1046h	P046	PACT RXDT7	PACT RXDT6	PACT RXDT5	PACT RXDT4	PACT RXDT3	PACT RXDT2	PACT RXDT1	PACT RXDT0
TXBUFP	1047h	P047	PACT TXDT7	PACT TXDT6	PACT TXDT5	PACT TXDT4	PACT TXDT3	PACT TXDT2	PACT TXDT1	PACT TXDT0
OPSTATE	1048h	P048	PACT OP8 STATE	PACT OP7 STATE	PACT OP6 STATE	PACT OP5 STATE	PACT OP4 STATE	PACT OP3 STATE	PACT OP2 STATE	PACT OP1 STATE
CDFLAGS	1049h	P049	CMD/DEF INT 7 FLAG	CMD/DEF INT 6 FLAG	CMD/DEF INT 5 FLAG	CMD/DEF INT 4 FLAG	CMD/DEF INT 3 FLAG	CMD/DEF INT 2 FLAG	CMD/DEF INT 1 FLAG	CMD/DEF INT 0 FLAG
CPCTL1	104Ah	P04A	CP2 INT ENA	CP2 INT FLAG	CP2 CAPT RISING EDGE	CP2 CAPT FALLING EDGE	CP1 INT ENA	CP1 INT FLAG	CP1 CAPT RISING EDGE	CP1 CAPT FALLING EDGE
CPCTL2	104Bh	P04B	CP4 INT ENA	CP4 INT FLAG	CP4 CAPT RISING EDGE	CP4 CAPT FALLING EDGE	CP3 INT ENA	CP3 INT FLAG	CP3 CAPT RISING EDGE	CP3 CAPT FALLING EDGE
CPCTL3	104Ch	P04C	CP6 INT ENA	CP6 INT FLAG	CP6 CAPT RISING EDGE	CP6 CAPT FALLING EDGE	CP5 INT ENA	CP5 INT FLAG	CP5 CAPT RISING EDGE	CP5 CAPT FALLING EDGE
CPPRE	104Dh	P04D	BUFFER HALF/FULL INT ENA	BUFFER HALF/FULL INT FLAG	INPUT CAPT PRESCALE SELECT 3	INPUT CAPT PRESCALE SELECT 2	INPUT CAPT PRESCALE SELECT 1	CP6 EVENT ONLY	EVENT COUNTER SW RESET	OP SET/CLR SELECT
WDRST	104Eh	P04E	Watchdog Reset Key							
PACTPRI	104Fh	P04F	PACT STEST	—	PACT GROUP 1 PRIORITY	PACT GROUP 2 PRIORITY	PACT GROUP 3 PRIORITY	PACT MODE SELECT	PACT WD PRESCALE SELECT 1	PACT WD PRESCALE SELECT 0

12

12.11.1 Set-Up Control Register (PACTSCR)

The PACTSCR register determines the time base for the PACT module, enables the command/definition area, and controls the default timer overflow.

Set-Up Control Register (PACTSCR)
[Memory Address – 1040h]

Bit #	7	6	5	4	3	2	1	0
P040	DEFTIM OVRFL INT ENA	DEFTIM OVRFL INT FLAG	CMD/DEF AREA ENA	FAST MODE SELECT	PACT PRE-SCALE SELECT3	PACT PRE-SCALE SELECT2	PACT PRE-SCALE SELECT1	PACT PRE-SCALE SELECT0
	RW–0	RC–0	RW–0	RP–0	RP–0	RP–0	RP–0	RP–0

R = Read, W = Write, P = Privilege write only, C = Clear, *–n* = Value of the bit after the register is reset

Bit 0 – 3 **PACT PRESCALE SELECT0 – 3.**

These four bits select a prescaler divide rate for the PACT module. The bits specify the divide of the system clock from +2 to +16, giving 15 possible choices. The actual divide rate is also determined by the value of the FAST MODE SELECT bit. The possible combinations are shown below the FAST MODE SELECT bit description.

Bit 4 **FAST MODE SELECT.**

This bit determines if the system clock is divided by 8 before entering into the 4-bit prescale. This bit, as well as the PACT PRESCALE SELECT 0–3 bits, determines the time base for the PACT module. The possible combinations are shown below.

Divide Rate		
PACT PRESCALE SELECT	**FAST MODE SELECT**	
3 2 1 0	1	0
0 0 0 0	2	16
0 0 0 1	2	16
0 0 1 0	3	24
0 0 1 1	4	32
0 1 0 0	5	40
0 1 0 1	6	48
0 1 1 0	7	56
0 1 1 1	8	64
1 0 0 0	9	72
1 0 0 1	10	82
1 0 1 0	11	88
1 0 1 1	12	96
1 1 0 0	13	104
1 1 0 1	14	112
1 1 1 0	15	120
1 1 1 1	16	128

Bit 5 **CMD/DEF AREA ENA.** Command and Definition Area Enable.

This bit determines if the command/definition area of the dual-port RAM is enabled. This allows the PACT module to use this area for commands and definitions.

0 = Command/definition area is ignored.
1 = Enables the commands and definitions.

Bit 6 **DEFTIM OVRFL INT FLAG**. Default Timer Overflow Interrupt Flag.

This bit indicates the status of the PACT default timer overflow interrupt. This bit is cleared by reset or when a zero is written to it; it is set by overflow.

0 = Default timer overflow interrupt inactive.
1 = Default timer overflow interrupt pending.

Bit 7 **DEFTIM OVRFL INT ENA**. Default Timer Overflow Interrupt Enable.

This bit controls the default timer overflow interrupting capability.

0 = Disables interrupt.
1 = Enables interrupt.

12.11.2 Command/Definition Area Start Register (CDSTART)

The CDSTART register defines the starting address of the command/definition area and enables the interrupts for that area.

Command/Definition Area Start Register (CDSTART)
[Memory Address – 1041h]

Bit #	7	6	5	4	3	2	1	0
P041	CMD/DEF AREA INT ENA	—	CMD/DEF AREA START BIT 5	CMD/DEF AREA START BIT 4	CMD/DEF AREA START BIT 3	CMD/DEF AREA START BIT 2	—	—
	RW–0		RW–0	RW–0	RW–0	RW–0		

R = Read, W = Write, –*n* = Value of the bit after the register is reset

Bit 0, 1 **Reserved.** Read data is indeterminate.

Bit 2 – 5 **CMD/DEF AREA START BIT 2–5**. Command and Definition Area Start Bits 2 – 5.

These bits define the start address of the Command / Definition Area. There are 16 possible locations for this area to start. The address is the same as if you consider bits 7, 6, 1, and 0 of this register to be set as equal to 1. A table of the bits and the corresponding addresses is shown below.

CMD/DEF AREA START Bit				CMD/DEF Area Start		
5	4	3	2	Address	Register	Notes
0	0	0	0	00C3h	R0C3	
0	0	0	1	00C7h	R0C7	
0	0	1	0	00CBh	R0CB	
0	0	1	1	00CFh	R0CF	
0	1	0	0	00D3h	R0D3	
0	1	0	1	00D7h	R0D7	
0	1	1	0	00DBh	R0DB	
0	1	1	1	00DFh	R0DF	
1	0	0	0	00E3h	R0E3	
1	0	0	1	00E7h	R0E7	
1	0	1	0	00EBh	R0EB	
1	0	1	1	00EFh	R0EF	†
1	1	0	0	00F3h	R0F3	†
1	1	0	1	00F7h	R0F7	Invalid
1	1	1	0	00FBh	R0FB	Invalid
1	1	1	1	00FFh	R0FF	Invalid

† Invalid in Mode B.

Bit 6 **Reserved.** Read data is indeterminate.

Bit 7 **CMD/DEF AREA INT ENA**. Command and Definition Area Interrupt Enable.

This bit enables interrupts from the command/definition area.

0 = Disables interrupt.
1 = Enables interrupt.

12.11.3 Command/Definition Area End Register (CDEND)

The CDEND register defines the end address of the command/definition area.

Command/Definition Area End Register (CDEND)
[Memory Address – 1042h]

Bit #	7	6	5	4	3	2	1	0
P042	—	CMD/DEF AREA END BIT 6	CMD/DEF AREA END BIT 5	CMD/DEF AREA END BIT 4	CMD/DEF AREA END BIT 3	CMD/DEF AREA END BIT 2	—	—
		RW–0	RW–0	RW–0	RW–0	RW–0		

R = Read, W = Write, *–n* = Value of the bit after the register is reset

Bit 0, 1 **Reserved.** Read data is indeterminate.

Bit 2 – 6 **CMD/DEF AREA END BIT 2–6.** Command/Definition Area End Bits 2–6.

These bits define the end address of the command/definition area. There are 32 possible locations for the command/definition area end. The address is the same as if you consider bit 7 of this register to be set to 1, and bit 1 and bit 0 to be set to 0. A table of the bits and the corresponding addresses is shown below.

CMD/DEF AREA END Bit					CMD/DEF END	
6	**5**	**4**	**3**	**2**	**Address**	**Register**
0	0	0	0	0	0080h	R080
0	0	0	0	1	0084h	R084
0	0	0	1	0	0088h	R088
0	0	0	1	1	008Ch	R08C
0	0	1	0	0	0090h	R090
0	0	1	0	1	0094h	R094
0	0	1	1	0	0098h	R098
0	0	1	1	1	009Ch	R09C
0	1	0	0	0	00A0h	R0A0
0	1	0	0	1	00A4h	R0A4
0	1	0	1	0	00A8h	R0A8
0	1	0	1	1	00ACh	R0AC
0	1	1	0	0	00B0h	R0B0
0	1	1	0	1	00B4h	R0B4
0	1	1	1	0	00B8h	R0B8
0	1	1	1	1	00BCh	R0BC
1	0	0	0	0	00C0h	R0C0
1	0	0	0	1	00C4h	R0C4
1	0	0	1	0	00C8h	R0C8
1	0	0	1	1	00CCh	R0CC
1	0	1	0	0	00D0h	R0D0
1	0	1	0	1	00D4h	R0D4
1	0	1	1	0	00D8h	R0D8
1	0	1	1	1	00DCh	R0DC
1	1	0	0	0	00E0h	R0E0
1	1	0	0	1	00E4h	R0E4
1	1	0	1	0	00E8h	R0E8
1	1	0	1	1	00ECh	R0EC
1	1	1	0	0	00F0h	R0F0
1	1	1	0	1	00F4h	R0F4
1	1	1	1	0	00F8h	R0F8
1	1	1	1	1	00FCh	R0FC

Bit 7 **Reserved.** Read data is indeterminate.

12.11.4 Buffer Pointer Register (BUFPTR)

The BUFPTR register defines the address of the buffer pointer.

Buffer Pointer Register (BUFPTR)
[Memory Address – 1043h]

Bit #	7	6	5	4	3	2	1	0
P043	1	1	BUFFER POINTER BIT 5	BUFFER POINTER BIT 4	BUFFER POINTER BIT 3	BUFFER POINTER BIT 2	BUFFER POINTER BIT 1	0
	R–1	R–1	R–1	R–1	R–0	R–0	R–0	R–0

R = Read, *–n* = Value of the bit after the register is reset

Bit 0 **Reserved.**

Bit 1 – 5 **BUFFER POINTER BIT 1–5.**

These bits define the address of the buffer pointer. The buffer pointer points to the next available address out of 32 possible locations in the circular buffer. These addresses are the same as if you consider bit 7 and bit 6 of this register to be set to 1, and bit 0 to be set to 0. A table of the bits and the corresponding addresses is shown below.

BUFFER POINTER Bit					Buffer Pointer	
5	4	3	2	1	Address	Register
0	0	0	0	0	00C0h	R0C0
0	0	0	0	1	00C2h	R0C2
0	0	0	1	0	00C4h	R0C4
0	0	0	1	1	00C6h	R0C6
0	0	1	0	0	00C8h	R0C8
0	0	1	0	1	00CAh	R0CA
0	0	1	1	0	00CCh	R0CC
0	0	1	1	1	00CEh	R0CE
0	1	0	0	0	00D0h	R0D0
0	1	0	0	1	00D2h	R0D2
0	1	0	1	0	00D4h	R0D4
0	1	0	1	1	00D6h	R0D6
0	1	1	0	0	00D8h	R0D8
0	1	1	0	1	00DAh	R0DA
0	1	1	1	0	00DCh	R0DC
0	1	1	1	1	00DEh	R0DE
1	0	0	0	0	00E0h	R0E0
1	0	0	0	1	00E2h	R0E2
1	0	0	1	0	00E4h	R0E4
1	0	0	1	1	00E6h	R0E6
1	0	1	0	0	00E8h	R0E8
1	0	1	0	1	00EAh	R0EA
1	0	1	1	0	00ECh	R0EC
1	0	1	1	1	00EEh	R0EE
1	1	0	0	0	00F0h	R0F0
1	1	0	0	1	00F2h	R0F2
1	1	0	1	0	00F4h	R0F4
1	1	0	1	1	00F6h	R0F6
1	1	1	0	0	00F8h	R0F8
1	1	1	0	1	00FAh	R0FA
1	1	1	1	0	00FCh	R0FC
1	1	1	1	1	00FEh	R0FE

Bit 6, 7 **Reserved.**

12.11.5 PACT-SCI Control Register (SCICTLP)

The SCICTLP register controls the functions of the mini-SCI.

PACT-SCI Control Register (SCICTLP)
[Memory Address – 1045h]

Bit #	7	6	5	4	3	2	1	0
P045	PACT RXRDY	PACT TXRDY	PACT PARITY	PACT FE	PACT SCI RX INT ENA	PACT SCI TX INT ENA	—	PACT SCI SW RESET
	RC–0	R–1	R–0	RC–0	RW–0	RW–0		RW–0

R = Read, W = Write, C = Clear, –*n* = Value of the bit after the register is reset

Bit 0 **PACT SCI SW RESET**. PACT SCI Software Reset.

When set, this bit puts the SCI into a software reset state so that the parameters of the SCI can be set up. This bit must be cleared to allow the SCI to function.

0 = SCI in operating mode.
1 = SCI in software reset mode.

Bit 1 **Reserved.** Read data is indeterminate.

Bit 2 **PACT SCI TX INT ENA**. PACT SCI Transmit Interrupt Enable.

This bit enables the interrupt to occur when the transmit buffer is empty.

0 = Does not generate an interrupt when the transmit buffer is empty.
1 = Generates an interrupt when the transmit buffer is empty.

Bit 3 **PACT SCI RX INT ENA**. PACT SCI Receive Interrupt Enable.

This bit enables the interrupt to occur when the receive buffer is full.

0 = Does not generate an interrupt when the receive buffer is full.
1 = Generates an interrupt when the receive buffer is full.

Bit 4 **PACT FE**. PACT Framing Error.

This bit is a flag to show that a framing error was detected. This bit will remain set until cleared by a PACT SCI software reset, by a system reset, or when a zero is written to it.

0 = No framing error.
1 = Framing error was detected.

Bit 5 **PACT PARITY**. PACT Receive Data Parity Bit.

This bit is set as the result of the incoming parity calculation. To perform a parity check on incoming data, this bit is compared to a 0 or 1 for even or odd parity.

0 = Received data was even parity.
1 = Received data was odd parity.

Bit 6 **PACT TXRDY**. PACT Transmit Ready.

This bit shows when the transmit buffer is empty. This bit is set by a system reset, by the PACT SCI software reset, or when the SCI TX DATA register has been shifted out.

0 = Transmit buffer is full.
1 = Transmit buffer is empty.

Bit 7 **PACT RXRDY**. PACT Receive Ready.

This bit shows when the receive buffer is full. This bit is cleared by a system reset, by the PACT SCI software reset, when a zero is written to it, or when the SCI RX DATA register is read.

0 = Receive buffer is empty.
1 = Receive buffer is full.

12.11.6 PACT-SCI RX Data Register (RXBUFP)

This register contains the data received by the SCI.

PACT-SCI RX Data Register (RXBUFP)
[Memory Address – 1046h]

Bit #	7	6	5	4	3	2	1	0
P046	PACT RXDT7	PACT RXDT6	PACT RXDT5	PACT RXDT4	PACT RXDT3	PACT RXDT2	PACT RXDT1	PACT RXDT0
	R–0	R–0	R–0	R–0	R–0	R–0	R–0	R–0

R = Read, –*n* = Value of the bit after the register is reset

Bits 0 – 7 **PACT RXDT 0 –7**. PACT Receive Data 0 – 7.

12.11.7 PACT-SCI TX Data Register (TXBUFP)

This register contains the data to be transmitted by the SCI.

PACT-SCI TX Data Register (TXBUFP)
[Memory Address – 1047h]

Bit #	7	6	5	4	3	2	1	0
P047	PACT TXDT7	PACT TXDT6	PACT TXDT5	PACT TXDT4	PACT TXDT3	PACT TXDT2	PACT TXDT1	PACT TXDT0
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R = Read, W = Write, –*n* = Value of the bit after the register is reset

Bits 0 – 7 **PACT TXDT 0 –7**. PACT Transmit Data 0 – 7.

12.11.8 Output Pins 1–8 State Register (OPSTATE)

The OPSTATE register contains information about the current state of the output pins.

Output Pin 1–8 State Register (OPSTATE)
[Memory Address – 1048h]

Bit #	7	6	5	4	3	2	1	0
P048	PACT OP8 STATE	PACT OP7 STATE	PACT OP6 STATE	PACT OP5 STATE	PACT OP4 STATE	PACT OP3 STATE	PACT OP2 STATE	PACT OP1 STATE
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R = Read, W = Write, –*n* = Value of the bit after the register is reset

Bit 0 – 7 **PACT OP1 – 8 STATE**. PACT Output Pins 1 – 8 State Bits.

These bits reflect the current state of the output pins OP8 to OP1. Each bit is the actual state of the corresponding pin. Writing a 1 to any bit in this register will modify the corresponding output pin as determined by the OP SET/CLR SELECT bit (P04D.0).

Bit OPx Write	OP SET/CLR SELECT	Result
1	1	PACT OPx STATE = 1
1	0	PACT OPx STATE = 0
0	x	PACT OPx STATE remains unchanged

Upon reset, all pins are initialized to the low state.

Example 1, if OP SET/CLR SELECT = 1

```
11001011    OP STATE Register
11110000    Write to OP STATE Register
--------
11111011    New value in OP STATE Register.
```

Example 2, if OP SET/CLR SELECT = 0

```
11001011    OP STATE Register
11110000    Write to OP STATE Register
--------
00001011    New value in OP STATE Register.
```

12

12.11.9 Command/Definition Entry Flags Register (CDFLAGS)

The CDFLAGS register contains information about the command/definition interrupts.

Command/Definition Entry Flags Register (CDFLAGS)
[Memory Address – 1049h]

Bit #	7	6	5	4	3	2	1	0
P049	CMD/DEF INT 7 FLAG	CMD/DEF INT 6 FLAG	CMD/DEF INT 6 FLAG	CMD/DEF INT 4 FLAG	CMD/DEF INT 3 FLAG	CMD/DEF INT 2 FLAG	CMD/DEF INT 1 FLAG	CMD/DEF INT 0 FLAG
	RC–0	RC–0	RC–0	RC–0	RC–0	RC–0	RC–0	RC–0

R = Read, C = Clear, –*n* = Value of the bit after the register is reset

Bit 0 – 7 **CMD/DEF INT 0 – 7 FLAG**. Command/Definition Interrupt 0 – 7 Flag.

These bits are the interrupt flags for the command/definition area. If an interrupt has been enabled in a 4-byte command or definition, then the appropriate bit in this register will be set when the interrupt conditions are met. The actual bit set is determined by the command or definition's entry address. The entry address is derived from bits 2, 3, and 4 of the address. If the command or definition is at address 006Ch, 006Dh, 006Eh, or 006Fh (32 bits), then 0110 11xx is the binary value of the address. The entry address comes from bits 4–3–2 = 011 = Entry 3. Thus, the flag associated with entry address 3 will be used when this command or definition causes an interrupt.

These flags are not affected by the CMD/DEF AREA INT ENA bit (CDSTART.7).

CMD/DEF INT FLAG Set	Command or Definition Entry That Generated Interrupt Request
CMD/DEF INT 0 FLAG	(Entry Address >> 2) ENTRY mod 8 = 0
CMD/DEF INT 1 FLAG	(Entry Address >> 2) ENTRY mod 8 = 1
CMD/DEF INT 2 FLAG	(Entry Address >> 2) ENTRY mod 8 = 2
CMD/DEF INT 3 FLAG	(Entry Address >> 2) ENTRY mod 8 = 3
CMD/DEF INT 4 FLAG	(Entry Address >> 2) ENTRY mod 8 = 4
CMD/DEF INT 5 FLAG	(Entry Address >> 2) ENTRY mod 8 = 5
CMD/DEF INT 6 FLAG	(Entry Address >> 2) ENTRY mod 8 = 6
CMD/DEF INT 7 FLAG	(Entry Address >> 2) ENTRY mod 8 = 7

12.11.10 Set-Up CP Control Register 1 (CPCTL1)

The CPCTL1 register controls the functions of the CP1 and CP2 pins.

Set-Up CP Control Register 1 (CPCTL1)
[Memory Address – 104Ah]

Bit #	7	6	5	4	3	2	1	0
P04A	CP2 INT ENA	CP2 INT FLAG	CP2 CAPT RISING EDGE	CP2 CAPT FALLING EDGE	CP1 INT ENA	CP1 INT FLAG	CP1 CAPT RISING EDGE	CP1 CAPT FALLING EDGE
	RW–0	RC–0	RW–0	RW–0	RW–0	RC–0	RW–0	RW–0

R = Read, W = Write, C = Clear, *–n* = Value of the bit after the register is reset

Bit 0 **CP1 CAPT FALLING EDGE**. CP1 Capture Falling Edge.

This bit selects the falling edge on pin CP1 to cause a timer capture. See the table following the bit 1 description for all possible combinations.

Bit 1 **CP1 CAPT RISING EDGE**. CP1 Capture Rising Edge.

This bit selects the rising edge on pin CP1 to cause a timer capture. The table below shows all possible combinations.

CPx CAPT RISING EDGE	CPx CAPT FALLING EDGE	Capture On Selected Edges
0	0	Disables Captures
0	1	Captures on Falling Edges Only
1	0	Captures on Rising Edges Only
1	1	Captures on Both Rising and Falling Edges

Bit 2 **CP1 INT FLAG**. CP1 Interrupt Flag.

This bit indicates that the selected edge has occurred on pin CP1. This bit must be cleared by the program during an interrupt routine when CP1 INT ENA is set.

0 = Capture interrupt from selected edge of CP1 inactive.
1 = Capture interrupt from selected edge of CP1 pending.

Bit 3 **CP1 INT ENA**. CP1 Interrupt Enable.

If set, this bit enables the interrupt when the selected edge occurs on pin CP1.

0 = Disables interrupt.
1 = Enables interrupt.

Bit 4 **CP2 CAPT FALLING EDGE**. CP2 Capture Falling Edge.

This bit selects the falling edge on pin CP2 to cause a timer capture. See the table following the bit 1 description for all possible combinations.

Bit 5 **CP2 CAPT RISING EDGE**. CP2 Capture Rising Edge.

This bit selects the rising edge on pin CP2 to cause a timer capture. See the table following the bit 1 description for all possible combinations.

Bit 6 **CP2 INT FLAG**. CP2 Interrupt Flag.

This bit indicates that the selected edge has occurred on pin CP2. This bit must be cleared by the program during an interrupt routine when CP2 INT ENA is set.

0 = Capture interrupt from selected edge of CP2 inactive.
1 = Capture interrupt from selected edge of CP2 pending.

Bit 7 **CP2 INT ENA**. CP2 Interrupt Enable.

If set, this bit enables the interrupt when the selected edge occurs on pin CP2.

0 = Disables interrupt.
1 = Enables interrupt.

12.11.11 Set-Up CP Control Register 2 (CPCTL2)

The CPCTL2 register controls the functions of the CP3 and CP4 pins.

Set-Up CP Control Register 2 (CPCTL2)
[Memory Address – 104Bh]

Bit #	7	6	5	4	3	2	1	0
P04B	CP4 INT ENA	CP4 INT FLAG	CP4 CAPT RISING EDGE	CP4 CAPT FALLING EDGE	CP3 INT ENA	CP3 INT FLAG	CP3 CAPT RISING EDGE	CP3 CAPT FALLING EDGE
	RW–0	RC–0	RW–0	RW–0	RW–0	RC–0	RW–0	RW–0

R = Read, W = Write, C = Clear, *–n* = Value of the bit after the register is reset

Bit 0 **CP3 CAPT FALLING EDGE**. CP3 Capture Falling Edge.

This bit selects the falling edge on pin CP3 to cause a timer capture. See the table following the bit 1 description for all possible combinations

Bit 1 **CP3 CAPT RISING EDGE**. CP3 Capture Rising Edge.

This bit selects the rising edge on pin CP3 to cause a timer capture. The table below shows all possible combinations.

CPx CAPT RISING EDGE	CPx CAPT FALLING EDGE	Capture On Selected Edges
0	0	Disables Captures
0	1	Captures on Falling Edges Only
1	0	Captures on Rising Edges Only
1	1	Captures on Both Rising and Falling Edges

Bit 2 **CP3 INT FLAG**. CP3 Interrupt Flag.

This bit indicates that the selected edge has occurred on pin CP3. This bit must be cleared by the program during an interrupt routine when CP3 INT ENA is set.

0 = Capture interrupt from selected edge of CP3 inactive.
1 = Capture interrupt from selected edge of CP3 pending.

Bit 3 **CP3 INT ENA**. CP3 Interrupt Enable.

If set, this bit enables the interrupt when the selected edge occurs on pin CP3.

0 = Disables interrupt.
1 = Enables interrupt.

12

Bit 4 **CP4 CAPT FALLING EDGE**. CP4 Capture Falling Edge.

This bit selects the falling edge on pin CP4 to cause a timer capture. See the table following the bit 1 description for all possible combinations.

Bit 5 **CP4 CAPT RISING EDGE**. CP4 Capture Rising Edge.

This bit selects the rising edge on pin CP4 to cause a timer capture. See the table following the bit 1 description for all possible combinations.

Bit 6 **CP4 INT FLAG**. CP4 Interrupt Flag.

This bit indicates that the selected edge has occurred on pin CP4. This bit must be cleared by the program during an interrupt routine when CP4 INT ENA is set.

0 = Capture interrupt from selected edge of CP4 inactive.
1 = Capture interrupt from selected edge of CP4 pending.

Bit 7 **CP4 INT ENA**. CP4 Interrupt Enable.

If set, this bit enables the interrupt when the selected edge occurs on pin CP4.

0 = Disables interrupt.
1 = Enables interrupt.

12.11.12 Set-Up CP Control Register 3 (CPCTL3)

The CPCTL3 register controls the functions of the CP5 and CP6 pins.

Set-Up CP Control Register 3 (CPCTL3)
[Memory Address – 104Ch]

Bit #	7	6	5	4	3	2	1	0
P04C	CP6 INT ENA	CP6 INT FLAG	CP6 CAPT RISING EDGE	CP6 CAPT FALLING EDGE	CP5 INT ENA	CP5 INT FLAG	CP5 CAPT RISING EDGE	CP5 CAPT FALLING EDGE
	RW–0	RC–0	RW–0	RW–0	RW–0	RC–0	RW–0	RW–0

R = Read, W = Write, C = Clear, *–n* = Value of the bit after the register is reset

Bit 0 **CP5 CAPT FALLING EDGE**. CP5 Capture Falling Edge.

This bit selects the falling edge on pin CP5 to cause a timer capture. See the table following the bit 1 description for all possible combinations.

Bit 1 **CP5 CAPT RISING EDGE**. CP5 Capture Rising Edge.

This bit selects the rising edge on pin CP5 to cause a timer capture. The table below shows all possible combinations.

CPx CAPT RISING EDGE	CPx CAPT FALLING EDGE	Capture On Selected Edges
0	0	Disables Captures
0	1	Captures on Falling Edges Only
1	0	Captures on Rising Edges Only
1	1	Captures on Both Rising and Falling Edges

Bit 2 **CP5 INT FLAG**. CP5 Interrupt Flag.

This bit indicates that the selected edge has occurred on pin CP5. This bit must be cleared by the program during an interrupt routine when CP5 INT ENA is set.

0 = Capture interrupt from selected edge of CP5 inactive.
1 = Capture interrupt from selected edge of CP5 pending.

Bit 3 **CP5 INT ENA**. CP5 Interrupt Enable.

If set, this bit enables the interrupt when the selected edge occurs on pin CP5.

0 = Disables interrupt.
1 = Enables interrupt.

Bit 4 **CP6 CAPT FALLING EDGE**. CP6 Capture Falling Edge.

This bit selects the falling edge on pin CP6 to cause a timer capture. See the table following the bit 1 description for all possible combinations.

Bit 5 **CP6 CAPT RISING EDGE**. CP6 Capture Rising Edge.

This bit selects the rising edge on pin CP6 to cause a timer capture. See the table following the bit 1 description for all possible combinations.

Bit 6 **CP6 INT FLAG**. CP6 Interrupt Flag.

This bit indicates that the selected edge has occurred on pin CP6. This bit must be cleared by the program during an interrupt routine when CP6 INT ENA is set.

0 = Capture interrupt from selected edge of CP6 inactive.
1 = Capture interrupt from selected edge of CP6 pending.

Bit 7 **CP6 INT ENA**. CP6 Interrupt Enable.

If set, this bit enables the interrupt when the selected edge occurs on pin CP6.

0 = Disables interrupt.
1 = Enables interrupt.

12.11.13 CP Input Control Register (CPPRE)

The CPPRE register controls input and output functions.

CP Input Control Register (CPPRE)
[Memory Address – 104Dh]

Bit #	7	6	5	4	3	2	1	0
P04D	BUFFER HALF/FULL INT ENA	BUFFER HALF/FULL INT FLAG	INPUT CAPT PRE-SCALE SELECT3	INPUT CAPT PRE-SCALE SELECT2	INPUT CAPT PRE-SCALE SELECT1	CP6 EVENT ONLY	EVENT COUNTER SW RESET	OP SET/CLR SELECT
	RW–0	RC–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

R = Read, W = Write, C = Clear, –*n* = Value of the bit after the register is reset

Bit 0 **OP SET/CLR SELECT**. Output Pin Set/Clear Write Function Select.

This bit controls how the outputs OP1 to OP8 are set or cleared by software.

- When OP SET/CLR = 1, a write to P048 will cause the output pins corresponding to the locations that were written as 1 to be set in the high state. The output pins corresponding to the locations that were written as 0 remain unchanged.
- When OP SET/CLR = 0, a write to P048 will cause the output pins corresponding to the locations that were written as 1 to be set in the low state. The output pins corresponding to the locations that were written as 0 remain unchanged.

Refer to the following table and to the example in subsection 12.11.8.

Bit OPx WRITE	OP SET/CLR SELECT	Result
1	1	PACT OPx STATE = 1
1	0	PACT OPx STATE = 0
0	x	PACT OPx STATE remains unchanged

Bit 1 **EVENT COUNTER SW RESET**. 8-Bit Event Counter Software Reset.

This bit resets the 8-bit event counter. When set, the 8-bit counter is continuously cleared. This bit *must* be cleared to enable the event counter to operate.

0 = Event counter operating.
1 = Event counter cleared.

Bit 2 **CP6 EVENT ONLY**. CP6 8-Bit Event Counter Input Only.

This bit must be cleared to allow 32-bit captures triggered by CP6. This bit does not disable the 16-bit captures on event (CP6) when triggered by a command/definition area command.

0 = CP6 increments event counter and causes 32-bit captures.
1 = CP6 increments event counter only.

Bit 3 – 5 **INPUT CAPT PRESCALE SELECT1–3**. Input Capture Prescale Select 1 – 3.

These bits set the prescaler rate for pins CP3 to CP6. The bits allow a divide rate of +1 to +8. The prescale rate does not affect the event counter.

INPUT CAPT PRESCALE SELECT			Divide Rate
3	2	1	
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

Bit 6 **BUFFER HALF/FULL INT FLAG**. Buffer Half/Full Interrupt Flag.

This bit is set when the circular buffer becomes half or completely full. It is cleared when a zero is written to this bit or during RESET.

0 = Interrupt inactive.
1 = Interrupt pending.

Bit 7 **BUFFER HALF/FULL INT ENA**. Buffer Half/Full Interrupt Enable.

This bit determines whether or not the circular buffer can generate an interrupt on the half full and full buffer boundaries.

0 = Disables interrupt.
1 = Enables interrupt.

12.11.14 Global Function Control Register (PACTPRI)

The PACTPRI register controls the watchdog time-out rate, the PACT interrupt priority levels, and the PACT operating mode.

Global Function Control Register (PACTPRI)
[Memory Address – 104Fh]

Bit #	7	6	5	4	3	2	1	0
P04F	PACT STEST	—	PACT GROUP 1 PRIORITY	PACT GROUP 2 PRIORITY	PACT GROUP 3 PRIORITY	PACT MODE SELECT	PACT WD PRE-SCALE SELECT1	PACT WD PRE-SCALE SELECT0
	RP–0		RP–0	RP–0	RP–0	RP–0	RP–0	RP–0

R = Read, P = Privileged write only, C = Clear, *–n* = Value of the bit after the register is reset

Bit 0, 1 **PACT WD PRESCALE SELECT0–1**. PACT Watchdog Prescale Select 0 – 1.

These bits select the watchdog time-out rate. You can write to these bits only during privilege mode (after reset).

PACT WD PRESCALE SELECT1	PACT WD PRESCALE SELECT0	Options
0	0	Watchdog reset on bit 9 of default timer
0	1	Watchdog reset on bit 15 of default timer
1	0	Watchdog reset on bit 19 of default timer
1	1	Disable watchdog

Bit 2 **PACT MODE SELECT**. PACT Mode Select.

This bit selects the mode for the PACT module to operate in.

0 = PACT operates in mode A
1 = PACT operates in mode B

Bit 3 **PACT GROUP 3 PRIORITY**. PACT Group 3 Priority Select.

This bit assigns the interrupt priority level of the PACT group 3 interrupt vectors.

0 = PACT group 3 interrupts are level 1 (high-priority) requests.
1 = PACT group 3 interrupts are level 2 (low-priority) requests.

Bit 4 **PACT GROUP 2 PRIORITY**. PACT Group 2 Priority Select.

This bit assigns the interrupt priority level of the PACT group 2 interrupt vectors.

0 = PACT group 2 interrupts are level 1 (high-priority) requests.
1 = PACT group 2 interrupts are level 2 (low-priority) requests.

Bit 5 **PACT GROUP 1 PRIORITY**. PACT Group 1 Priority Select.

This bit assigns the interrupt priority level of the PACT group 1 interrupt vectors.

0 = PACT group 1 interrupts are level 1 (high-priority) requests.
1 = PACT group 1 interrupts are level 2 (low-priority) requests.

Bit 6 **Reserved.** Read data is indeterminate.

Bit 7 **PACT STEST**.

This bit must be cleared to ensure proper operation.

Chapter 13

Assembly Language Instruction Set

An assembly language instruction set is a symbolic language that presents binary machine code in a more readable form. The TMS370 family is supported by a 73-function instruction set that uses a wide variety of addressing modes.

This chapter includes the following topics:

Topic **Page**

13.1 Instruction Operation

The assembly language instruction set provides a convenient method of programming the CPU. Each TMS370 assembly language instruction converts directly to one machine operation and consists of these elements:

- **A function mnemonic.** The mnemonic specifies the type of CPU operation.
- **Zero to three operands.** The operands indicate where the CPU can find or store data during an instruction execution. The type and combination of operands determine the actual opcode(s) for an instruction. The MOV instruction, for example, has 27 different options, each with its own opcode.

A typical two-operand instruction is shown below:

MNEMONIC	SOURCE	DESTINATION
ADD	#9,	R3

The example above can be read like this: add the value 9 to the contents of register number 3 and place the result back into register number 3. The destination serves as a second source as well as the final address of the result; moreover, registers can be directly manipulated without having to use intermediate registers. Note that this instruction form differs from the mnemonic-destination-source arrangement that some microprocessors use.

The following example shows how the instruction above might appear in a complete program line.

LABEL	INSTRUCTION	OPERANDS	COMMENT
XXXXX	ADD	#9,R3	;comment

There should be at least one space between each entry type. The label and comment entries are optional.

The 73 instructions are supported by 246 opcodes that provide flexible control of CPU program flow. Some instructions such as CLRC and TEST A share the same opcode to help you understand all of the functions of an opcode. Some instructions use 16-bit opcodes, depending on the type of instruction and/or the addressing mode used. The assembler constructs several bit manipulation instructions from other instructions in order to simplify writing and enhance the readability of the program.

13.2 Symbol Definitions

In order to understand the instructions described in this chapter, you must know what the symbols in the syntax descriptions represent. Table 13–1 lists the instruction set symbols.

Table 13–1. TMS370 Symbols Defined

Symbol	Definition	Symbol	Definition
A	Register A or R0 in register file	off16	16-bit signed offset
B	Register B or R1 in register file	Rd	Destination register in register file
C	Carry flag	Rn	Register n of register file
cnd	Condition	Rname	Symbol-defined register bit
d/D	Destination operand (8-bit/16-bit)	Rp	Register pair
iop8	8-bit immediate operand	Rpd	Destination register pair
iop16	16-bit immediate operand	Rps	Source register pair
label	16-bit label	Rs	Source register in register file
LSB	Least significant bit	s	Source operand
LSbyte	Least significant byte	SP	Stack pointer
MSB	Most significant bit	ST	Status register
MSbyte	Most significant byte	V	Overflow/borrow flag
N	Sign flag	XADDR	16-bit address
name	Symbol-defined for a bit	Z	Zero flag
PC	Program counter	(x)	Contents of memory at address x
PCN	16-bit address of next instruction	((x))	Contents of memory location designated by contents at address x
Pd	Destination register in peripheral file ($0 \leq d \leq 255$)	< >	Indicates an entry that must be typed in. For example, <label> indicates that a label must be entered. The brackets themselves are not entered.
Pn	Register n of peripheral file ($0 \leq n \leq 255$)	→	Is assigned to
Pname	Symbol-defined peripheral bit	←	Becomes equal to
Ps	Source register in peripheral file ($0 \leq s \leq 255$)	#	Immediate operand
off8	8-bit signed offset	@	Indirect addressing operand

13.3 Addressing Modes

Each TMS370 assembly language instruction includes from zero to three operands. Each operand has an addressing mode. The addressing mode specifies how the CPU calculates the address of the data needed by the instruction. The power of the TMS370 is enhanced by the large number of addressing modes available.

The 14 addressing modes are divided into two classes:

- **General,** which uses an 8-bit addressing range
- **Extended,** which uses a 16-bit addressing range

Table 13–2 shows the 14 addressing modes, each with a sample instruction and its execution. The subsections that follow describe these modes.

Table 13–2. Overview of Addressing Modes

Addressing Mode	Example	Execution
General:		
Implied	LDSP	(B) → (SP)
Register	MOV R5,R4	(0005) → (0004)
Peripheral	MOV P025,A	(1025) → A
Immediate	ADD #123,R3	123 + (03) → (03)
PC Relative	JMP offset	PCN + offset → (PC)
Stack Pointer Relative	MOV 2(SP),A	(2 + (SP)) → (A)
Extended:		
Absolute Direct	MOV A,1234	(A) → (1234)
Absolute Indexed	MOV 1234(B),A	(1234 + (B)) → (A)
Absolute Indirect	MOV @R4,A	((R3:R4)) → (A)
Absolute Offset Indirect	MOV 12(R4),A	(12 + (R3:R4)) → (A)
Relative Direct	JMPL 1234	PCN + 1234 → (PC)
Relative Indexed	JMPL 1234(B)	PCN + 1234 + (B) → (PC)
Relative Indirect	JMPL @R4	PCN + (R3:R4) → (PC)
Relative Offset Indirect	JMPL 12(R4)	PCN + 12 + (R3:R4) → (PC)

A number of instructions use more than one addressing mode, and several instructions, such as MOV, are very versatile.

13.3.1 General Addressing Modes

Instructions using the **general addressing** modes have an 8-bit range of operation and deal with the register file, peripheral file, or a nearby destination. The general addressing modes are:

- Implied
- Register
- Peripheral
- Immediate
- Program counter relative
- Stack pointer relative

Most of these modes can use *any* register as a source and/or destination, preventing the bottleneck found on other microprocessors that use only one or two registers.

13.3.1.1 Implied Addressing Mode

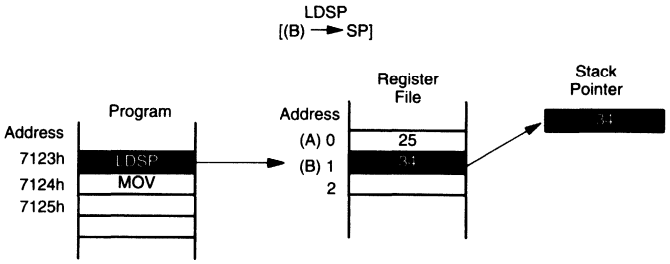
In the **implied addressing** mode, the instruction type alone determines where the data is to be found. You do not have to specify the operands, because they are inherently specified in the instruction.

For example, the LDSP (load stack pointer) instruction always copies the contents of register B to the stack pointer (SP). Neither the source nor destination is explicitly stated; they are implied in the instruction itself. These are the instructions that use the implied addressing mode:

CLRC	Clear the carry bit	LDSP	Load stack pointer
RTS	Return from subroutine	RTI	Return from interrupt
SETC	Set carry	STSP	Store stack pointer
EINT	Enable interrupts	EINTH	Enable high-level interrupts
EINTL	Enable low-level interrupts		

Figure 13–1 shows an example of the implied addressing mode.

Figure 13–1. Implied Addressing Mode



13.3.1.2 Register Addressing Mode

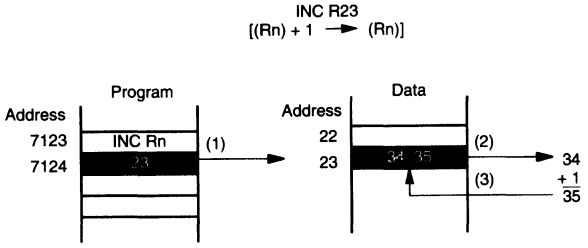
The register file (RF) of the TMS370 consists of the first 128 or 256 bytes of memory (the number of bytes differs according to the device that you are using). In the **register addressing** mode, instructions use a one-byte value to specify an address (location) in the RF. Any location in the RF can be accessed in one memory cycle by instructions using this mode. Extended addressing modes require two cycles to access the register file.

In register file addressing, the operand is stated by Rn, where n is the 8-bit address number. The address number can be a decimal (0–255) or hexadecimal (0–0FF) number. Hexadecimal numbers require a leading zero, but no suffix. Registers R0 and R1 of the register file are also known as registers A and B and are referenced as such by most instructions to reduce the size of the program. For example, the instruction `MOV A,B` uses one byte of code, while the instruction `MOV R3,R4` uses three bytes of code. Any register can be specified by a symbol that has been equated to that register. This is illustrated in the following example:

```
        MOV   R16,R011  ;Move contents of 0010h to 0011h
CAT     .EQU  R16       ;Equate register 16 to symbol CAT
DOG     .EQU  R17       ;Equate register 17 to symbol DOG
        MOV   CAT,DOG   ;Move contents of 0010h to 0011h
```

Note that the entry .EQU is an assembler directive, not an assembly language instruction. For more information on assembler directives, refer to the *TMS370 Family Assembly Language Tools User's Guide*. Figure 13–2 shows an example of the register addressing mode.

Figure 13–2. Register Addressing Mode



Note: Numbers in parentheses represent order of execution.

13

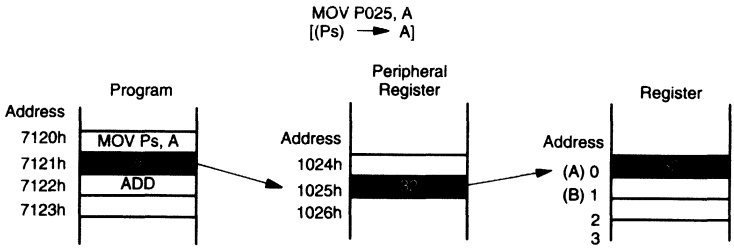
13.3.1.3 Peripheral Addressing Mode

The **peripheral addressing** mode is used for program control of the peripheral on-chip modules such as timers, interrupts, and I/O ports. Devices with bus expansion can address a small amount of external memory as peripheral file (PF) space. The PF of the TMS370 is allocated 256 bytes of memory. Each PF register is accessed by an 8-bit operand designated as Pn, with n being either a decimal (0–255) or hexadecimal (0–0FF) number. Hexadecimal numbers require a leading zero but no suffix. The CPU assumes the most significant byte of a peripheral address to be 010h. As described for register file addressing in subsection 13.3.1.2, the Pn designation, like the Rn designation, can be substituted by a symbol by using the equate (.EQU) assembler directive as shown in the example below.

```
        MOV   R16,P020 ;Move contents of 0010h to 1020h
CAT     .EQU  R16      ;Equate register 16 to symbol CAT
DOG     .EQU  P32      ;Equate peripheral file 32 to symbol DOG
        MOV   CAT,DOG  ;Move contents of 0010h to 1020h
```

The use of designated symbols is optional but is particularly suited for the register and peripheral addressing modes. Figure 13–3 shows an example of peripheral file addressing.

Figure 13–3. Peripheral Addressing Mode



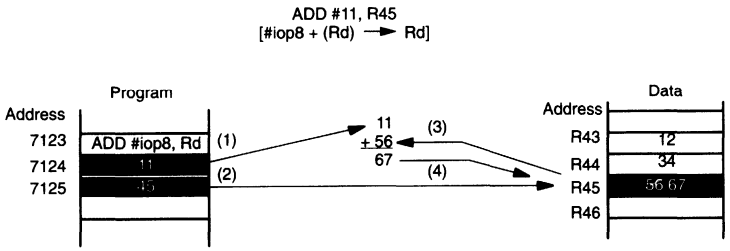
13.3.1.4 Immediate Addressing Mode

The **immediate addressing** mode uses a constant value as the operand that immediately follows the function mnemonic. This mode allows nonchanging data to be incorporated into the instruction. The constant can be in the form of a decimal number, a hexadecimal number, or a symbolic label, but the constant is always preceded by the number sign (#). Note that hexadecimal numbers require *both* a leading numeric digit *and* the h suffix. Some examples of immediate addressing are as follows:

```
     MOV   #0Fh,A        ;Store the value 15 in register A
     MOV   #(3*54),R022  ;Store the value 162 at location 022h
CNT  .EQU  12            ;Equate 12 to symbol CNT
     ADD   #CNT,R34      ;Add the value 12 to register 34, place
                         ;result in register 34.
```

Figure 13–4 illustrates an instruction using the immediate addressing mode.

Figure 13–4. Immediate Addressing Mode



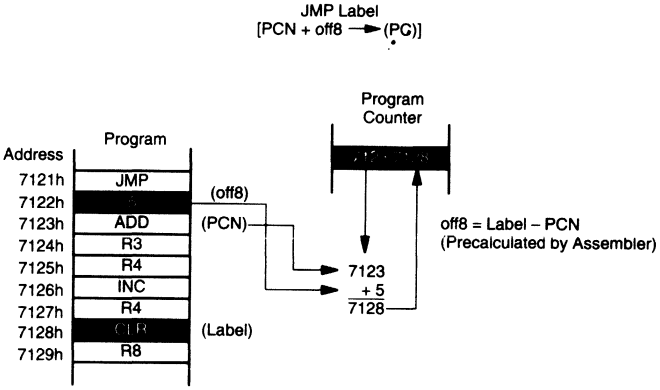
Note: Numbers in parentheses represent order of execution.

13.3.1.5 Program Counter Relative Addressing Mode

The **program counter relative addressing** mode adds an 8-bit signed offset to the address of the next instruction to produce the address of the succeeding instruction. The new address is placed in the program counter register. The range of the 8-bit offset is within 128 bytes before or 127 bytes after the instruction following the jump. When labels are used, the signed offset is automatically calculated by the assembler. Recall that the PCN is the location (address) of the next instruction.

Figure 13–5 illustrates object code generated by a jump instruction using the program counter relative addressing mode.

Figure 13–5. Program Counter Relative Addressing Mode



13.3.1.6 Stack Pointer Relative Addressing Mode

The **stack pointer relative addressing** mode adds an 8-bit signed constant to the existing 8-bit contents of the stack pointer register. The result is truncated to an 8-bit address of the data. The second operand in the stack pointer relative mode is always register A.

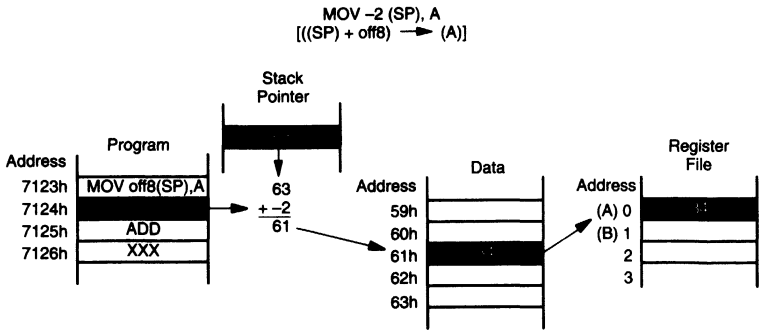
This addressing mode is useful in accessing arguments that are passed to a subroutine on the stack. You must insure that the resulting address location is within the implemented register file, because overflows or underflows will execute without warning.

Only the CMP and MOV instructions use this mode. An example of stack relative addressing is as follows:

```
MOV -2(SP), A
```

In this example, the value of –2 plus the stack pointer equals the address of the data to be moved to register A. Figure 13–6 illustrates this instruction operation.

Figure 13–6. Stack Pointer Relative Addressing Mode



13.3.2 Extended Addressing Modes

The **extended addressing** modes provide a sophisticated method for stepping through tables, picking out array values, and generating addresses. These modes allow the program to access data from anywhere in the memory. Extended addressing modes consist of four main types:

- Direct
- Indirect
- Indexed
- Offset indirect

Each of these four types can be subdivided into absolute and relative modes for a total of eight extended addressing modes.

- **Extended absolute addressing** modes always use register A or the PC (program counter) as one of the operands in generating a 16-bit address. The extended absolute addressing modes are used only by the branch (BR), CALL, compare (CMP), and move (MOV) instructions. The BR and CALL instructions use these modes exclusively.

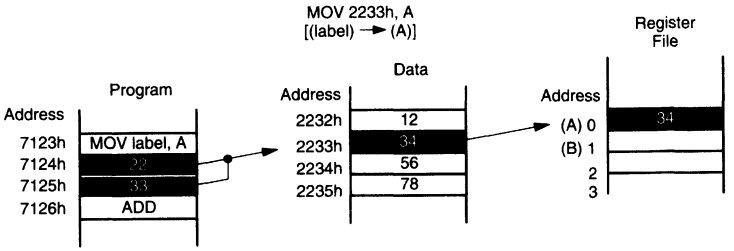
- The **extended relative addressing** modes are similar to the extended absolute addressing modes but include the additional step of combining the operand with the PCN value before placing the 16-bit address into the program counter. These modes are similar to the program counter relative mode. A 16-bit signed offset is used to calculate the succeeding instruction address. The succeeding instruction address is calculated at execution time by using the signed 16-bit offset according to the instruction's addressing mode.

  The extended relative addressing modes are useful in relocatable code because operation is based on the differences in address position instead of on the addresses themselves. This makes the extended relative addressing modes well-suited for high-level languages that often use position-independent code. Extended relative addressing is used by the CALLR and JMPL instructions.

13.3.2.1 Direct Addressing Modes

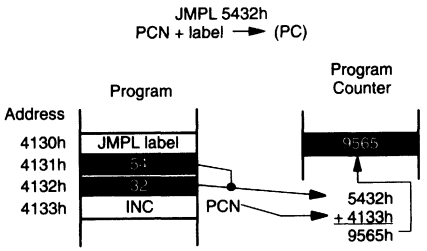
Direct addressing mode instructions use an address as the operand. The 16-bit address is written as either a constant value or a label and immediately follows the opcode in the source code. The **absolute direct addressing** mode acts upon the address itself as shown in Figure 13–7.

Figure 13–7. Absolute Direct Addressing Mode



The **relative direct addressing** mode (Figure 13–8) adds the address of the next instruction to the 16-bit operand to produce the address of the succeeding instruction. If a label is used in the instruction, the assembler automatically calculates the offset to use as the operand.

Figure 13–8. Relative Direct Addressing Mode



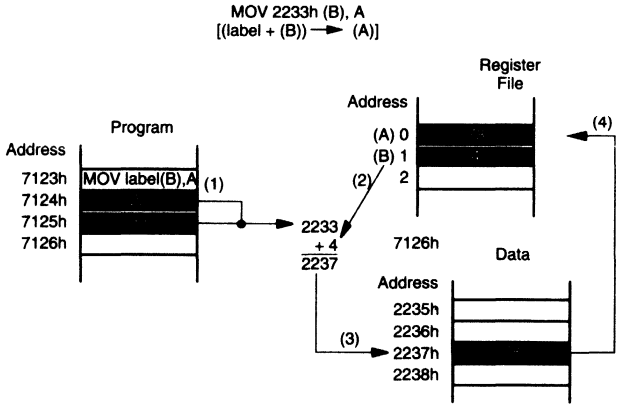
13.3.2.2 Indexed Addressing Modes

The **absolute indexed addressing** mode generates a 16-bit address by adding the unsigned contents of register B to a 16-bit unsigned constant. The assembly language statement for the indexed addressing modes contains the direct memory address written as a 16-bit value or a label, followed by a B in parentheses: MOV 1234(B), or MOV LABEL(B).

- The MOV and CMP instructions can use absolute indexed addressing to step easily through a small table or to pick out a particular array value.
- The CALL and BR instructions can use this mode to execute code according to a decision table and the value in register B.

Figure 13–9 illustrates how the object code produced by an instruction using this mode generates a 16-bit effective address.

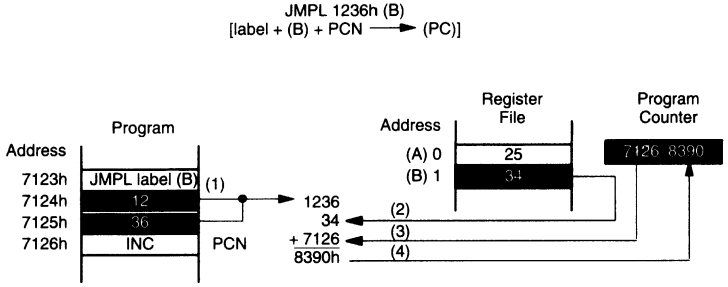
Figure 13–9. Absolute Indexed Addressing Mode



Note: Numbers in parentheses represent order of execution.

The **relative indexed addressing** mode includes the operation described for the absolute indexed addressing mode with the following additional step: the address of the next instruction is added to the sum of register B and the signed 16-bit constant offset to produce the address of the next instruction. The relative indexed addressing mode is shown in Figure 13–10.

Figure 13–10. Relative Indexed Addressing Mode



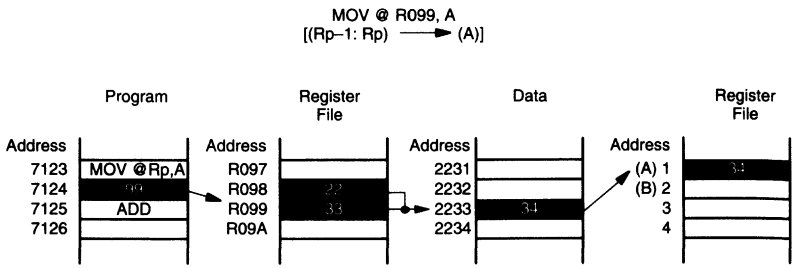
Note: Numbers in parentheses represent order of execution.

13.3.2.3 Indirect Addressing Modes

In **indirect addressing** modes, instructions use the contents of a register pair as the 16-bit address of the data. The indirect register file address is written as a register number (Rn) preceded by the commercial at (@) symbol. The LSbyte of the address is contained in Rn, and the MSbyte of the address is contained in the previous register (Rn–1). The TMS370 can use any register pair as an indirect register.

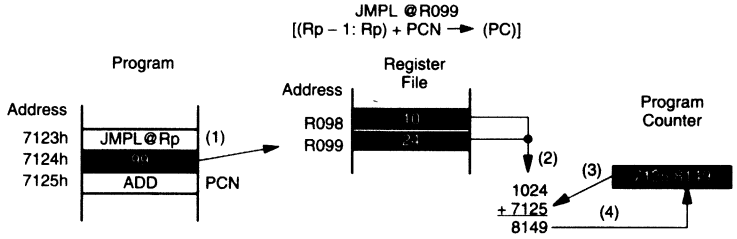
Figure 13–11 shows how the **absolute indirect addressing** mode uses the register pair in the calculation.

Figure 13–11. Absolute Indirect Addressing Mode



The **relative indirect addressing** mode (Figure 13–12) adds the address of the next instruction to the register pair contents before obtaining the destination address.

Figure 13–12. Relative Indirect Addressing Mode

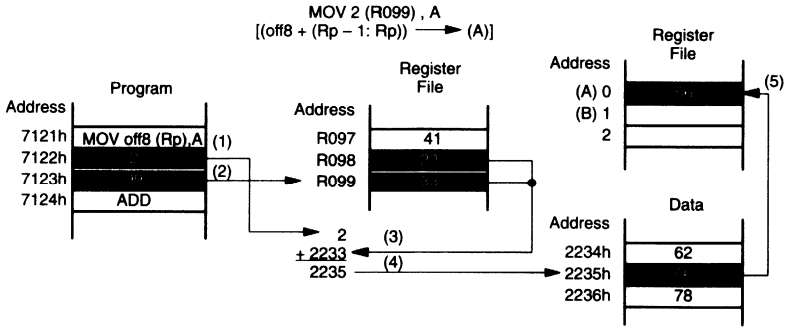


Note: Numbers in parentheses represent order of execution.

13.3.2.4 Offset Indirect Addressing Modes

The **offset indirect addressing** modes are similar to the indirect addressing modes previously described. The **absolute offset indirect addressing** mode generates a 16-bit address by adding an 8-bit signed offset to an address taken from a register pair. Offset indirect addressing is useful for stepping through tables or for finding a particular value in a table by using two values to generate the address. Figure 13–13 illustrates how the object code produced by an instruction using the offset indirect addressing mode generates a 16-bit effective address.

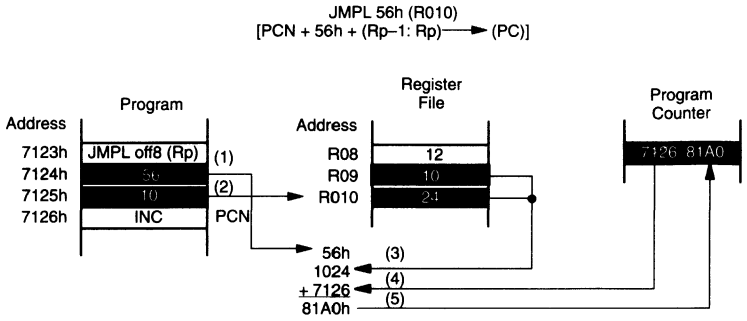
Figure 13–13. Absolute Offset Indirect Addressing Mode



Note: Numbers in parentheses represent order of execution.

The **relative offset indirect addressing** mode adds the address of the next instruction with the sum of the 8-bit signed offset and the register pair before obtaining the destination address.

Figure 13–14. Relative Offset Indirect Addressing Mode



Note: Numbers in parentheses represent order of execution.

13.3.3 Additional Addressing Modes

In some cases, the operation of an instruction does not fit into any of the addressing modes previously described. Some modes illustrated by instructions such as MOVW #iop(B),Rpd provide unique capabilities for table addressing. Other modes illustrated by instructions like the LDST #iop8 give access to the status register bits (shown in Figure 13–15). The individual instruction description can be referenced for a list of that instruction's operations.

13.3.4 Status Register

Most of the instructions affect the bits in the status register. The status register is presented in Figure 13–15 as a quick reference to aid in programming.

Figure 13–15. Status Register (ST)

Status Register (ST)

Bit #	7	6	5	4	3	2	1	0
P04F	C	N	Z	V	IE2	IE1	—	—
	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0		

R = Read, W = Write, –*n* = Value of the bit after the register is reset

13.4 Instruction Set Overview

The tables in this section list the instruction set, including pertinent characteristics and an opcode/instruction map.

Table 13–3 lists all instruction formats, opcodes, byte lengths, cycles/instruction, operands, status bits affected, and an operational description.

Table 13–3. TMS370 Family Instruction Overview

Mnemonic		Opcode	Bytes	Cycles t_c	Status C N Z V	Operation Description
ADC	B,A	69	1	8	x x x x	(s) + (d) + (C) → (d) Add the source, destination, and carry bit together. Store at the destination address.
	Rs,A	19	2	7		
	Rs,B	39	2	7		
	Rs,Rd	49	3	9		
	#iop8,A	29	2	6		
	#iop8,B	59	2	6		
	#iop8,Rd	79	3	8		
ADD	B,A	68	1	8	x x x x	(s) + (d) → (d) Add the source and destination operands at the destination address.
	Rs,A	18	2	7		
	Rs,B	38	2	7		
	Rs,Rd	48	3	9		
	#iop8,A	28	2	6		
	#iop8,B	58	2	6		
	#iop8,Rd	78	3	8		
AND	A,Pd	83	2	9	0 x x 0	(s) AND (d) → (d) AND the source and destination operands together and store at the destination address.
	B,A	63	1	8		
	B,Pd	93	2	9		
	Rs,A	13	2	7		
	Rs,B	33	2	7		
	Rs,Rd	43	3	9		
	#iop8,A	23	2	6		
	#iop8,B	53	2	6		
	#iop8,Rd	73	3	8		
	#iop8,Pd	A3	3	10		
BR	label	8C	3	9	– – – –	XADDR → (PC) Branch to the destination address.
	@Rp	9C	2	8		
	label(B)	AC	3	11		
	off8(Rp)	F4 EC	4	16		
BTJO†	A,Pd,off8	86	3	10	0 x x 0	If (s) AND (d) ≠ 0, then PCN + offset → (PC). If the AND of the source and destination operands ≠ 0 (corresponding 1 bits), the PC will add the offset, and the jump will be taken.
	B,A,off8	66	2	10		
	B,Pd,off8	96	3	10		
	Rs,A,off8	16	3	9		
	Rs,B,off8	36	3	9		
	Rs,Rd,off8	46	4	11		
	#iop8,A,off8	26	3	8		
	#iop8,B,off8	56	3	8		
	#iop8,Rd,off8	76	4	10		
	#iop8,Pd,off8	A6	4	11		

† Add two to the cycle count if a jump is taken.

Note: Legend :

0 Status bit always cleared.
1 Status bit always set.
x Status bit cleared or set on results.
– Status bit not affected.

Table 13–3. TMS370 Family Instruction Overview (Continued)

Mnemonic		Opcode	Bytes	Cycles t_C	Status C N Z V	Operation Description
BTJZ†	A,Pd,off8	87	3	10	0 x x 0	If (s) AND (not d) ≠ 0, then (PCN) + offset → (PC). If any 1 in the source corresponds to a 0 in the destination, the PC adds the offset, and the jump is taken.
	B,A,off8	67	2	10		
	B,Pd,off8	97	3	10		
	Rs,A,off8	17	3	9		
	Rs,B,off8	37	3	9		
	Rs,Rd,off8	47	4	11		
	#iop8,A,off8	27	3	8		
	#iop8,B,off8	57	3	8		
	#iop8,Rd,off8	77	4	10		
	#iop8,Pd,off8	A7	4	11		
CALL	label	8E	3	13	– – – –	Push PC MSbyte, PC LSbyte, XADDR → (PC)
	@Rp	9E	2	12		
	label(B)	AE	3	15		
	off8(Rp)	F4 EE	4	20		
CALLR	label	8F	3	15	– – – –	Call relative Push PC MSbyte, PC LSbyte, PCN + (XADDR) → (PC)
	@Rp	9F	2	14		
	label(B)	AF	3	17		
	off8(Rp)	F4 EF	4	22		
CLR	A	B5	1	8	0 0 1 0	0 → (Rd) Clear the destination operand.
	B	C5	1	8		
	Rd	D5	2	6		
CLRC		B0	1	9	0 x x 0	0 → (C) Clear the carry bit. N and Z bits are set on the result of A.
CMP	label,A	8D	3	11	x x x x	Compare; (d) – (s) computed. Set flags on the result of the source operand subtracted from the destination operand. Operands are not affected by operation.
	@Rp,A	9D	2	10		
	label(B),A	AD	3	13		
	off8(Rp),A	F4 ED	4	18		
	off8(SP),A	F3	2	8		
	B,A	6D	1	8		
	Rs,A	1D	2	7		
	Rs,B	3D	2	7		
	Rs,Rd	4D	3	9		
	#iop8,A	2D	2	6		
	#iop8,B	5D	2	6		
	#iop8,Rd	7D	3	8		
CMPBIT	Rname	75	3	8	0 x x 0	Complement bit; invert the bit
	Pname	A5	3	10		
COMPL	A	BB	1	8	x x x 0	2s complement; 00h – (s) → (d)
	B	CB	1	8		
	Rn	DB	2	6		

† Add two to the cycle count if a jump is taken.

Note: Legend :

0 Status bit always cleared.

1 Status bit always set.

x Status bit cleared or set on results.

– Status bit not affected.

Table 13–3. TMS370 Family Instruction Overview (Continued)

Mnemonic		Opcode	Bytes	Cycles t_c	Status C N Z V	Operation Description
DAC	B,A	6E	1	10	x x x x	(s) + (d) + (C) → (d) (BCD) The source, destination, and carry bit are added, and the BCD sum is stored at the destination address.
	Rs,A	1E	2	9		
	Rs,B	3E	2	9		
	Rs,Rd	4E	3	11		
	#iop8,A	2E	2	8		
	#iop8,B	5E	2	8		
	#iop8,Rd	7E	3	10		
DEC	A	B2	1	8	x x x x	(d) – 1 → (d) Decrement destination operand by 1.
	B	C2	1	8		
	Rn	D2	2	6		
DINT		F0 00	2	6	0 0 0 0	0 → (ST)(global interrupt enable bits) 0 → IE1, 0 → IE2.
DIV	Rs,A	F4 F8	3	55–63 14	0 x x 0 1 1 1 1	A:B/Rs → A(= quo),B(= REM) Integer divide, 16 by 8 bits. Overflow detected.
DJNZ†	A,off8	BA	2	10	– – – –	(d) – 1 → (d); If (d) ≠ 0, then PCN + offset → (PC). Decrement and jump if not 0.
	B,off8	CA	2	10		
	Rn,off8	DA	3	8		
DSB	B,A	6F	1	10	x x x x	(d) – (s) – 1 + (C) → (d) (BCD) The source operand is subtracted from the destination; this sum is then reduced by 1, and the carry bit is then added to it. The result is stored as a BCD number.
	Rs,A	1F	2	9		
	Rs,B	3F	2	9		
	Rs,Rd	4F	3	11		
	#iop8,A	2F	2	8		
	#iop8,B	5F	2	8		
	#iop8,Rd	7F	3	10		
EINT		F0 0C	2	6	0 0 0 0	0Ch → (ST)(global interrupt enable bit) 1 → IE1, 1 → IE2.
EINTH		F0 04	2	6	0 0 0 0	04h → (ST)(high-priority global interrupt enable bit) 1 → IE1, 0 → IE2
EINTL		F0 08	2	6	0 0 0 0	08h → (ST)(low-priority global interrupt enable bit) 0 → IE1, 1 → IE2
IDLE		F6	1	6	– – – –	(PC) → (PC) until interrupt (PC) + 1 → (PC) after return from interrupt. Stops µC execution until an interrupt. Entry to low-power modes.
INC	A	B3	1	8	x x x x	(d) + 1 → (d) Increase the destination operand by 1.
	B	C3	1	8		
	Rn	D3	2	6		
INCW	#iop8,Rp	70	3	11	x x x x	(Rp) + operand→ (Rp) Add 8-bit immediate operand to register pair.

† Add two to the cycle count if a jump is taken.

Note: Legend :

0 Status bit always cleared.
1 Status bit always set.
x Status bit cleared or set on results.
– Status bit not affected.

Table 13–3. TMS370 Family Instruction Overview (Continued)

Mnemonic		Opcode	Bytes	Cycles t_c	Status C N Z V	Operation Description
INV	A	B4	1	8	0 x x 0	NOT(d) → (d)
	B	C4	1	8		1s complement the destination operand.
	Rn	D4	2	6		
JBIT0†	Rname,off8	77	4	10	0 x x 0	Jump if bit = 0
	Pname,off8	A7	4	11		
JBIT1†	Rname,off8	76	4	10	0 x x 0	Jump if bit = 1
	Pname,off8	A6	4	11		
JMP	off8	00	2	7	– – – –	PCN + off8 → (PC)
						Jump unconditionally using an 8-bit offset.
JMPL	label	89	3	9	– – – –	PCN + D → (PC)
	@Rp	99	2	8		Jump unconditionally using a 16-bit offset.
	label(B)	A9	3	11		
	off8(Rp)	F4 E9	4	16		
Jcnd†					– – – –	Conditional jump
	JC	03	2	5		Carry
	JEQ	02	2	5		Jump equal
	JG	0E	2	5		Greater than, signed
	JGE	0D	2	5		Greater than or equal, signed
	JHS	0B	2	5		Higher or same, unsigned
	JL	09	2	5		Less than, signed
	JLE	0A	2	5		Less than or equal, signed
	JLO	0F	2	5		Lower value, unsigned
	JN	01	2	5		Negative, signed
	JNC	07	2	5		No carry
	JNE	06	2	5		Jump not equal
	JNV	0C	2	5		No overflow, signed
	JNZ	06	2	5		Not zero
	JP	04	2	5		Positive, signed
	JPZ	05	2	5		Positive or zero, signed
	JV	08	2	5		Overflow, signed
	JZ	02	2	5		Zero
LDSP		FD	1	7	– – – –	(B) → (SP)
						Load stack pointer with contents of register B.
LDST	#iop8	F0	2	6	x x x x	(s) → (ST)
						Load ST register.

† Add two to the cycle count if a jump is taken.

Note: Legend :

0 Status bit always cleared.
1 Status bit always set.
x Status bit cleared or set on results.
– Status bit not affected.

13

Table 13–3. TMS370 Family Instruction Overview (Continued)

Mnemonic		Opcode	Bytes	Cycles t_c	Status C N Z V	Operation Description
MOV	A,B	C0	1	9	0 x x 0	(s) → (d) Replace the destination operand with the source operand.
	A,Rd	D0	2	7		
	A,Pd	21	2	8		
	A,label	8B	3	10		
	A,@Rp	9B	2	9		
	A,label(B)	AB	3	12		
	A,off8(Rp)	F4 EB	4	17		
	A,off8(SP)	F2	2	7		
	Rs,A	12	2	7		
	Rs,B	32	2	7		
	label,A	8A	3	10		
	@Rp,A	9A	2	9		
	label(B),A	AA	3	12		
	off8(Rp),A	F4 EA	4	17		
	off8(SP),A	F1	2	7		
	B,A	62	1	8		
	B,Rd	D1	2	7		
	B,Pd	51	2	8		
	Rs,Rd	42	3	9		
	Rs,Pd	71	3	10		
	Ps,A	80	2	8		
	Ps,B	91	2	8		
	Ps,Rd	A2	3	10		
	#iop8,A	22	2	6		
	#iop8,B	52	2	6		
	#iop8,Rd	72	3	8		
	#iop8,Pd	F7	3	10		
MOVW	Rps,Rpd	98	3	12	0 x x 0	(s) → (Rpd–1:Rpd) Copy the source register word to the destination register pair.
	#iop16,Rpd	88	4	13		
	#iop16(B),Rpd	A8	4	15		
	#iop8(Rp),Rpd	F4 E8	5	20		
MPY	B,A	6C	1	47	0 x x 0	(s) x (d) → (A:B) Multiply the source and destination operands; store the result in registers A (MSbyte) and B (LSbyte).
	Rs,A	1C	2	46		
	Rs,B	3C	2	46		
	Rs,Rd	4C	3	48		
	#iop8,A	2C	2	45		
	#iop8,B	5C	2	45		
	#iop8,Rd	7C	3	47		
NOP		FF	1	7	– – – –	No operation

Note: Legend:

0 Status bit always cleared.
1 Status bit always set.
x Status bit cleared or set on results.
– Status bit not affected.

Table 13–3. TMS370 Family Instruction Overview (Continued)

Mnemonic		Opcode	Bytes	Cycles t_C	Status C N Z V	Operation Description
OR	A,Pd	84	2	9	0 x x 0	(s) OR (d) → (d)
	B,A	64	1	8		
	B,Pd	94	2	9		
	Rs,A	14	2	7		
	Rs,B	34	2	7		Logically OR the source and destination operands, and store the results at the destination address.
	Rs,Rd	44	3	9		
	#iop8,A	24	2	6		
	#iop8,B	54	2	6		
	#iop8,Rd	74	3	8		
	#iop8,Pd	A4	3	10		
POP	A	B9	1	9	0 x x 0	((SP)) → (d)
	B	C9	1	9		(SP) –1 → (SP)
	Rd	D9	2	7		
	ST	FC	1	8	x x x x	
PUSH	A	B8	1	9	0 x x 0	(SP) + 1 → (SP)
	B	C8	1	9		(s) → ((SP))
	Rs	D8	2	7		Copy the operand onto the stack.
	ST	FB	1	8	– – – –	Copy the status register onto the stack.
RL	A	BE	1	8	x x x 0	Bit(n) → Bit(n + 1)
	B	CE	1	8		Bit(7) → Bit(0) and Carry
	Rn	DE	2	6		
RLC	A	BF	1	8	x x x 0	Bit(n) → Bit(n + 1)
	B	CF	1	8		Carry → Bit(0)
	Rn	DF	2	6		Bit(7) → Carry
RR	A	BC	1	8	x x x 0	Bit(n + 1) → Bit(n)
	B	CC	1	8		Bit(0) → Bit(7) and Carry
	Rn	DC	2	6		
RRC	A	BD	1	8	x x x 0	Bit(n + 1) → Bit(n)
	B	CD	1	8		Carry → Bit(7)
	Rn	DD	2	6		Bit(0) → Carry
RTI		FA	1	12	x x x x	Pop PCL, PCH, POP ST
						Return from interrupt.
RTS		F9	1	9	– – – –	Pop PCL, PCH
SBB	B,A	6B	1	8	x x x x	(d) – (s) – 1 + (C) → (d)
	Rs,A	1B	2	7		Subtract with borrow.
	Rs,B	3B	2	7		Destination minus source minus 1 plus carry;
	Rs,Rd	4B	3	9		stored at the destination address.
	#iop8,A	2B	2	6		
	#iop8,B	5B	2	6		
	#iop8,Rd	7B	3	8		
SBIT0	Rname	73	3	8	0 x x 0	Set bit to 0
	Pname	A3	3	10		
SBIT1	Rname	74	3	8	0 x x 0	Set bit to 1
	Pname	A4	3	10		

13

Note: Legend :

0 Status bit always cleared.
1 Status bit always set.
x Status bit cleared or set on results.
– Status bit not affected.

Table 13–3. TMS370 Family Instruction Overview (Concluded)

Mnemonic		Opcode	Bytes	Cycles t_c	Status C N Z V	Operation Description
SETC		F8	1	7	1 0 1 0	Axh → (ST) Set the carry bit. IE1 and IE2 unchanged.
STSP		FE	1	8	– – – –	(SP) → (B) Copy the SP into register B.
SUB	B,A	6A	1	8	x x x x	(d) – (s) → (d) Store the destination operand minus the source operand into the destination.
	Rs,A	1A	2	7		
	Rs,B	3A	2	7		
	Rs,Rd	4A	3	9		
	#iop8,A	2A	2	6		
	#iop8,B	5A	2	6		
	#iop8,Rd	7A	3	8		
SWAP	A	B7	1	11	0 x x 0	s(7–4,3–0) → d(3–0,7–4) Swap the operand's high and low nibbles.
	B	C7	1	11		
	Rn	D7	2	9		
TRAP	n	EF–E0	1	14	– – – –	Vector n → (PC), n = 0 → 15 Trap to subroutine; Push PCN Trap 0 = EF
TST	A	B0	1	9	0 x x 0	Test; Set flags from register.
	B	C6	1	10		
XCHB	A	B6	1	10	0 x x 0	(B) ←→ (Rd) Swap the contents of register B with (Rd).
	B	C6	1	10		
	Rd	D6	2	8		
XOR	A,Pd	85	2	9	0 x x 0	(s) XOR (d) → (d) Logically exclusive OR the source and destination operands; store at the destination address.
	B,A	65	1	8		
	B,Pd	95	2	9		
	Rs,A	15	2	7		
	Rs,B	35	2	7		
	Rs,Rd	45	3	9		
	#iop8,A	25	2	6		
	#iop8,B	55	2	6		
	#iop8,Rd	75	3	8		
	#iop8,Pd	A5	3	10		

Note: Legend:
0 Status bit always cleared.
1 Status bit always set.
x Status bit cleared or set on results.
– Status bit not affected.

Table 13–4 provides an opcode-to-instruction cross-reference of all 73 instructions and 246 opcodes of the TMS370 instruction set. To check the instruction of a known opcode, locate the left (high) digit across the top or bottom of the table, then find the right (low) digit along the side of the table. The intersection contains the instruction mnemonic, operands, and byte/cycle peculiar to that opcode. Some opcodes, such as B0, are shared by two instructions, in which case, both mnemonics are shown along with the byte/cycle count.

Table 13–4. TMS370 Family Opcode/Instruction Map

LSN \ MSN	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	JMP ra 2/7							INCW #n,Rp 3/11	MOV Ps,A 2/8			CLRC / TST A 1/9	MOV A,B 1/9	MOV A,Rd 2/7	TRAP 15 1/14	LDST n 2/6
1	JN ra 2/5		MOV A,Pd 2/8			MOV B,Pd 2/8		MOV Rs,Pd 3/10		MOV Ps,B 2/7				MOV B,Rd 2/7	TRAP 14 1/14	MOV n(SP),A 2/7
2	JZ ra 2/5	MOV Rs,A 2/7	MOV #n,A 2/6	MOV Rs,B 2/7	MOV Rs,Rd 3/9	MOV #n,B 2/6	MOV B,A 1/8	MOV #n,Rd 3/8			MOV Ps,Rd 3/10	DEC A 1/8	DEC B 1/8	DEC Rn 2/6	TRAP 13 1/14	MOV A,n(SP) 2/7
3	JC ra 2/5	AND Rs,A 2/7	AND #n,A 2/6	AND Rs,B 2/7	AND Rs,Rd 3/9	AND #n,B 2/6	AND B,A 1/8	AND #n,Rd 3/8	AND A,Pd 2/9	AND B,Pd 2/9	AND #n,Pd 3/10	INC A 1/8	INC B 1/8	INC Rn 2/6	TRAP 12 1/14	CMP n(SP),A 2/8
4	JP ra 2/5	OR Rs,A 2/7	OR #n,A 2/6	OR Rs,B 2/7	OR Rs,Rd 3/9	OR #n,B 2/6	OR B,A 1/8	OR #n,Rd 3/8	OR A,Pd 2/9	OR B,Pd 2/9	OR #n,Pd 3/10	INV A 1/8	INV B 1/8	INV Rn 2/6	TRAP 11 1/14	extend inst,2 opcodes
5	JPZ ra 2/5	XOR Rs,A 2/7	XOR #n,A 2/6	XOR Rs,B 2/7	XOR Rs,Rd 3/9	XOR #n,B 2/6	XOR B,A 1/8	XOR #n,Rd 3/8	XOR A,Pd 2/9	XOR B,Pd 2/9	XOR #n,Pd 3/10	CLR A 1/8	CLR B 1/8	CLR Rd 2/6	TRAP 10 1/14	
6	JNZ ra 2/5	BTJO Rs,A,ra 3/9	BTJO #n,A,ra 3/8	BTJO Rs,B,ra 3/9	BTJO Rs,Rd,ra 4/11	BTJO #n,B,ra 3/8	BTJO B,A,ra 2/10	BTJO #n,Rd,ra 4/10	BTJO A,Pd,ra 3/11	BTJO B,Pd,ra 3/10	BTJO #n,Pd,ra 4/11	XCHB A 1/10	XCHB A / TST B 1/10	XCHB Rd 2/8	TRAP 9 1/14	IDLE 1/6
7	JNC ra 2/5	BTJZ Rs,A,ra 3/9	BTJZ #n,A,ra 3/8	BTJZ Rs,B,ra 3/9	BTJZ Rs,Rd,ra 4/11	BTJZ #n,B,ra 3/8	BTJZ B,A,ra 2/10	BTJZ #n,Rd,ra 4/10	BTJZ A,Pd,ra 3/10	BTJZ B,Pd,ra 3/10	BTJZ #n,Pd,ra 4/11	SWAP A 1/11	SWAP B 1/11	SWAP Rn 2/9	TRAP 8 1/14	MOV #n,Pd 3/10
8	JV ra 2/5	ADD Rs,A 2/7	ADD #n,A 2/6	ADD Rs,B 2/7	ADD Rs,Rd 3/9	ADD #n,B 2/6	ADD B,A 1/8	ADD #n,Rd 3/8	MOVW #16,Rpd 4/13	MOVW Rps,Rpd 3/12	MOVW #16(B),Rpd 4/15	PUSH A 1/9	PUSH B 1/9	PUSH Rs 2/7	TRAP 7 1/14	SETC 1/7
9	JL ra 2/5	ADC Rs,A 2/7	ADC #n,A 2/6	ADC Rs,B 2/7	ADC Rs,Rd 3/9	ADC #n,B 2/6	ADC B,A 1/8	ADC #n,Rd 3/8	JMPL lab 3/9	JMPL @Rp 2/8	JMPL lab(B) 3/11	POP A 1/9	POP B 1/9	POP Rd 2/7	TRAP 6 1/14	RTS 1/9
A	JLE ra 2/5	SUB Rs,A 2/7	SUB #n,A 2/6	SUB Rs,B 2/7	SUB Rs,Rd 3/9	SUB #n,B 2/6	SUB B,A 1/8	SUB #n,Rd 3/8	MOV lab,A 3/10	MOV @Rp,A 2/9	MOV lab(B),A 3/12	DJNZ A,ra 2/10	DJNZ B,ra 2/10	DJNZ Rn,ra 3/8	TRAP 5 1/14	RTI 1/12
B	JHS ra 2/5	SBB Rs,A 2/7	SBB #n,A 2/6	SBB Rs,B 2/7	SBB Rs,Rd 3/9	SBB #n,B 2/6	SBB B,A 1/8	SBB #n,Rd 3/8	MOV A,lab 3/10	MOV A,@Rp 2/9	MOV A,lab(B) 3/12	COMPL A 1/8	COMPL B 1/8	COMPL Rn 2/6	TRAP 4 1/14	PUSH ST 1/8

Table 13–4. TMS370 Family Opcode/Instruction Map (Concluded)

LSN \ MSN	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C	JNV ra 2/5	MPY Rs,A 2/46	MPY #n,A 2/45	MPY Rs,B 2/46	MPY Rs,Rd 3/48	MPY #n,B 2/45	MPY B,A 1/47	MPY #n,Rd 3/47	BR lab 3/9	BR @Rp 2/8	BR lab(B) 3/11	RR A 1/8	RR B 1/8	RR Rn 2/6	TRAP 3 1/14	POP ST 1/8
D	JGE ra 2/5	CMP Rs,A 2/7	CMP #n,A 2/6	CMP Rs,B 2/7	CMP Rs,Rd 3/9	CMP #n,B 2/6	CMP B,A 1/8	CMP #n,Rd 3/8	CMP lab,A 3/11	CMP @Rp,A 2/10	CMP lab(B),A 3/13	RRC A 1/8	RRC B 1/8	RRC Rn 2/6	TRAP 2 1/14	LDSP 1/7
E	JG ra 2/5	DAC Rs,A 2/9	DAC #n,A 2/8	DAC Rs,B 2/9	DAC Rs,Rd 3/11	DAC #n,B 2/8	DAC B,A 1/10	DAC #n,Rd 3/10	CALL lab 3/13	CALL @Rp 2/12	CALL lab(B) 3/15	RL A 1/8	RL B 1/8	RL Rn 2/6	TRAP 1 1/14	STSP 1/8
F	JLO ra 2/5	DSB Rs,A 2/9	DSB #n,A 2/8	DSB Rs,B 2/9	DSB Rs,Rd 3/11	DSB #n,B 2/8	DSB B,A 1/10	DSB #n,Rd 3/10	CALLR lab 3/15	CALLR @Rp 2/14	CALLR lab(B) 3/17	RLC A 1/8	RLC B 1/8	RLC Rn 2/6	TRAP 0 1/14	NOP 1/7

Second byte of two-byte instructions (F4xx):			
F4	8	MOVW n(Rp) 4/15	DIV Rs,A 3/14-63
F4	9	JMPL n(Rp) 4/16	
F4	A	MOV n(Rp),A 4/17	
F4	B	MOV A,n(Rp) 4/16	
F4	C	BR n(Rp) 4/17	
F4	D	CMP n(Rp),A 4/18	
F4	E	CALL n(Rp) 4/20	
F4	F	CALLR n(Rp) 4/22	

#	=	Immediate operand
#16	=	Immediate 16-bit number
lab	=	16-bit label
n	=	Immediate 8-bit number
Pd	=	Peripheral register containing destination byte
Pn	=	Peripheral register
Ps	=	Peripheral register containing source byte
ra	=	Relative address
Rd	=	Register containing destination byte
Rn	=	Register file
Rp	=	Register pair
Rpd	=	Destination register pair
Rps	=	Source register pair
Rs	=	Register containing source byte

Note: All conditional jumps (opcodes 01–0F), BTJO, BTJZ, and DJNZ instructions use two additional cycles if the branch is taken. The BTJO, BTJZ, and DJNZ instructions have a relative address as the last operand.

13.5 Instruction Set Descriptions

The TMS370 instruction set contains 73 instructions that are supported by 246 opcodes. Each operation has an associated opcode. Some instructions, including those using the offset indirect addressing mode, have 16-bit (or dual) opcodes. In two cases, an opcode is shared by two instructions to help you understand all of the functions of an opcode. Several bit manipulation instructions are constructed by the assembler out of other instructions in order to simplify writing and enhance the readability of the program.

The following pages contain the individual instruction descriptions. The instructions are in alphabetical order by mnemonic. Refer to Table 13–1 on page 13-3 for the symbol definitions that are used in the instruction descriptions.

Syntax **ADC** *s, Rd*

Execution (s) + (Rd) + (C) → (Rd)

Options

inst	operands	bytes	cycles	opcode	operation
ADC	B,A	1	8	69	(B)+(A)+(C) → (A)
ADC	Rs,A	2	7	19	(Rs)+(A)+(C) → (A)
ADC	Rs,B	2	7	39	(Rs)+(B)+(C) → (B)
ADC	Rs,Rd	3	9	49	(Rs)+(Rd)+(C) → (Rd)
ADC	#iop8,A	2	6	29	iop8+(A)+(C) → (A)
ADC	#iop8,B	2	6	59	iop8+(B)+(C) → (B)
ADC	#iop8,Rd	3	8	79	iop8+(Rd)+(C) → (Rd)

Status Bits Affected

C Set to 1 on carryout of (s) + (Rd) + (C)
Z Set on result
N Set on result
V (C XOR N) AND (source [bit 7] XNOR destination [bit 7])

Description

ADC adds the contents of the source, the destination register, and the carry bit and stores the result in the destination register.

Adding a 0 to the destination register is equivalent to a conditional increment (increment on carry).

You can use ADC for multiprecision addition of signed or unsigned integers. For example, the 16-bit integer in register pair (R2,R3) can be added to the 16-bit integer in (A,B) as follows:

```
        ADD  R3,B          ;Low-order bytes added
        ADC  R2,A          ;High-order bytes added
```

Examples

```
LABEL1  ADC R66,R117       ;Adds the contents of
                           ;register 66, register
                           ;117, and the carry bit
                           ;and stores the sum in
                           ;register 117

        ADC   B,A          ;Adds the contents of
                           ;register B, register A,
                           ;and the carry bit,and
                           ;stores the sum in
                           ;register A

        ADC #03Ch,R29      ;Adds #3Ch, contents of
                           ;register 29, and the
                           ;carry bit, and stores
                           ;the sum in register 29
```

Syntax **ADD** *s, Rd*

Execution (s) + (Rd) → (Rd)

Options

inst	operands	bytes	cycles	opcode	operation
ADD	B,A	1	8	68	(B)+(A) → (A)
ADD	Rs,A	2	7	18	(Rs)+(A) → (A)
ADD	Rs,B	2	7	38	(Rs)+(B) → (B)
ADD	Rs,Rd	3	9	48	(Rs)+(Rd) → (Rd)
ADD	#iop8,A	2	6	28	iop8+(A) → (A)
ADD	#iop8,B	2	6	58	iop8+(B) → (B)
ADD	#iop8,Rd	3	8	78	iop8+(Rd) → (Rd)

Status Bits Affected

C Set to 1 on carry-out of (s) + (Rd)
Z Set on result
N Set on result
V (C XOR N) AND (Source [bit 7] XNOR Destination [bit 7])

Description ADD adds two bytes and stores the result in the destination register. You can use ADD for signed 2s complement or unsigned addition.

Examples

```
LABEL       ADD B,A          ;Adds the contents of
                             ;registers B and A and
                             ;stores the results in A

            ADD R7,A         ;Adds the contents of R7
                             ;and A and stores the
                             ;results in A

            ADD #TOTAL,R13   ;Adds the value of
                             ;TOTAL to R13 and stores
                             ;the result in R13
```

Syntax **AND** *s , Rd*

Execution (s) AND (Rd) → (Rd)

Options

Inst	operands	bytes	cycles	opcode	operation
AND	A,Pd	2	9	83	(A) AND (Pd) → (Pd)
AND	B,A	1	8	63	(B) AND (A) → (A)
AND	B,Pd	2	9	93	(B) AND (Pd) → (Pd)
AND	Rs,A	2	7	13	(Rs) AND (A) → (A)
AND	Rs,B	2	7	33	(Rs) AND (B) → (B)
AND	Rs,Rd	3	9	43	(Rs) AND (Rd) → (Rd)
AND	#iop8,A	2	6	23	iop8 AND (A) → (A)
AND	#iop8,B	2	6	53	iop8 AND (B) → (B)
AND	#iop8,Rd	3	8	73	iop8 AND (Rd) → (Rd)
AND	#iop8,Pd	3	10	A3	iop8 AND (Pd) → (Pd)

Status Bits Affected

C ← 0
N Set on result
Z Set on result
V ← 0

Description AND logically ANDs the two 8-bit operands. Each bit in the first operand is ANDed with the corresponding bit in the second operand. This is useful for clearing bits. If you need to clear a bit in the destination operand, put a 0 in the corresponding source bit. A 1 in a source bit will not change the corresponding destination bit.

Examples

```
LABEL   AND #01h,R12      ;Clear all bits in R12 except
                          ;bit 0, which will remain
                          ;unchanged

        AND  R7,A         ;AND the contents of R7 to A
                          ;and store the contents in A

        AND  B,P025       ;AND contents of B to P025,
                          ;store the contents in P025
```

Syntax **BR** *XADDR*

Execution XADDR → (PC)

Options

inst	operands	bytes	cycles	opcode	operation
BR	label	3	9	8C	label → (PC)
BR	label(B)	3	11	AC	label+(B) → (PC)
BR	off8(Rp)	4	16	F4 EC	(Rp–1:Rp)+off8 → (PC)
BR	@Rp	2	8	9C	(Rp–1:Rp) → (PC)

Note: label = unsigned 16-bit value
(B) = unsigned 8-bit value
off8 = signed 8-bit value

Status Bits Affected None

Description BR branches to *any* location in memory, including the on-chip RAM. BR supports the four extended absolute addressing modes:

- ❑ Direct
- ❑ Indirect
- ❑ Indexed
- ❑ Offset Indirect

The powerful concept of computed GOTOs is supported by the BR @Rp instruction. Additionally, an indexed branch instruction of the form BR TABLE(B) is an efficient way to execute one of several actions on the basis of a control input; this is similar to the Pascal CASE statement. The program can branch to up to 128 different jump statements. You can use BR to transfer control on character inputs, error codes, etc.

Examples

```
LABEL    BR   LABEL4        ;(PC) ← LABEL4

         BR   5432h         ;(PC) ← 5432h

         BR   LABEL5(B)     ;(PC) ← LABEL5 + (B)

         BR   1234h(B)      ;(PC) ← 1234h + (B)

         BR   @R12          ;(PC) ← (R11:R12) R12 = LSbyte

         BR   56(R10)       ;(PC) ← 56 + (R9:R10) R10 = LSbyte
```

Syntax **BTJO** *s,d,off8*

Execution If (s) AND (d) ≠ 0, then PCN + off8 → (PC), else PCN → (PC)

Options

inst	operands	bytes	cycles†	opcode	jump if
BTJO	A,Pd,off8	3	10/12	86	(A) AND (Pd) ≠ 0
BTJO	B,A,off8	2	10/12	66	(B) AND (A) ≠ 0
BTJO	B,Pd,off8	3	10/12	96	(B) AND (Pd) ≠ 0
BTJO	Rs,A,off8	3	9/11	16	(Rd) AND (A) ≠ 0
BTJO	Rs,B,off8	3	9/11	36	(Rd) AND (B) ≠ 0
BTJO	Rs,Rd,off8	4	11/13	46	(Rd) AND (Rs)≠ 0
BTJO	#iop8,A,off8	3	8/10	26	(A) AND off8 ≠ 0
BTJO	#iop8,B,off8	3	8/10	56	(B) AND off8 ≠ 0
BTJO	#iop8,Rd,off8	4	10/12	76	(Rd) AND off8 ≠ 0
BTJO	#iop8,Pd,off8	4	11/13	A6	(Pd) AND off8 ≠ 0

† The number of cycles to the left of the slash are valid when the jump is not taken; the number of cycles to the right of the slash are valid when the jump is taken.

Status Bits Affected

C ← 0
N Set on (s) AND (d)
Z Set on (s) AND (d)
V ← 0

Description BTJO jumps if at least one corresponding bit position in the source and destination are both 1. The source operand can be used as a bit mask to test for one or more 1 bits in the specified register. The operands are not changed by this instruction. If one or more corresponding 1 bits are found, the program branches to the offset (refer to the table below).

(s)	(d)	Jump?
00000001	xxxxxxx0	No
00000001	xxxxxxx1	Yes
00000011	xxxxxx00	No
11110000	1000xxxx	Yes
11110000	1001xxxx	Yes

Examples

```
LABEL    BTJO  #014,R4,ISSET    ;Jump to ISSET if R4
                                ;(bit 2) or R4  (bit
                                ;4) is a 1

         BTJO  #01,A,LOOP       ;Jump to LOOP if bit 0
                                ;of register A is a 1

         BTJO  R37,R113,START   ;Jump to START if any
                                ;1 bit of R113 corres-
                                ;ponds to a 1 bit
                                ;in R37
```

Syntax **BTJZ** *s,d,off8*

Execution If (s) AND NOT (d) ≠ 0, then PCN + off8 → (PC), else PCN → (PC)

Options

inst	operands	bytes	cycles†	opcode	jump if
BTJZ	A,Pd,off8	3	10/12	87	(A) AND NOT(Pd) ≠ 0
BTJZ	B,A,off8	2	10/12	67	(B) AND NOT(A) ≠ 0
BTJZ	B,Pd,off8	3	10/12	97	(Pd) AND NOT(B) ≠ 0
BTJZ	Rd,A,off8	3	9/11	17	(Rd) AND NOT(A) ≠ 0
BTJZ	Rd,B,off8	3	9/11	37	(Rd) AND NOT(B) ≠ 0
BTJZ	Rs,Rd,off8	4	11/13	47	(Rs) AND NOT(Rd) ≠ 0
BTJZ	#iop8,A,off8	3	8/10	27	off8 AND NOT (A) ≠ 0
BTJZ	#iop8,B,off8	3	8/10	57	off8 AND NOT (B) ≠ 0
BTJZ	#iop8,Rd,off8	4	10/12	77	off8 AND NOT (Rd) ≠ 0
BTJZ	#iop8,Pd,off8	4	11/13	A7	off8 AND NOT (Pd) ≠ 0

† The number of cycles to the left of the slash are valid when the jump is not taken; the number of cycles to the right of the slash are valid when the jump is taken.

Status Bits Affected

C ← 0
N Set on (s) AND NOT (Rd)
Z Set on (s) AND NOT (Rd)
V ← 0

Description BTJZ jumps if at least one corresponding bit position has a 1 in the source and a 0 in the destination (refer to the table below). The source operand can be used as a bit mask to test for zero bits in the specified register. The operands are not changed by this instruction. The jump is calculated starting from the opcode of the instruction immediately after the BTJZ.

(s)	(d)	Jump?
00000001	xxxxxxx0	Yes
00000001	xxxxxxx1	No
11000000	11xxxxxx	No
11110000	0111xxxx	Yes
11110000	0110xxxx	Yes

Examples

```
LABEL     BTJZ  A,P23,ZERO  ;If any 1 bits in A
                            ;correspond to 0 bits
                            ;in P23, 0 then jump to
                            ;ZERO

          BTJZ #0FFh,A,NEXT ;If A contains any 0
                            ;bits, jump to NEXT

          BTJZ  R7,R15,OUT  ;If any 0 bits in R15
                            ;correspond to 1 bits
                            ;in R7, jump to OUT
```

Syntax **CALL** *XADDR*

Execution

(SP) + 1 → (SP)
PCN MSbyte → ((SP))
(SP) + 1 → (SP)
PCN LSbyte → ((SP))
XADDR → (PC)
(The stack contains the address of the instruction immediately following the CALL.)

Options

inst	operands	bytes	cycles	opcode	operation
CALL	label	3	13	8E	label → (PC)
CALL	label(B)	3	15	AE	label+(B) → (PC)
CALL	off8(Rp)	4	20	F4 EE	(Rp–1:Rp)+off8 → (PC)
CALL	@Rp	2	12	9E	(Rp–1:Rp) → (PC)

Note: offset = signed 16-bit value
(B) = unsigned 8-bit value
off8 = signed 8-bit value

Status Bits Affected None

Description CALL invokes a subroutine and pushes the PC contents on the stack. The operand indicates the starting address of the subroutine. The extended addressing modes of the CALL instruction support powerful transfer of control functions.

Examples

```
LABEL   CALL  LABEL4      ;Push PC; (PC) ← LABEL4

        CALL  5432h       ;Push PC; (PC) ← 5432h

        CALL  LABEL5(B)   ;Push PC; (PC) ← LABEL5 + (B)

        CALL  1234h(B)    ;Push PC; (PC) ← 1234h + (B)

        CALL  @R12        ;Push PC; (PC) ← (R11:R12)
                          ;R12 = LSbyte

        CALL  56(R10)     ;Push PC; (PC) ← 56 +
                          ;(R9:R10)   R10 = LSbyte
```

Syntax

CALLR *XADDR*

Execution

(SP) + 1 → (SP)
PCN MSbyte → ((SP))
(SP) + 1 → (SP)
PCN LSbyte → ((SP))
XADDR + PCN → (PC)

Options

inst	operands	bytes	cycles	opcode	operation
CALLR	label	3	15	8F	off16 + PCN → (PC)
CALLR	label(B)	3	17	AF	off16 + (B) + PCN → (PC)
CALLR	off8(Rp)	4	22	F4 EF	(Rp–1:Rp) + off8 + PCN → (PC)
CALLR	@Rp	2	14	9F	(Rp–1:Rp) + PCN → (PC)

Note: off16 = signed 16-bit value
(B) = unsigned 8-bit value
off8 = signed 8-bit value

Status Bits Affected

None

Description

CALLR is similar to CALL, but it uses a value relative to the current program counter (PCN). The extended relative addressing modes of the CALLR instruction support powerful transfer of control functions. This is useful for relocatable code produced by linkers, compilers, or other high-level language structures. The assembler automatically calculates the correct offset value for the two modes by using labels in the operands.

Examples

Direct Addressing

```
LABEL     CALLR LABEL4      ;push PC ; (PC) ← PCN +
                            ;off16,  off16 = LABEL4-PCN

          CALLR 5432h       ;push PC ; (PC) ← PCN +
                            ;5432h
```

Indexed Addressing

```
          CALLR LABEL5(B)   ;push PC ; (PC) ← PCN +
                            ;off16 +(B)
                            ;off16=LABEL5 - PCN

          CALLR 1234h(B)    ;push PC ; (PC) ← PCN +
                            ;1234h + (B)
```

Indirect Addressing

```
          CALLR @R12        ;push PC ; (PC) ← PCN +
                            ;(R11:R12)
                            ;R12=LSbyte
```

Offset Indirect Addressing

```
          CALLR 56(R10)     ;push PC ; (PC) ← PCN
                            ;+ 56 + (R9:R10)
                            ;R10=LSbyte
```

Syntax **CLR** *Rd*

Execution 0 → (Rd)

Options

inst	operands	bytes	cycles	opcode	operation
CLR	A	1	8	B5	0 → (A)
CLR	B	1	8	C5	0 → (B)
CLR	Rd	2	6	D5	0 → (Rd)

Status Bits Affected

C ← 0
N ← 0
Z ← 1
V ← 0

Description CLR clears or initializes to 0 any register, including registers A and B.

Examples

```
LABEL    CLR  B          ;Clear register B

         CLR  A          ;Clear register A

         CLR  R105       ;Clear register 105
```

Syntax **CLRC**

Execution Set status bits

Options

inst	operands	bytes	cycles	opcode
CLRC	none	1	9	B0

Status Bits Affected

C ← 0
N Set on value of register A
Z Set on value of register A
V ← 0

Description CLRC clears the carry flag. This instruction may be required before an arithmetic or rotate instruction. The logical and move instructions typically clear the carry bit. The CLRC opcode is equivalent to the TST A opcode.

Example

```
LABEL  CLRC    ;Clear the carry bit
```

Syntax **CMP** *s,d*

Execution (d) – (s) computed but not stored

Options

Inst	operands	bytes	cycles	opcode	operation
General:					
CMP	B,A	1	8	6D	(A)–(B)
CMP	Rs,A	2	7	1D	(A)–(Rs)
CMP	Rs,B	2	7	3D	(B)–(Rs)
CMP	Rs,Rd	3	9	4D	(Rd)–(Rs)
CMP	#iop8,A	2	6	2D	(A)–iop8
CMP	#iop8,B	2	6	5D	(B)–iop8
CMP	#iop8,Rd	3	8	7D	(Rd)–iop8
Extended:					
CMP	label,A	3	11	8D	(A)–(label)
CMP	label(B),A	3	13	AD	(A)–(label+(B))
CMP	off8(Rp),A	4	18	F4 ED	(A)–((Rp–1:Rp)+off8)
CMP	@Rp,A	2	10	9D	(A)–((Rp–1:Rp))
CMP	off8(SP),A	2	8	F3	(A)–((SP)+off8)

Note: Operations are computed but not stored. Status bits are set on results.

Status Bits Affected

C 1 if (d) ≥ (s)
N Sign of result
Z 1 if (d) = (s)
V (C XOR N) AND (Source [bit 7] XOR Destination [bit 7])

Description CMP compares the destination operand to the source operand and sets the status bits. The CMP instruction is usually used in conjunction with a jump instruction. Table 13–5 shows which jump instructions can be used on status conditions set by CMP execution. There are only seven possible outcomes of the status register after a compare instruction. The jump instructions JC and JHS are equivalent after a compare.

Table 13–5. Compare Instruction Examples—Status Bit Values

Operand Opcodes (S) (D)	Status Bits CNZV	JGE	JG	JL	JLE	JLO	JHS	JC	JNC	JN	JP	JEQ /JZ	JPZ	JNE/ JNZ	JV	JNV
FF 00 81 00	0000	1	1	0	0	1	0	0	1	0	1	0	1	1	0	1
80 00 80 7F	0101	1	1	0	0	1	0	0	1	1	0	0	0	1	1	0
00 7F 20 30 90 A0	1000	1	1	0	0	0	1	1	0	0	1	0	1	1	0	1
7F 00 30 20 A0 90	0100	0	0	1	1	1	0	0	1	1	0	0	0	1	0	1
7F 80	1001	0	0	1	1	0	1	1	0	0	1	0	1	1	1	0
00 FF 00 81 00 80	1100	0	0	1	1	0	1	1	0	1	0	0	0	1	0	1
7F 7F	1010	1	0	0	1	0	1	1	0	0	0	1	1	0	0	1

Notes: 1) Signed Jumps: JGE, JG, JL, JLE.
Unsigned Jumps: JLO, JHS.
Test Bits: JC, JNC, JN, JP, JEQ/JZ, JPZ, JNE/JNZ, JV, JNZ

2) 1 = jump was taken; 0 = does not jump

Examples

```
LABEL   CMP  R13,R89       ;Set status bits on
                           ;result of R89 minus R13

        CMP  R39,B         ;Set status bits on result
                           ;of (B) minus R39

        CMP  #003,A        ;Set status bits on result
                           ;of (A) minus #03h

        CMP  TABLE(B),A    ;Set status bits  on result
                           ;of (A) minus (TABLE + (B))
```

Syntax **CMPBIT** *name*

Execution NOT <name> → <name>

Options

inst	operands	bytes	cycles	opcode	operation
CMPBIT	Rname	3	8	75	NOT (bit) → (bit) Register bits
CMPBIT	Pname	3	10	A5	NOT (bit) → (bit) PF bits

Status Bits Affected

C ← 0
N Set on result of (Mask XOR (s))
Z Set on result of (Mask XOR (s))
V ← 0

Description CMPBIT is an assembler constructed instruction that conveniently complements the value of the named bit without having to specify a register or mask. This enhances the readability of the program. The CMPBIT instruction assembles to the instructions XOR #iop8,Rd or XOR #iop8,Pd. The name for the bit is defined by the .DBIT assembler directive.

Examples

```
INT1ENA   .DBIT 7,P017      ;Interrupt 1 bit is now
                            ;named INT1ENA

TEST      .DBIT  4,R33      ;Bit 4 of register 33 is now
                            ;named TEST

LABEL     CMPBIT TEST       ;Invert the value of the TEST
                            ;bit.

          CMPBIT INT1ENA    ;Change the interrupt 1
                            ;enabled condition.
```

COMPL *2s Complement*

Syntax **COMPL** *Rn*

Execution 0 – Rn → Rn

Options

inst	operands	bytes	cycles	opcode	operation
COMPL	A	1	8	BB	NOT(A)+1 → (A)
COMPL	B	1	8	CB	NOT(B)+1 → (B)
COMPL	Rn	2	6	DB	NOT(Rn)+1 → (Rn)

Status Bits Affected

C Set on result
N Set on result
Z Set on result
V ← 0

Description COMPL provides a logical or 2s complement of the operand. This is the equivalent of an inversion of all the bits followed by an increment. The instruction is useful in doing arithmetic with signed numbers.

Examples

```
LABEL    COMPL  A          ;Complement register A

         COMPL  B          ;Complement register B

         COMPL  R82        ;Complement register 82
```

Syntax **DAC** *s,Rd*

Execution (s) + (Rd) + (C) → (Rd), produces a decimal result

Options

inst	operands	bytes	cycles	opcode	operation
DAC	B,A	1	10	6E	(B)+(A)+(C) → (A)
DAC	Rs,A	2	9	1E	(Rs)+(A)+(C) → (A)
DAC	Rs,B	2	9	3E	(Rs)+(B)+(C) → (B)
DAC	Rs,Rd	3	11	4E	(Rs)+(Rd)+(C) → (Rd)
DAC	#iop8,A	2	8	2E	iop8+(A)+(C) → (A)
DAC	#iop8,B	2	8	5E	iop8+(B)+(C) → (B)
DAC	#iop8,Rd	3	10	7E	iop8+(Rd)+(C) → (Rd)

Status Bits Affected

C 1 if value of (s) + (Rd) + C > 99
N Set on result
Z Set on result
V Undefined

Description DAC adds bytes in binary-coded decimal (BCD) form. Each byte is assumed to contain two BCD digits. DAC is not defined for non-BCD operands. DAC with an immediate operand of zero value is equivalent to a conditional increment of the destination operand (increment destination on carry).

The DAC instruction automatically performs a decimal adjust on the binary sum of (s) + (d) + C. The carry bit is added to facilitate adding multibyte BCD strings; therefore, the carry bit must be cleared before execution of the first DAC instruction.

Examples

```
LABEL    DAC #024h,A    ;If register A contains 097h
                        ;and C = 0,then the final result
                        ;put into A is 021h and the carry
                        ;bit is set

         DAC  R55,R7    ;Add the BCD value of R55,
                        ;and the carry bit to the
                        ;BCD value of R7

         DAC  B,A       ;Add the carry bit to the
                        ;BCD value in register B
                        ;to register A
```

Syntax **DEC** *Rn*

Execution (Rn) – 1 → (Rn)

Options

inst	operands	bytes	cycles	opcode	operation
DEC	A	1	8	B2	(A)-1 → (A)
DEC	B	1	8	C2	(B)-1 → (B)
DEC	Rn	2	6	D2	(Rn)-1 → (Rn)

Status Bits Affected

C 0 if (Rn) decrements from 00h to FFh; 1 otherwise
N Set on result
Z Set on result
V 1 if (Rn) decrements from 80h to 7Fh; 0 otherwise

Description DEC subtracts 1 from any register. It is useful for counting and addressing byte arrays.

Examples

```
LABEL   DEC R102        ;Decrement R102 by 1

        DEC  A          ;Subtract 1 from the contents of
                        ;register A

        DEC  B          ;Subtract 1 from the contents of
                        ;register B
```

Syntax **DINT**

Execution 0 → (ST)

Options

inst	operands	bytes	cycles	opcode
DINT	none	2	6	F0 00

Status Bits Affected

C ← 0
N ← 0
Z ← 0
V ← 0
IE1 ← 0
IE2 ← 0

Description DINT simultaneously disables all interrupts. Since the interrupt enable flags are stored in the status register, the POP ST or RETI instructions may re-enable interrupts, even though a DINT instruction has been executed. During the interrupt service, the interrupt enable bit is automatically cleared after the old status register value has been pushed onto the stack. The DINT instruction is equal to the LDST #00 instruction.

Example

```
LABEL     DINT      ;Disable high- and low-level interrupts.
```

Syntax **DIV** *Rs, A*

Execution A:B/(Rs) → A(=quo), B(=rem)

Options

inst	operands	bytes	cycles	opcode	operation
DIV	Rs,A	3	55-63	F4 F8	(A:B)/(Rs) Quotient → A Remainder → B

Note: If overflow occurs, 14 cycles are used, and C,N,Z,V = 1

Status Bits Affected

C ← 0
N Set on results (register A)
Z Set on results (register A)
V ← 0

Description DIV divides the 16-bit value in the A:B register pair by the 8-bit value in the specified register. The resulting 8-bit quotient is stored in A. Overflow conditions are checked before execution; if an overflow is detected, the operands are left unchanged, the status bits C,N,Z, and V are set to 1, and the instruction is aborted. Execution time varies from 55–63 cycles, depending on the operands, with an overflow condition taking only 14 cycles. The average execution time is 57 cycles.

Example

```
LABEL   DIV  R10,A     ;R10 is divided into the
                      ;A:B register pair (A = MSbyte)

        JC OVERFLOW   ;Carry is 1 on overflow conditions
```

Syntax **DJNZ** *Rn,off8*

Execution (Rn) – 1 → (Rn)
If (Rn) ≠ 0, then PCN + (off8) → (PC), else PCN → (PC)

Type Single Operand

Options

Inst	operands	bytes	cycles†	opcode	operation
DJNZ	A,off8	2	10/12	BA	(A)–1 → (A), jump if (A) ≠ 0
DJNZ	B,off8	2	10/12	CA	(B)–1 → (B) jump if (B) ≠ 0
DJNZ	Rn,off8	3	8/10	DA	(Rn)–1 → (Rn) jump if (Rn) ≠ 0

† The number of cycles to the left of the slash are valid when the jump is not taken; the number of cycles to the right of the slash are valid when the jump is taken.

Status Bits Affected None

Description DJNZ is used for looping control. It combines the DEC and the JNZ instructions, providing a faster and more compact instruction. DJNZ does not change the status bits.

Examples

```
LABEL   DJNZ  R15,THERE   ;Decrement R15. If R15 ≠ 0,
                          ;jump to THERE

        DJNZ  A,AGAIN     ;Decrement A; if A ≠ 0,
                          ;jump to AGAIN

        DJNZ  B,BACK      ;Decrement B; if B ≠ 0,
                          ;jump to BACK
```

Syntax **DSB** *s,Rd*

Execution (Rd) – (s) – 1 + (C) → (Rd) (decimal result)

Options

inst	operands	bytes	cycles	opcode	operation
DSB	B,A	1	10	6F	(A)–(B)–1+(C) → (A)
DSB	Rs,A	2	9	1F	(A)–(Rs)–1+(C) → (A)
DSB	Rs,B	2	9	3F	(B)–(Rs)–1+(C) → (B)
DSB	Rs,Rd	3	11	4F	(Rd)–(Rs)–1+(C) → (Rd)
DSB	#iop8,A	2	8	2F	(A)–iop8–1+(C) → (A)
DSB	#iop8,B	2	8	5F	(B)–iop8–1+(C) → (B)
DSB	#iop8,Rd	3	10	7F	(Rd)–iop8–1+(C) → (Rd)

Status Bits Affected

C 1 if no borrow required, 0 if borrow required
N Set on result
Z Set on result
V Undefined

Description DSB performs multiprecision BCD subtraction. The DSB instruction with an immediate operand of zero value is equivalent to a conditional decrement of the destination operand, depending on the carry bit. The carry bit functions as a no-borrow bit; if no borrow is required, the carry bit should be set to 1. You can accomplish this by executing the SETC instruction. The DSB instruction is undefined for non-BCD operands.

Example

```
LABEL    DSB  R15,R76      ;R76 minus R15 minus 1 plus
                           ;the carry bit is stored
                           ;in R76

         DSB  A,B         ;Register B minus register
                           ;A minus 1 plus the carry
                           ;bit is stored in
                           ;register B

         DSB  #0,R5       ;R5 - 1 →  R5, if C = 0
                           ;R5 →  R5 if C = 1
```

Syntax **EINT**

Execution 0Ch → (ST)

Options

inst	operands	bytes	cycles	opcode
EINT	none	2	6	F0 0C

Status Bits Affected

C ← 0
N ← 0
Z ← 0
V ← 0
IE1 ← 1
IE2 ← 1

Description EINT simultaneously enables all global interrupts. Since the interrupt enable flags are stored in the status register, the POP ST or RETI instructions may disable interrupts, even though an EINT instruction has been executed. During the interrupt service, the interrupt enable bit is automatically cleared after the old status register value has been pushed onto the stack. Thus, the EINT instruction must be included inside the interrupt service routine to permit nested or multilevel interrupts. This instruction is equivalent to the LDST #00Ch instruction.

Example

```
LABEL    EINT              ;All interrupts are enabled.
```

Syntax **EINTH**

Execution 04h → (ST)

Options

inst	operands	bytes	cycles	opcode
EINTH	none	2	6	F0 04

Status Bits Affected

C ← 0
N ← 0
Z ← 0
V ← 0
IE1 ← 1
IE2 ← 0

Description EINTH is similar to the EINT instruction but enables only high-level (1) interrupts and disables low-level interrupts. This assembles to the LDST #04h instruction.

Example

```
LABEL     EINTH        ;All level 1 interrupts are enabled.
```

Syntax **EINTL**

Execution 08h → (ST)

Options

Inst	**operands**	**bytes**	**cycles**	**opcode**
EINTL	none	2	6	F0 08

Status Bits Affected

C ← 0
N ← 0
Z ← 0
V ← 0
IE1 ← 0
IE2 ← 1

Description EINTL is similar to the EINT instruction but enables only low-level (2) interrupts while disabling high-level interrupts. This assembles to the LDST #08h instruction.

Example

```
LABEL     EINTL        ;All level 2 interrupts are enabled.
```

Syntax **IDLE**

Execution (PC) + 1 → (PC) after return from interrupt

Options

inst	operands	bytes	cycles	opcode
IDLE	none	1	6 (minimum)	F6

Status Bits Affected None

Description The IDLE instruction causes the device to enter one of three modes: halt, standby, or idle. Two of these modes, halt and standby, use only a fraction of the normal operating power.

- ❑ In standby mode, the on-chip oscillator and timer 1 module remain active.
- ❑ In halt mode, the oscillator is off, and the chip consumes the least amount of power.

If you execute an IDLE instruction when low-power modes are disabled through a programmable contact (mask-ROM devices only), the device will *always* enter the idle mode.

Appropriate interrupts must be enabled before the device enters idle mode. For more information on the low-power or idle modes, refer to Section 4.2.

Examples

```
LABEL     IDLE              ;Enter idle mode and
                            ;wait for interrupt
```

Syntax **INC** *Rn*

Execution (Rn) + 1 → (Rn)

Options

inst	operands	bytes	cycles	opcode	operation
INC	A	1	8	B3	(A)+1 → (A)
INC	B	1	8	C3	(B)+1 → (B)
INC	Rn	2	6	D3	(Rn)+1 → (Rn)

Status Bits Affected

C 1 if (Rd) incremented from FFh to 00h; 0 otherwise
N Set on result
Z Set on result
V 1 if (Rn) incremented from 7Fh to 80h; 0 otherwise

Description INC increments the value of any register. It is useful for incrementing counters.

Examples

```
LABEL    INC  A          ;Increment register A by 1

         INC  B          ;Increment register B by 1

         INC  R43        ;Increment register 43 by 1
```

Syntax **INCW** *#iop8,Rp*

Execution (Rp) + #iop8 → (Rp)

Options

inst	operands	bytes	cycles	opcode	operation
INCW	#iop8,Rp	3	11	70	iop8+(Rp–1:Rp) → (Rp–1:Rp) iop8= 8-bit immediate operand

Status Bits Affected

C Set to 1 on carry out of iop8 + (Rp)
N Set on result
Z Set on MSbyte
V (C XOR N) AND (MSB iop8 XNOR MSB (Rd))

Description INCW increments the value of any register pair by the amount specified. The register pair can be incremented by as much as 127 or decremented by as much as 128. This instruction is useful for incrementing counters into large tables. The iop8 is sign-extended in order to perform 16-bit 2s-complement addition. The JC and JNC are commonly used after the INCW instruction for loop control.

Examples

```
LABEL    INCW #1,R10        ;Increment R9:R10 by 1

         INCW #-1,R10       ;Decrement register R9:R10 by 1

         INCW #100,R255     ;Increment register pair
                            ;R254:R255
```

Syntax **INV** *Rn*

Execution NOT(Rn) → (Rn)

Options

inst	operands	bytes	cycles	opcode	operation
INV	A	1	8	B4	NOT(A) → (A)
INV	B	1	8	C4	NOT(B) → (B)
INV	Rn	2	6	D4	NOT(Rn) → (Rn)

Status Bits Affected

C ← 0
N Set on result
Z Set on result
V ← 0

Description INV performs a 1s complement of the operand. A 1s complement inverts the value of every bit in the register. You can perform a 2s complement of the operand by following the INV instruction with an increment (INC) or by simply using the COMPL instruction.

Examples

```
LABEL     INV A          ;Invert register A (0s become 1s,
                         ;1s become 0s)

          INV B          ;Invert register B

          INV R82        :Invert register 82
```

Syntax **JBIT0** *name, off8*

Execution If bit (name) = 0, then PCN + off8 → (PC), else PCN → PC

Options

inst	operands	bytes	cycles	opcode
JBIT0	Rname,off8	4	10	77
JBIT0	Pname,off8	4	11	A7

Note: Add 2 cycles if jump is taken

Status Bits Affected

C ← 0
N Set on (s) AND NOT (Rd)
Z Set on (s) AND NOT (Rd)
V ← 0

Description The JBIT0 is an assembler-constructed instruction that conveniently jumps to the label if the value of the named bit is zero. This enhances the readability of the program because the source does not have to specify both the register containing the bit and also a mask. JBIT0 is assembled to BTJZ #iop8,Rd,label or BTJZ #iop8,Pd,label. The name for the bit is defined by the .DBIT assembler directive.

Example

```
MCDATA    .DBIT 2,P010        ;MC data in bit 2 of
                              ;SCCR0 (P010) is now
                              ;named MCDATA

 BIT4     .DBIT  4,R3         ;Bit 4 of register 3 is
                              ;now named BIT4

          JBIT0 BIT4,THERE    ;Jump to THERE if bit 4 in
                              ;register 3 is zero.

          JBIT0 MCDATA,HERE   ;Jump to HERE if the MC pin
                              ;is zero
```

Syntax **JBIT1** *name, off8*

Execution bit (name) = 1, then PCN + off8 → (PC), else PCN → (PC)

Options

inst	operands	bytes	cycles	opcode	operation
JBIT1	Rname,off8	4	10	76	register bits
JBIT1	Pname,off8	4	11	A6	peripheral bits

Note: Add two cycles if jump is taken.

Status Bits Affected

C ← 0
N Set on (s) AND (Rd)
Z Set on (s) AND (Rd)
V ← 0

Description The JBIT1 is an assembler-constructed instruction that conveniently jumps to the label if the value of the named bit is one. This instruction enhances the readability of the program because the source does not have to specify both the register containing the bit and also a mask. This instruction assembles to BTJO #iop8,Rd,label or BTJO #iop8,Pd,label. The name for the bit is defined by the .DBIT assembler directive.

Examples

```
BUSYP     .DBIT 7,P01C       ;Busy bit in PEECTL
                             ;(program EEPROM) is now
                             ;named BUSYP

BIT0      .DBIT 0,R100       ;Bit 0 of register 100 is
                             ;now named BIT0

LABEL     BIT1 BIT0, THERE   ;Jump to THERE if bit 0 in
                             ;register 100 is a one.

          JBIT1 BUSYP,HERE   ;Jump to HERE if the program
                             ;EEPROM is busy.
```

Syntax ***Jcnd*** *off8*

Execution If tested condition is true, (PCN) + off8 → (PC), else PCN → (PC)

Status Bits Affected None

Description The *Jcnd* instructions are commonly used after a CMP instruction to branch according to the relative values of the operands tested. After MOV operations, a JZ or JNZ can be used to test whether the value moved was equal to zero; in this case, JN and JPZ are used to test the sign bit of the value moved. In addition, the program can check the overflow bit V after executing an arithmetic instruction with the JV or JNV instructions.

All *Jcnd* instructions are two bytes in length and require 5 cycles to execute; however, if the jump is taken, the instruction requires 7 cycles.

Instruction	Mnemonic	Opcode	C	N	Z	V	Operation
Jump if Carry	JC	03	1	x	x	x	
Jump if No Carry	JNC	07	0	x	x	x	
Jump if Equal	JEQ	02	x	x	1	x	
Jump if Not Equal	JNE	06	x	x	0	x	
Jump if Nonzero	JNZ	06	x	x	0	x	
Jump if Zero	JZ	02	x	x	1	x	
Jump if Lower	JLO	0F	0	x	0	x	
Jump if Higher or Same	JHS	0B	–	x	–	x	(C = 1) OR (Z = 1)
							—Signed Operation—
Jump if Greater	JG	0E	x	–	–	–	Z OR (N XOR V) = 0
Jump if Greater or Equal	JGE	0D	x	–	x	–	N XOR V = 0
Jump if Less	JL	09	x	–	x	–	N XOR V = 1
Jump if Less or Equal	JLE	0A	x	–	–	–	Z OR (N XOR V) = 1
Jump if Negative	JN	01	x	1	x	x	
Jump if Positive	JP	04	x	0	0	x	
Jump if Positive or Zero	JPZ	05	x	0	x	x	
Jump if No Overflow	JNV	0C	x	x	x	0	
Jump if Overflow	JV	08	x	x	x	1	

Note: Legend :

0	Status bit always cleared.
1	Status bit always set.
x	Status bit cleared or set on results.
–	Status bit not affected.

These two-instruction jump sequences are used in place of the single conditional jumps for the numbers shown in the tables. ($ is the current PC value.)

Signed Number Jumps			Unsigned Number Jumps		
Condition	**True**	**False**	**Condition**	**True**	**False**
d < s	JL	JGE	d < s	JLO	JHS
d ≤ s	JLE	JG	d ≤ s	2	3
d = s	JEQ	JNE	d = s	JEQ	JNE
d ≥ s	JGE	JL	d ≥ s	JHS	JLO
d > s	JG	JLE	d > s	3	2
Negative	JN	JPZ			
Positive	JP	1			
Positive or 0	JPZ	JN			

Notes: 1) JZ LABEL
JN LABEL

2) JEQ LABEL
JLO LABEL

3) JEQ $+4
JHS LABEL

Status Bit Jumps		
Bits	**True**	**False**
C	JC	JNC
N	JN	JPZ
Z	JZ	JNZ
V	JV	JNV

Examples

```
LABEL   JNC   TABLE      ;If the carry bit is clear,
                         ;jump to TABLE

        JP    HERE       ;If the negative and zero flags
                         ;are clear, jump to HERE

        JZ    NEXT       ;If the zero flag is set, jump
                         ;to NEXT
```

Syntax **JMP** *off8*

Execution PCN + off8 → (PC)

Options

inst	operands	bytes	cycles	opcode	operation
JMP	off8	2	7	00	PCN+off8 → (PC)

Status Bits Affected None

Description JMP jumps unconditionally to the address specified in the operand. The second byte of the JMP instruction contains the 8-bit relative address of the operand. The operand address must therefore be within –128 to +127 bytes of the location of the instruction following the JMP instruction. The assembler will indicate an error if the target address is beyond –128 to +127 bytes from the next instruction. For a longer jump, you can use the BR (branch) or the JMPL instruction.

Example

```
LABEL     JMP THERE          ;Load the PC with the address
                             ;of THERE
```

Syntax **JMPL** *XADDR*

Execution PCN + D → (PC)

Options

Inst	operands	bytes	cycles	opcode	operation
JMPL	label	3	9	89	off16+PCN → (PC)
JMPL	label(B)	3	11	A9	off16+(B)+PCN → (PC)
JMPL	off8(Rp)	4	16	F4 E9	(Rp–1:Rp)+off8+PCN → (PC)
JMPL	@Rp	2	8	99	(Rp–1:Rp)+PCN → (PC)

Note: offset = signed 16-bit value
off8 = signed 8-bit value
(B) = unsigned 8-bit value

Status Bits Affected None

Description JMPL is similar to JMP instruction but generates a 16-bit (instead of 8-bit) signed offset to the program counter.

Examples

```
LABEL    JMPL  LABEL4        ;(PC) ← PCN + offset
                             ;offset=LABEL4-PCN

         JMPL 5432h         ;(PC) ← PCN + 5432h

         JMPL LABEL5(B)     ;(PC) ← PCN + off8 + (B)
                             ;offset=LABEL5 - PCN

         JMPL  1234h(B)     ;(PC) ← PCN + 1234h + (B)

         JMPL  @R12         ;(PC) ← PCN + (R11:R12)
                             ;R12=LSbyte

         JMPL  56(R10)      ;(PC) ← PCN + 56 + (R9:R10)
                             ;R10 = LSbyte

         JMPL -2(R10)       ;(PC) ← PCN -2 + (R9:R10)
                             ;R10 = LSbyte
```

Syntax **LDSP**

Execution (B) → (SP)

Options

Inst	operands	bytes	cycles	opcode
LDSP	none	1	7	FD

Status Bits Affected None

Description LDSP copies the contents of register B to the stack pointer (SP). Use LDSP to initialize the stack pointer.

Example

```
MOV     #080h,B         ;Register B = SP value.

LABEL   LDSP            ;Copy register B to the stack
                        ;pointer.
```

Syntax **LDST** *#iop8*

Execution (iop8) → (ST)

Options

inst	operands	bytes	cycles	opcode	operation
LDST	#iop8	2	6	F0	iop8 → (ST)

Status Bits Affected

C Set on value loaded
N Set on value loaded
Z Set on value loaded
V Set on value loaded
IE1 Set on value loaded
IE2 Set on value loaded

Description The LDST copies the immediate value operand to the status register (ST). Any combination of bits can be loaded into the status register using this command. Some instructions such as EINT, EINTL, EINTH or DINT are assembled into this instruction.

Example

```
LABEL     LDST #08Ch          ;Copy immediate value to
                              ;the status register and
                              ;set IE2 bit
```

Syntax **MOV** *s,d*

Execution (s) → (d)

Options

inst	operands	bytes	cycles	opcode	operation
Register:					
MOV	A,B	1	9	C0	(A) → (B)
MOV	A,Rd	2	7	D0	(A) → (Rd)
MOV	B,A	1	8	62	(B) → (A)
MOV	B,Rd	2	7	D1	(B) → (Rd)
MOV	Rs,A	2	7	12	(Rs) → (A)
MOV	Rs,B	2	7	32	(Rs) → (B)
MOV	Rs,Rd	3	9	42	(Rs) → (Rd)
MOV	#iop8,A	2	6	22	iop8 → (A)
MOV	#iop8,B	2	6	52	iop8 → (B)
MOV	#iop8,Rd	3	8	72	iop8 → (Rd)
Peripheral:					
MOV	A,Pd	2	8	21	(A) → (Pd)
MOV	B,Pd	2	8	51	(B) → (Pd)
MOV	Rs,Pd	3	10	71	(Rs) → (Pd)
MOV	Ps,A	2	8	80	(Ps) → (A)
MOV	Ps,B	2	8	91	(Ps) → (B)
MOV	Ps,Rd	3	10	A2	(Ps) → (Rd)
MOV	#iop8,Pd	3	10	F7	iop8 → (Pd)
Extended:					
MOV	A,@Rp	2	9	9B	(A) → ((Rp–1:Rp))
MOV	A,label	3	10	8B	(A) → (label)
MOV	A,label(B)	3	12	AB	(A) → (label+(B))
MOV	A,off8(SP)	2	7	F2	(A) → (off8+(SP))
MOV	A,off8(Rp)	4	17	F4 EB	(A) → (off8+(Rp–1:Rp))
MOV	@Rp,A	2	9	9A	((Rp–1:Rp)) → (A)
MOV	label,A	3	10	8A	(label) → (A)
MOV	label(B),A	3	12	AA	(label+(B)) → (A)
MOV	off8(SP),A	2	7	F1	(off8 + (SP)) → (A)
MOV	off8(Rp),A	4	17	F4 EA	(off8 +(Rp–1:Rp)) → (A)

13 **Status Bits Affected**

C ← 0
N Set on value loaded
Z Set on value loaded
V ← 0

Description

MOV transfers values within the memory space. Immediate values can be loaded directly into the registers. In extended addressing modes, the processor must use register A. A MOV instruction that uses register A or B as an operand requires fewer bytes. When the MOV Pn,Rn and MOV Rn,Pn instructions are assembled into machine code, their operands are reversed.

Examples

```
MOV    A,B              ;Move the contents of register
                        ;A to register B

MOV    R32,R105         ;Move the contents of register
                        ;32 to register 105

MOV    #010h,R3         ;Move #010h to register 3

MOV    A,LABEL(B)       ;Move the contents of register A to the
                        ;location LABEL+B

MOV    A,2Fh(R32)       ;Move the contents of register A to the
                        ;location 002Fh+(R31:R32)

MOV    @ROF,A           ;Use the contents of the register pair ROE:ROF
                        ;as address. Move the contents of that
                        ;address to register A

MOV    LABEL,A          ;Move the contents of the location at
                        ;LABEL to register A
```

Syntax **MOVW** *s,Rpd*

Execution (s) → (Rpd)

Options

inst	operands	bytes	cycles	opcode	operation
MOVW	#iop16,Rpd	4	13	88	iop16 → (Rpd–1:Rpd)
MOVW	Rps,Rpd	3	12	98	(Rps–1:Rps) → (Rpd–1:Rpd)
MOVW	#iop8(Rp),Rpd	5	20	F4 E8	(Rp–1:Rp)+iop8 → (Rpd–1:Rpd)
MOVW	#iop16(B),Rpd	4	15	A8	(B) + iop16 → (Rpd–1:Rpd)

Status Bits Affected

C ← 0
N Set on MSbyte moved
Z Set on MSbyte moved
V ← 0

Description

MOVW moves a two-byte value to the register pair indicated by the destination register number. (Note that Rpd should be greater than 0.) The destination points to the LSbyte of the destination register pair. The source can be a 16-bit constant, another register pair, or an indexed address.

For the indexed address, the source must be of the form "#ADDR(B)" where ADDR is a 16-bit constant or address. This 16-bit value is added (via 16-bit addition) to the contents of register B, and the result is placed in the destination register pair. This stores an indexed address into a register pair for use later in indirect addressing mode. This is not to be confused with the extended addressing instruction LABEL(B).

Examples

```
LABEL   MOVW  #1234h,R3       ;1234h → (R2:R3)
        MOVW  R5,R3           ;(R4:R5) → (R2:R3)
                              ;R5,R3 = LSbyte

        MOVW #TAB(B),R3       ;TAB + (B) → (R2:R3)
                              ;R3 = LSbyte

        MOVW #127(R200),R34   ;127 + (R199:R200) →
                              ;(R33:R34)

        MOVW #-128(R200),R34  ;(R199:R200) - 128 →
                              ;(R33:R34)
```

Syntax **MPY** *s,Rn*

Execution (s) × (Rn) → (A:B)
Result always stored in A,B; A = MSbyte

Options

Inst	operands	bytes	cycles	opcode	operation
MPY	B,A	1	47	6C	(A) X (B) → (A:B)
MPY	Rs,A	2	46	1C	(A) X (Rs) → (A:B)
MPY	Rs,B	2	46	3C	(B) X (Rs) → (A:B)
MPY	Rs,Rd	3	48	4C	(Rd) X (Rs) → (A:B)
MPY	#iop8,A	2	45	2C	(A) X iop8 → (A:B)
MPY	#iop8,B	2	45	5C	(B) X iop8 → (A:B)
MPY	#iop8,Rd	3	47	7C	(Rd) X iop8 → (A:B)

Status Bits Affected
C ← 0
N Set on MSbyte of results (register A)
Z Set on MSbyte of results (register A)
V ← 0

Description MPY performs an 8-bit multiply for a general source and destination operand. The 16-bit result is placed in the A, B register pair with the most significant byte in A. Multiplying by a power of two is a convenient means of performing double-byte shifts.

- ❑ If a double-byte shift is three places or less, then it can be faster to use RLC or RRC instead of multiply.
- ❑ To shift a single byte, it is almost always faster to use RLC or RRC.

Examples

```
LABEL    MPY  R3,A          ;Multiply (R3) with (A), store
                            ;result in A, B register pair

         MPY #032h,B       ;Multiply 32h with (B), store
                            ;in register pair A, B

         MPY  R12,R7       ;Multiply (R12) with (R7) and
                            ;store in A, B register pair
```

Syntax **NOP**

Execution (PC) + 1 → (PC)

Options

inst	operands	bytes	cycles	opcode
NOP	none	1	7	FF

Status Bits Affected None

Description NOP is useful as a pad instruction during program development to "patch out" unwanted or erroneous instructions or to leave room for code changes during development. It is also useful in software timing loops.

Example

```
LABEL    NOP
```

Syntax **OR** *s,Rd*

Execution (s) OR (Rd) → (Rd)

Options

inst	operands	bytes	cycles	opcode	operation
OR	A,Pd	2	9	84	(A) OR (Pd) → (Pd)
OR	B,Pd	2	9	94	(B) OR (Pd) → (Pd)
OR	B,A	1	8	64	(B) OR (A) → (A)
OR	Rs,A	2	7	14	(Rs) OR (A) → (A)
OR	Rs,B	2	7	34	(Rs) OR (B) → (B)
OR	Rs,Rd	3	9	44	(Rs) OR (Rd) → (Rd)
OR	#iop8,A	2	6	24	iop8 OR (A) → (A)
OR	#iop8,B	2	6	54	iop8 OR (B) → (B)
OR	#iop8,Rd	3	8	74	iop8 OR (Rd) → (Rd)
OR	#iop8,Pd	3	10	A4	iop8 OR (Pd) → (Pd)

Status Bits Affected

C ← 0
N Set on result
Z Set on result
V ← 0

Description OR logically ORs the two operands. The OR operation is used to set bits in a register. If a register needs a 1 in the destination, then a 1 is placed in the corresponding bit location in the source operand.

Examples

```
LABEL    OR   A,R12          ;OR register A with R12,
                             ;store in R12

         OR   #00Fh,A        ;Set lower nibble of A to 1s,
                             ;leave upper nibble unchanged

         OR   R8,B           ;OR (R8) with (B), store in B
```

Syntax **POP** *d*

Execution
((SP)) → (d)
(SP) – 1 → (SP)
(Move value then decrement SP)

Options

inst	operands	bytes	cycles	opcode	operation
POP	A	1	9	B9	((SP)) → (A); (SP) – 1 → (SP)
POP	B	1	9	C9	((SP)) → (B); (SP) – 1 → (SP)
POP	Rd	2	7	D9	((SP)) → (Rd); (SP) – 1 → (SP)
POP	ST	1	8	FC	((SP)) → (ST); (SP) – 1 → (SP)

Status Bits Affected
C ← 0
N Set on value POPed
Z Set on value POPed
V ← 0

Note:

POP ST affects all status bits.

Description POP pulls a value from the top of the stack. The stack can be used to save or to pass values between routines. POP ST can replace the status register with the contents on the stack. This one-byte instruction is usually executed in conjunction with a previously performed PUSH ST instruction.

Examples

```
LABEL     POP  R32     ;Load R32 with value on top of stack

          POP ST       ;Load status register with
                       ;value on top of stack
```

Syntax **PUSH** *s*

Execution
(SP) + 1 → (SP)
(s) → ((SP))
(Increment SP then move value)

Options

Inst	operands	bytes	cycles	opcode	operation
PUSH	A	1	9	B8	(SP) + 1 → (SP); (A) → ((SP))
PUSH	B	1	9	C8	(SP) + 1 → (SP); (B) → ((SP))
PUSH	Rs	2	7	D8	(SP) + 1 → (SP); (Rs) → ((SP))
PUSH	ST	1	8	FB	(SP) + 1 → (SP); (ST) → ((SP))

Status Bits Affected
C ← 0
N Set on value PUSHed
Z Set on value PUSHed
V ← 0
Note: Status bits are unchanged for PUSH ST

Description
PUSH places a value on the top of the stack. The stack is used to save or pass values between routines.

The status register can be pushed on the stack with the statement PUSH ST. This one-byte instruction is usually executed in conjunction with a subsequently performed POP ST instruction.

Examples

```
LABEL     PUSH  A          ;Move (A) to top of stack

          PUSH  ST         ;Move status to top of stack
```

Syntax **RL** *Rn*

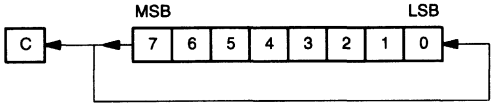
Execution
Bit(n) → Bit(n+1)
Bit(7) → Bit(0) and carry

Options

inst	operands	bytes	cycles	opcode
RL	A	1	8	BE
RL	B	1	8	CE
RL	Rn	2	6	DE

Status Bits Affected
C Set to bit 7 of the original operand
N Set on result
Z Set on result
V ← 0

Description
RL circularly shifts the destination contents one bit to the left. The MSB is shifted into the LSB; the carry bit is also set to the original MSB value.



For example, if register B contains the value 93h, then RL changes the contents of B to 27h and sets the carry bit.

Examples

```
LABEL    RL R102

         RL  A

         RL  B
```

Syntax **RLC** *Rn*

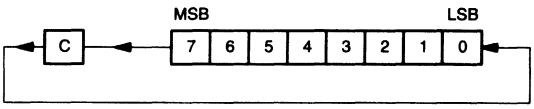
Execution
Bit(n) → Bit(n+1)
Carry → Bit(0)
Bit(7) → Carry

Options

inst	operands	bytes	cycles	opcode
RLC	A	1	8	BF
RLC	B	1	8	CF
RLC	Rn	2	6	DF

Status Bits Affected
C Set to bit 7 of the original operand
N Set on result
Z Set on result
V ← 0

Description
RLC circularly shifts the destination contents one bit to the left and through the carry. The original carry bit contents shift into the LSB, and the original MSB shifts into the carry bit.



For example, if register B contains the value 93h and the carry bit is a zero, then the RLC instruction changes the operand value to 26h and the carry to one.

Rotating left effectively multiplies the value by 2. Using multiple rotates, any power of 2 (2, 4, 8, 16,...) can be achieved. This type of multiply can be faster than the MPY (multiply) instruction. This instruction is also useful in rotates where a value is contained in more than one byte (such as an address) or in multiplying a large multibyte number by 2. Take care to assure that the carry is at the proper value. The SETC or CLRC instructions can be used to set up the correct value.

Examples

```
LABEL    RLC  R72

         RLC  A

         RLC  B
```

Syntax **RR** *Rn*

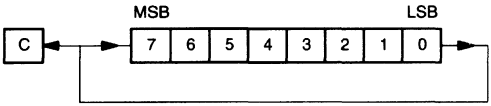
Execution Bit(n+1) → Bit(n)
Bit(0) → Bit (7) and carry

Options

inst	operands	bytes	cycles	opcode
RR	A	1	8	BC
RR	B	1	8	CC
RR	Rn	2	6	DC

Status Bits Affected
C Set to bit 0 of the original value
N Set on result
Z Set on result
V ← 0

Description RR circularly shifts the destination contents one bit to the right. The LSB is shifted into the MSB, and the carry bit is also set to the original LSB value.



For example, if register B contains the value 93h, then the RR B instruction changes the contents of B to C9h and sets the carry status bit.

Example

```
LABEL    RR  A
```

Syntax **RRC** *Rn*

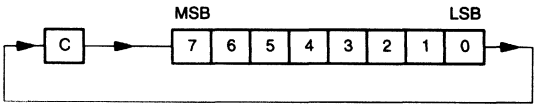
Execution
Bit(n+1) → Bit(n)
Carry → Bit(7)
Bit(0) → Carry

Options

inst	operands	bytes	cycles	opcode
RRC	A	1	8	BD
RRC	B	1	8	CD
RRC	Rn	2	6	DD

Status Bits Affected
C Set to bit 0 of the original value
N Set on result
Z Set on result
V ← 0

Description
RRC circularly shifts the destination contents one bit to the right through the carry. The carry bit contents shift into the MSB, and the LSB shifts into the carry bit.



For example, if register B contains the value 93h and the carry bit is zero, then RRC changes the operand value to 49h and sets the carry bit.

When the carry bit is 0, this instruction effectively divides the value by two. A value of 80h becomes 40h. Repetitive use of this instruction can divide the value by any power of two. Care must be taken to assure the correct value in the carry bit.

Example
```
LABEL     RRC R32
```

Syntax **RTI**

Execution

((SP)) → (PC LSbyte)
(SP) – 1 → (SP)
((SP)) → (PC MSbyte)
(SP) – 1 → (SP)
((SP)) → (ST)
(SP) – 1 → (SP)

Options

inst	operands	bytes	cycles	opcode
RTI	none	1	12	FA

Status Bits Affected Status register is loaded from the stack

Description RTI is typically the last instruction executed in an interrupt service routine. RTI restores the status register to the state it was in immediately before the interrupt occurred and branches back to the program at the instruction boundary where the interrupt occurred. In an interrupt routine, there must be an equal number of POPs and PUSHs so that the stack is pointing to the correct return address and not some other data.

Example

```
LABEL RTI     ;Return to main program from interrupt routine
```

Syntax **RTS**

Execution

((SP)) → (PC LSbyte)
(SP) – 1 → (SP)
((SP)) → (PC MSbyte)
(SP) – 1 → (SP)

Options

inst	operands	bytes	cycles	opcode
RTS	none	1	9	F9

Status Bits Affected None

Description RTS is typically the last instruction executed in a subroutine. RTS branches to the location immediately following the subroutine call instruction. In the called subroutine, there must be an equal number of POPs and PUSHes so that the stack is pointing to the return address and not some other data.

Example

```
LABEL RTS      ;Return to main program from subroutine
```

Syntax **SBB** *s,Rd*

Execution (Rd) – (s) – 1 + (C) → (Rd)

Options

inst	operands	bytes	cycles	opcode	operation
SBB	B,A	1	8	6B	(A) – (B) – 1 + (C) → (A)
SBB	Rs,A	2	7	1B	(A) – (Rs) – 1 + (C) → (A)
SBB	Rs,B	2	7	3B	(B) – (Rs) – 1 + (C) → (B)
SBB	Rs,Rd	3	9	4B	(Rd) – (Rs) – 1 + (C) → (Rd)
SBB	#iop8,A	2	6	2B	(A) – iop8 – 1 + (C) → (A)
SBB	#iop8,B	2	6	5B	(B) – iop8 – 1 + (C) → (B)
SBB	#iop8,Rd	3	8	7B	(Rd) – iop8 – 1 + (C) → (Rd)

Status Bits Affected

C Set to 1 if no borrow; 0 otherwise
N Set on result
Z Set on result
V ((C XOR N) AND (Source[Bit 7] XOR Destination[Bit 7]))

Description SBB performs multibyte 2s-complement subtraction. An SBB instruction with an immediate operand of zero value is equivalent to a conditional decrement of the destination operand, depending on the carry value. If (s) = 0 and (C) = 0, then (Rd) is decremented. A borrow occurs if the result is negative. In this case, the carry bit is set to 0. The carry bit acts as the no-borrow bit.

Examples

```
LABEL    SBB   #023h,B    ;Subtract 23h from (B), sub-
                          ;tract 1, add the carry bit
                          ;and store in register B

         SUB   R3,R21     ;R20:R21 and R2:R3 contain 1
         SBB   R2,R20     ;bit numbers. SUB subtracts
                          ;the LSbyte, and the SBB will
                          ;use the carry as a borrow
                          ;during the subtract of
                          ;the MSbyte.
```

Syntax **SBIT0** *name*

Execution 0 → <*name*>

Options

inst	operands	bytes	cycles	opcode	operation
SBIT0	Rname	3	8	73	0 → <bit> Register bits
SBIT0	Pname	3	10	A3	0 → <bit> Peripheral bits

Status Bits Affected

C ← 0
N Set on result
Z Set on result
V ← 0

Description SBIT0 is an assembler-constructed instruction that conveniently clears the value of the named bit without having to specify a register or mask. This enhances the readability of the program. This instruction assembles to the instructions AND #iop8,Rd or AND #iop8,Pd. The name for the bit is defined by the .DBIT assembler directive.

Examples

```
INT1ENA   .DBIT 7,P01C      ;The interrupt 1 enable
                            ;bit is now named INT1ENA

TEST      .DBIT 4,R33       ;Bit 4 of register 33
                            ;is now named TEST

LABEL     SBIT0 TEST        ;Clears the value of the
                            ;TEST bit

          SBIT0 INT1ENA     ;Disables interrupt 1
```

Syntax **SBIT1** *name*

Execution 1 → <name>

Option

inst	operands	bytes	cycles	opcode	operation
SBIT1	Rname	3	8	74	1 → <bit> Register bits
SBIT1	Pname	3	10	A4	1 → <bit> Peripheral bits

Status Bits Affected

C ← 0
N Set on result
Z Set on result
V ← 0

Description SBIT1 is an assembler-constructed instruction that conveniently sets the value of the named bit without having to specify a register or mask. This enhances the readability of the program. This instruction assembles to the instructions OR #iop8,Rd or OR #iop8,Pd. The name for the bit is defined by the .DBIT assembler directive.

Examples

```
INT1ENA   .DBIT 7,P01C      ;The interrupt 1 enable bit
                            ;is now named INT1ENA

TEST      .DBIT 4,R33       ;Bit 4 of register 33 is now
                            ;named TEST

LABEL     SBIT1 TEST        ;Sets the value of the TEST
                            ;bit to 1

          SBIT1 INT1ENA     ;Enables interrupt 1
```

Syntax **SETC**

Execution 1 → (C)

Options

inst	operands	bytes	cycles	opcode
SETC	none	1	7	F8

Status Bits Affected

C ← 1
N ← 0
Z ← 1
V ← 0

Description SETC sets the carry flag. This instruction can be used before an arithmetic or rotate instruction. The IE1 and IE2 enable bits are not affected.

Example

```
LABEL     SETC        ;Set the carry bit in the status
                      ;register
                      ;Status register = 0Axh
```

Syntax **STSP**

Execution (SP) → (B)

Options

inst	operands	bytes	cycles	opcode
STSP	none	1	8	FE

Status Bits Affected None

Description STSP copies the contents of the stack pointer to register B. This instruction can test the stack size. The indexed addressing mode can reference operands on the stack after executing this instruction.

Example

```
LABEL     STSP        ;Copy the contents of stack pointer
                      ;to register B
```

Syntax **SUB** *s,Rd*

Execution (Rd) – (s) → (Rd)

Options

inst	operands	bytes	cycles	opcode	operation
SUB	B,A	1	8	6A	(A) – (B) → (A)
SUB	Rs,A	2	7	1A	(A) – (Rs) → (A)
SUB	Rs,B	2	7	3A	(B) – (Rs) → (B)
SUB	Rs,Rd	3	9	4A	(Rd) – (Rs) → (Rd)
SUB	#iop8,A	2	6	2A	(A) – iop8 → (A)
SUB	#iop8,B	2	6	5A	(B) – iop8 → (B)
SUB	#iop8,Rd	3	8	7A	(Rd) – iop8 → (Rd)

Status Bits Affected

C Set to 1 if no borrow, otherwise set to 0
N Set on result
Z Set on result
V ((C XOR N) AND (Source[Bit 7] XOR Destination[Bit 7]))

Description SUB performs 2s-complement subtraction. The carry bit is set to 0 if a borrow is required. The carry bit acts as a no-borrow bit in this case.

Examples

```
LABEL   SUB  R19,B        ;(B) minus (R19) is
                         ;stored in B

        SUB  076h,A       ;(A) minus 076h is stored
                         ;in A

        SUB  R4,R9        ;(R9) minus (R4) is stored
                         ;in R9
```

Syntax **SWAP** *Rn*

Execution Bits (7,6,5,4, / 3,2,1,0) → Bits (3,2,1,0, / 7,6,5,4)

Options

inst	operands	bytes	cycles	opcode
SWAP	A	1	11	B7
SWAP	B	1	11	C7
SWAP	Rn	2	9	D7

Status Bits Affected

C Set to bit 4 of original register or bit 0 of result register
N Set on results
Z Set on results
V ← 0

Description SWAP exchanges the first four bits with the second four bits. This instruction is equivalent to four consecutive RL (rotate left) instructions. It is especially useful for packed BCD operations.

Examples

```
LABEL    SWAP R45    ;Switch Low and High nibbles of R45

         SWAP  A     ;Switch Low and High nibbles of A

         SWAP  B     ;Switch Low and High nibbles of B
```

Syntax **TRAP** *n* where n = 0 thru 15

Execution

(SP) + 1 → (SP)
(PC MSbyte) → ((SP))
(SP) + 1 → (SP)
(PC LSbyte) → ((SP))
(Entry vector) → (PC)

Options

					entry-vector	
inst	**operands**	**bytes**	**cycles**	**opcode**	**MSbyte**	**LSbyte**
TRAP	0	1	14	EF	7FDEh	7FDFh
TRAP	1	1	14	EE	7FDCh	7FDDh
TRAP	2	1	14	ED	7FDAh	7FDBh
TRAP	3	1	14	EC	7FD8h	7FD9h
TRAP	4	1	14	EB	7FD6h	7FD7h
TRAP	5	1	14	EA	7FD4h	7FD5h
TRAP	6	1	14	E9	7FD2h	7FD3h
TRAP	7	1	14	E8	7FD0h	7FD1h
TRAP	8	1	14	E7	7FCEh	7FCFh
TRAP	9	1	14	E6	7FCCh	7FCDh
TRAP	10	1	14	E5	7FCAh	7FCBh
TRAP	11	1	14	E4	7FC8h	7FC9h
TRAP	12	1	14	E3	7FC6h	7FC7h
TRAP	13	1	14	E2	7FC4h	7FC5h
TRAP	14	1	14	E1	7FC2h	7FC3h
TRAP	15	1	14	E0	7FC0h	7FC1h

Status Bits Affected None

Description Trap is a one-byte subroutine call. The operand <n> is a trap number that identifies a location in the trap vector table (addresses 07FC0h to 07FDFh in memory). The contents of the two-byte vector location form a 16-bit trap vector to which a subroutine call is performed. When you invoke the same routine more than once,the TRAP is more efficient than a CALL because fewer bytes are needed. The subroutine addresses are stored like all other addresses in memory, with the least significant byte in the higher-addressed location, as indicated above.

Example

```
LABEL     TRAP 0              ;Execute subroutine at TRAPONE

.sect trap,07FC0h             ;Define section starting
                              ;at 7FC0h

.word TRAP15,TRAP14           ;Define TRAPS 15 AND 14
                              ;subroutine entry points
```

Syntax **TST** {**A** | **B**}

Execution C,N,Z,V bits affected

Options

inst	operands	bytes	cycles	opcode
TST	**A**	1	9	B0
TST	**B**	1	10	C6

Status Bits Affected

C ← 0
N Set or cleared based on operand
Z Set or cleared based on operand
V ← 0

Description TST sets the status bits according to the value in register A or B. This allows conditional jumps on the value in the register.

Example

```
LABEL     TST A       ;Check for zero and negative
                      ;conditions in register A

          TST  B      ;Check for zero and negative
                      ;conditions in register B
```

Syntax **XCHB** *Rd*

Execution (B) ←→ (Rd)

Options

inst	operands	bytes	cycles	opcode	operation
XCHB	A	1	10	B6	(A) ← → (B)
XCHB	B	1	10	C6	(B) ← → (B) (TST B)
XCHB	Rd	2	8	D6	(Rd) ← → (B)

Status Bits Affected

C ← 0
N Set on original contents of B
Z Set on original contents of B
V ← 0

Description XCHB exchanges a register with register B without going through an intermediate location. The XCHB instruction with register B as the operand is equivalent to the TST B instruction.

Examples

```
LABEL     XCHB  A      ;Exchange register B with
                       ;register A

          XCHB  R3     ;Exchange register B with R3
```

13

Syntax **XOR** *s,d*

Execution (s) XOR (d) → (d)

Options

inst	operands	bytes	cycles	opcode	operation
XOR	A,Pd	2	9	85	(A) XOR (Pd) → (Pd)
XOR	B,A	1	8	65	(B) XOR (A) → (A)
XOR	B,Pd	2	9	95	(B) XOR (Pd) → (Pd)
XOR	Rs,A	2	7	15	(Rs) XOR (A) → (A)
XOR	Rs,B	2	7	35	(Rs) XOR (B) → (B)
XOR	Rs,Rd	3	9	45	(Rs) XOR (Rd) → (Rd)
XOR	#iop8,A	2	6	25	iop8 XOR (A) → (A)
XOR	#iop8,B	2	6	55	iop8 XOR (B) → (B)
XOR	#iop8,Rd	3	8	75	iop8 XOR (Rd) → (Rd)
XOR	#iop8,Pd	3	10	A5	iop8 XOR (Pd) → (Pd)

Status Bits Affected

C ← 0
N Set on result
Z Set on result
V ← 0

Description XOR performs a bit-wise exclusive OR operation on the operands. The XOR instruction can complement bits in the destination operand. This operation can also toggle a bit in a register. If the bit value in the destination must be the opposite from what it currently is, then the source should contain a 1 in that bit location.

Examples

```
LABEL   XOR  R98,R125     ;XOR (R98) with (R125),
                          ;store in R125

        XOR  #01,R20      ;Toggle bit 0 in R20

        XOR  B,A          ;XOR (B) with (A), store
                          ;in register A
```

Chapter 14

Design Aids

To help you in developing your system, this chapter contains sample TMS370 applications and covers the following topics:

Topic **Page**

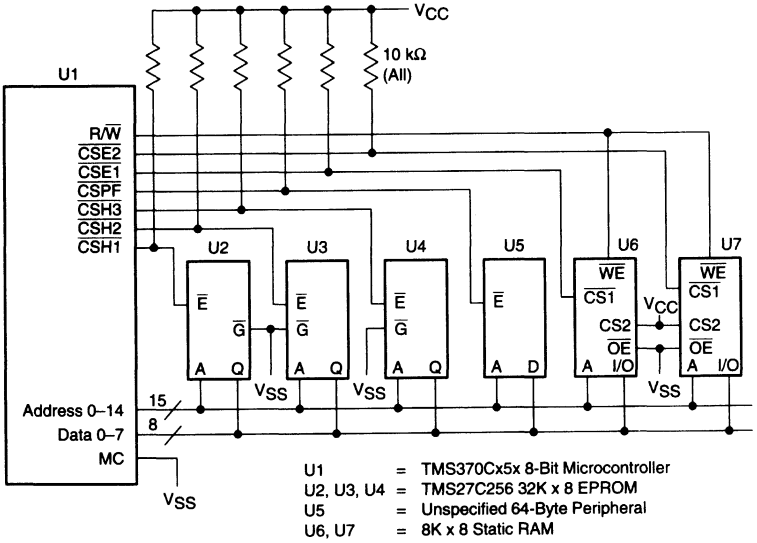
14.1 Microcomputer Interface Example

The following exercise is one method of interfacing the TMS370 family with common memory. The goals of this example are to:

- Interface with the maximum amount of memory,
- Use the least expensive logic elements,
- Use a minimum amount of parts, and
- Maintain sufficient system speed.

The example shown in Figure 14–1 illustrates a balance of these goals. In this case, the TMS370C050 is used with three TMS27C256s to provide 96K bytes of EPROM and two HM626LP-15s to give 16K bytes of RAM. Peripheral devices using up to 64 bytes of memory space can also interface to the memory, giving a total memory of 116K bytes; 112K bytes of external memory and 4K bytes of memory internal to the microcomputer. The current timings for the EPROM and RAM memory devices are given. Since specifications change from time to time, always check the latest data sheets for the devices used.

Figure 14–1. Microcomputer Interface Example



The devices used in the TMS370 interface example circuit are:

- TMS370C050—8-bit CMOS microcontroller
- TMS27C256—32K x 8 EPROM
- HM626LP—Hitachi 8K x 8 RAM

The timing specifications for the TMS27C256-25 EEPROM devices are as follows:

Symbol	Description	Min	Max
$t_{a(A)}$	Access time from address	—	250 ns
$t_{a(E)}$	Access time from enable	—	250 ns
t_{dis}	Output disable time	0 ns	60 ns
$t_{v(A)}$	Output data valid after address change	0 ns	—

Reference: 1993 TI *MOS Memory Data Book*

The timing specifications for the HM6264P-15 RAM device are as follows:

Symbol	Description	Min	Max
t_{AA}	Address access time	—	150 ns
t_{OHZ}	Out disable to output in high Z	0	
t_{C01}	Chip selection to output	—	150 ns
t_{HZ1}	Chip deselection to output in high Z	0 ns	50 ns
t_{CW}	Chip select to end of write	100 ns	—
t_{WP}	Write pulse width	90 ns	—
t_{DW}	Data to write time overlap	60 ns	—
t_{DH}	Data hold from write time	0 ns	—

Reference: *#M10 Hitachi Memory Data Book*

The TMS370 family is designed to use a clock speed of 20 MHz, so slower peripheral devices may not be able to react quickly enough to operate properly. The TMS370 family of devices have the ability to insert wait states to slow the memory accesses in three different ways.

- ❑ Use the AUTOWAIT DISABLE bit at SCCR1.4 to add one wait state to all external accesses.
- ❑ Use the PF AUTOWAIT bit at SCCR0.5 to add two wait states to the external peripheral file access.
- ❑ Allow the external device to pull the $\overline{\text{WAIT}}$ pin low and add as many wait states as required.

Table 14–1 shows the various combinations.

Table 14–1. Wait-State Control Bits

Wait-State Control Bits		Number of Clock Cycles per Access	
PF AUTOWAIT	**AUTOWAIT DISABLE**	**Peripheral File**	**External Memory**
0	0	3	3
0	1	2	2
1	0	4	3
1	1	4	2

The following subsections discuss the signal timings that must be considered for interfacing the TMS370 with external memory. With each system design, there are usually trade-offs due to speed and/or budget constraints. The timings given in Table 14–2 reflect worst-case specifications, and typical values have been avoided where possible.

Table 14–2. Memory Interface Timing

	Parameter	Min	Max	Unit
t_c	CLKOUT (system clock) cycle time	200	2000	ns
$t_{w(COL)}$	CLKOUT low pulse duration	0.5 t_c – 25	0.5 t_c	ns
$t_{w(COH)}$	CLKOUT high pulse duration	0.5 t_c	0.5t_c + 20	ns
$t_{d(COL-A)}$	Delay time, CLKOUT low to address R/$\overline{W}$, and $\overline{OCF}$ valid		0.25t_c + 75	ns
$t_{v(A)}$	Address valid to $\overline{EDS}$, $\overline{CSE1}$, $\overline{CSE2}$, $\overline{CSH1}$, $\overline{CSH2}$, $\overline{CSH3}$, and $\overline{CSPF}$ low	0.5 t_c – 90		ns
$t_{su(D)}$	Write data set-up time to $\overline{EDS}$ high	0.75 t_c – 80‡		ns
$t_{h(EH-A)}$	Address, R/$\overline{W}$, and $\overline{OCF}$ hold time from $\overline{EDS}$,$\overline{CSE1}$, $\overline{CSE2}$, $\overline{CSH1}$, $\overline{CSH2}$, $\overline{SH3}$, and $\overline{CSPF}$ high	0.5 t_c – 60		ns
$t_{h(Eh-D)W}$	Write data hold time from $\overline{EDS}$ high	0.75 t_c + 15		ns
$t_{d(DZ-EL)}$	Delay time, data bus high impedance to $\overline{EDS}$ low (read cycle)	0.25 t_c – 35		ns
$t_{d(EH-D)}$	Delay time, $\overline{EDS}$ high to data bus enable (read cycle)	1.25 t_c – 40		ns
$t_{d(EL-DV)}$	Delay time, $\overline{EDS}$ low to read data valid		t_c – 95‡	ns
$t_{h(EH-D)R}$	Read data hold time from $\overline{EDS}$ high	0		ns
$t_{su(WT-COH)}$	$\overline{WAIT}$ set-up time to CLKOUT high	0.25 t_c + 70§		ns
$t_{h(COH-WT)}$	$\overline{WAIT}$ hold time from CLKOUT high	0		ns
$t_{d(ED-WTV)}$	Delay time, $\overline{EDS}$ low to $\overline{WAIT}$ valid		0.5 t_c – 60	ns
t_w	Pulse duration, $\overline{EDS}$, $\overline{CSE1}$, $\overline{CSE2}$, $\overline{CSH1}$, $\overline{CSH2}$, $\overline{CSH3}$, and $\overline{CSPF}$ low	t_c – 80‡	t_c + 40‡	ns
$t_{d(AV-DV)R}$	Delay time, address valid to read data valid		1.5 t_c – 115‡	ns
$t_{d(AV-WTV)}$	Delay time, address valid to $\overline{WAIT}$ valid		t_c – 115	ns
$t_{d(AV-EH)}$	Delay time, address valid to $\overline{EDS}$ high (end of write)	1.5 t_c – 85‡		ns

† t_c = system clock cycle time = 4/CLKIN.

‡ If wait states, PFWait, or the autowait feature is used, add t_c to this value for each wait state invoked.

§ If the autowait feature is enabled, the $\overline{WAIT}$ input may assume a "don't care" condition until the third cycle of the access. The $\overline{WAIT}$ signal must be synchronized with the high pulse of the CLKOUT signal while still conforming to the minimum set-up time.

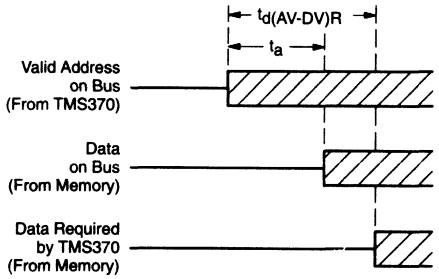
14.1.1 Read Cycle Timing

Interfacing the TMS370 with external memory devices requires a minimum amount of address-to-data access time, depending on the CPU clock speed and the number of wait states used. If the requirements are not met, incorrect data may be read. The requirements in this section are based on a 20-MHz clock frequency.

14.1.1.1 Valid Address-to-Data Read Time Requirement

The external device must meet the basic read cycle requirement: the valid address to data read time. This is the period from the instant the TMS370 outputs a valid address until the TMS370 requires data on the data memory pins. You can vary this requirement by using wait states to delay the moment the TMS370 reads data.

Figure 14–2. Valid Address-to-Data Read Timing



Name	Description	Formula	Time
$t_{d(AV-DV)R}$	TMS370 (0 wait) requires data	1.5 t_c–115	185 ns(too fast)
$t_{d(AV-DV)R}$	TMS370 (1 wait) requires data	2.5 t_c–115	385 ns(ok)
$t_{d(AV-DV)R}$	TMS370 (PF wait) requires data	3.5 t_c–115	585 ns(ok)
$t_{a(A)}$	TMS27C256-25 provides data		250 ns(ok)
t_{AA}	HM6264-15 provides data		150 ns(ok)

As indicated above, the EPROM (TMS27C256) cannot provide the data quickly enough when the TMS370 device runs at full speed (zero wait states). Therefore, the TMS370 device should use the autowait feature (SCCR1.4) to add a wait state (one clock cycle) to the timing in order to slow the bus accesses. The wait state extends the access time (data required by TMS370) to 385 ns; then, the EPROM is ready with the data. The autowait feature makes it possible to use the TMS370 in low-cost applications with cheaper, slower memory devices.

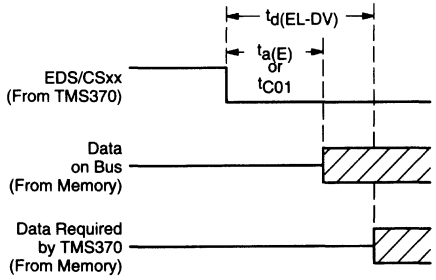
The HM6264-15 RAM can extend the TMS370's minimum address-to-data set-up time with no wait states. When you access external RAM comparable to the Hitachi device, you can turn off the autowait feature to speed up the system.

A peripheral device can have up to 585 ns to respond to the TMS370 if the PF wait states are enabled. If the extra wait states are not needed, the TMS370 treats the peripheral device like other memory.

14.1.1.2 Chip-Select Low-to-Data Read Requirements

This parameter states the amount of delay from the time the chip-select signal goes low to the time the TMS370 expects valid data on the bus. The chip-select signal ($\overline{CSxx}$ or $\overline{EDS}$) must be used with external memory to validate the memory cycle. Connecting the chip-select pin ($\overline{CSxx}$) of the TMS370 to the EPROM's enable pin ($\overline{E}$) enables the EPROM to enter the low-power standby mode when not providing data. This significantly lowers the power requirements for the system because only one EPROM operates in the full-power operation mode at any one time. The HM6264 also enters a low-power standby mode whenever the $\overline{CS1}$ pin is pulled high.

Figure 14–3. Chip-Select Low-to-Data Read Timing

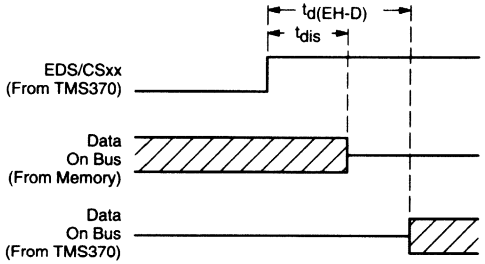


Name	Description	Formula	Time
$t_{d(EL-DV)}$	TMS370 (0 wait) requires data	t_c–95	105 ns(too fast)
$t_{d(EL-DV)}$	TMS370 (1 wait) requires data	2 t_c–95	305 ns(ok)
$t_{d(EL-DV)}$	TMS370 (pf wait) requires data	3 t_c–95	505 ns(ok)
$t_{a(E)}$	TMS27C256-25 provides data		250 ns(ok)
t_{C01}	HM6264-15 provides data		150 ns(ok)

14.1.1.3 Chip-Select High-to-Next Data Bus Drive Requirements

The TMS370 and the memory device should not drive the memory at the same time. This can lead to increased stress and noise spiking on the V_{CC} and V_{SS} lines and reduce the reliability of the device. Memory devices often continue to drive the memory for a short time after the chip-select signal goes high. This normally doesn't present a problem unless the chip-select signal is delayed by interface circuitry and the data is not delayed. If the chip-select high transition is delayed long enough (and the data is not), the TMS370 will initiate a write cycle while the memory is still providing data.

Figure 14–4. Chip-Select High-to-Next Data Bus Drive Timing

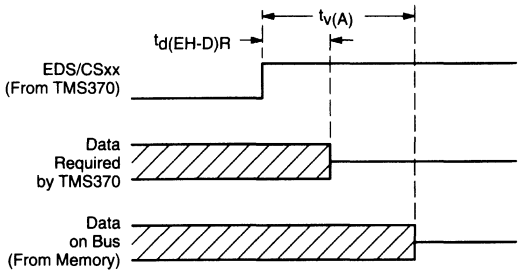


Name	Description	Formula	Time
$t_{d(EH-D)}$	TMS370 (all) drives memory	$1.2\ 5t_c-40$	210 ns
t_{dis}	TMS27C256-25 releases memory		60 ns
t_{OHZ}	HM6264-15 releases memory		50 ns

14.1.1.4 Read Data Hold After Chip Select High Requirements

The high transition of the chip-select signal ($\overline{CHxx}$) indicates the end of a data transfer (in this case, a read) cycle. The memory device must provide data up to this point, or incorrect data may be read. Most memories will continue to hold (or drive) the data memory for a short time after they are deselected, although the data may or may not be valid. After that period, the memories put their data outputs into the high-impedance state.

Figure 14–5. Read Data Hold After Chip-Select High Timing



Name	Description	Formula	Time
$t_{d(EH-D)R}$	TMS370 (all) needs data	—	0 ns
$t_{V(A)}$	TMS27C256-25 data	—	0 ns
t_{HZ}1	HM6264-15 holds data	—	0 ns

14.1.2 Write Cycle Timing

The write cycle timing is defined primarily by the characteristics of the RAM interfacing with the TMS370. The Hitachi HM6264 used in this example offers two types of write cycles. This application uses a write cycle in which the output enable pin ($\overline{OE}$) is always fixed low. With the $\overline{CS2}$ pin tied to V_{CC}, the $\overline{CS1}$ and R/$\overline{W}$ signals determine the read and write cycle boundaries. You can use a separate address decoder instead of the chip-select functions, but you must use the $\overline{EDS}$ to validate the memory cycle. The $\overline{EDS}$ signal has the same timing as the chip-select signals. Figure 14–6 shows the write cycle parameters that must be met; they are discussed in the paragraphs that follow.

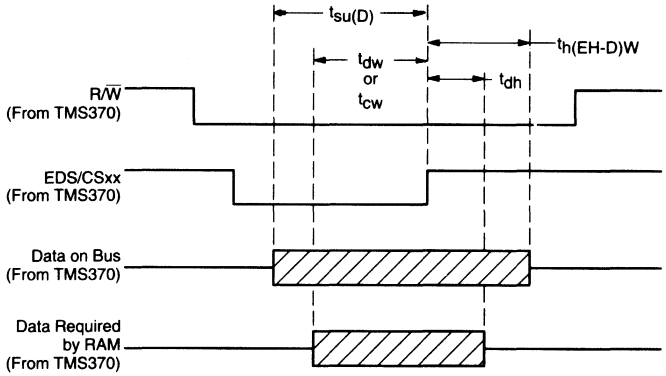
Name	Description	Formula	Time
t_W	TMS370 (no wait) pulse width provided	t_c–80	120 ns
t_W	TMS370 (PF wait) pulse width provided	3 t_c–80	520 ns
t_{CW}	HM6264-15 pulse width required		100 ns

14.1.2.1 Write Data Set-Up Time Requirements

The write data set-up time is the period the RAM needs to receive data before the chip select signal goes high (inactive).

Name	Description	Formula	Time
$t_{SU(D)}$	TMS370 (no wait) provides data	0.75 t_c–80	70 ns
$t_{SU(D)}$	TMS370 (PF wait) provides data	2.75 t_c–80	470 ns
t_{DW}	HM6264-15 requires data		60 ns

Figure 14–6. Write Data Set-Up Timing

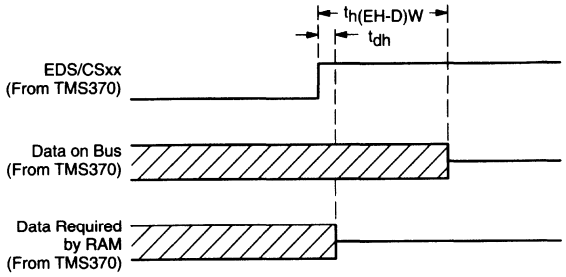


In the interface example, the TMS370 satisfies the HM6264-15 RAM's set-up requirement, even with no wait state. However, in a system design with added memory transceivers, set-up timing becomes more important.

14.1.2.2 Data Hold After Chip-Select High

The TMS370 must hold valid data on the bus until the RAM no longer needs it; otherwise, incorrect data may be written into the RAM. Most RAMs do not need data present on the pins following the chip-select's high transition. The TMS370 generally holds data much longer than required by most RAMs.

Figure 14–7. Write Data Hold After Chip-Select High



Name	Description	Formula	Time
$t_{h(EH-D)W}$	TMS370 (all) provides data	0.75 t_c+15	165 ns
t_{DH}	HM6264-15 requires data		0 ns

14.1.3 Design Options

The interface example illustrated in Figure 14–1 on page 14-2 shows a compromise of system speed and cost. This section suggests ways to establish design goals that will optimize your system performance.

14.1.3.1 Lower Cost

If system cost is more important, use slower memories that are less expensive. The slowest TMS27C256-25 EPROM has an access time of 250 ns.

- Access time from address to valid data (@ 20 MHz, t_c=200)

TMS370 (1 wait) requires data	$t_{D(AV-DV)R}$	2.5 t_c–115	385 ns
TMS27C256-25 provides data	$t_{A(A)}$		250 ns (ok)

- Access time from enable low to valid data (@ 20 MHz, t_c=200)

TMS370 (1 wait) requires	$t_{d(EL-DV)}$	2 t_c–95	305 ns
TMS27C256-25 provides data	$t_{A(E)}$	$\overline{E}$ pin	250 ns(ok)
TMS27C256-25 provides data	$t_{EN(G)}$	$\overline{G}$ pin	100 ns(ok)

14.1.3.2 Faster Speed

If the main objective is system speed, then you should use the slowest EPROM that will work with the TMS370 running without wait states. The TMS370 at 20 MHz has a read access time requirement of 185 ns. Therefore, use the TMS27C256-17 EPROM that provides data in 170 ns.

As in the low-cost suggestions above, the EPROM's $\overline{E}$ pin is not fast enough to use the chip-select strobe; use the EPROM's $\overline{G}$ pin instead. To get a low-power standby mode with the EPROMs, use general-purpose output lines from the TMS370 to the EPROM's $\overline{E}$ pin. The pins should be software enabled before the EPROM's program is entered.

- Access time from address to valid data:

TMS370 (no wait) requires data	$t_{D(AV-DV)R}$	1.5 t_c–115	185 ns
TMS27C256-17 provides data	$t_{A(A)}$		170 ns (ok)

- Access time from enable low to valid data:

TMS370 (no wait) requires	$t_{D(EL-DV)}$	t_c–95	105 ns
TMS27C256-17 provides data	$t_{A(E)}$	$\overline{E}$ pin	170 ns (not ok)
TMS27C256-17 provides data	$t_{EN(G)}$	$\overline{G}$ pin	75 ns (ok)

14.1.4 Bank Switching Examples

The programs in this section show how memory bank switching can be used by the circuit in Figure 14–1 (page 14-2). Memory bank switching allows two or more memory devices to share the same addresses. The programmable chip-select signals ($\overline{\text{CSHx}}$, $\overline{\text{CSEx}}$, and $\overline{\text{CSPF}}$) enable the memory devices or banks one at a time during a read or write cycle.

In the interface example in Figure 14–1, the three EPROM devices share addresses 8000h though FFFFh. Only one EPROM device (or bank) at a time, selected by $\overline{\text{CSH1}}$, $\overline{\text{CSH2}}$, or $\overline{\text{CSH3}}$, reads data. The two RAM devices are each mapped at addresses 2000h through 3FFFh. The write and read cycles use one RAM device at a time as determined by the $\overline{\text{CSE1}}$ and $\overline{\text{CSE2}}$ signals. The $\overline{\text{CSPF}}$ signal controls the peripheral memory device, which, in our example, is unspecified but defined to contain 64 bytes of memory. This device is mapped at addresses 10C0h through 10FFh.

To use external memory, devices with memory expansion must be configured for the microcomputer mode so that the chip-select signals are available. The external memory devices must have 3-state outputs because these devices share the data bus.

14.1.4.1 Initializing to EPROM/RAM Bank 1

This program initializes the ports to use bank 1 of the EPROM and the RAM as in Figure 14–1. The TMS370 must be in the microcomputer mode because the chip selects are not available in the microprocessor mode. After an external reset, the TMS370 executes from the internal memory.

```
PORTI   OR    #020h,P010      ;Enable peripheral file
                              ;autowait cycles
        AND   #0EFh,P011      ;Enable general memory wait
                              ;cycles (default condition
                              ;after reset)
        MOV   #0FFh,P021      ;Set port A up as a data memory
        MOV   #0FFh,P025      ;Set port B up as the low
                              ;address memory
        MOV   #07Fh,P029      ;Set port C 0-6 up as the High
                              ;address memory
        MOV   #000h,P02B      ;C7 is not needed for address
                              ;so make it a
                              ;general-purpose input.
        MOV   #000h,P02C      ;
        MOV   #0E7h,P02E      ;Set all CSxx to 1 when CSxx
                              ;are outputs
        MOV   #0D0h,P02D      ;Enable CSH1, CSE1, and
                              ;R/W functions.
        MOV   #0E7h,P02F      ;Turn all chip selects to outputs.
                              ;Pull-ups resistors are important
                              ;for power-up since CSxx are high-
                              ;impedance floating inputs.
```

14.1.4.2 Changing to EPROM Bank 2

This program illustrates how to change the EPROM bank without affecting the RAM banks. In this example, the program runs out of internal memory, disables all EPROM banks, and then enables EPROM bank 2. For this reason, the program must not reside in an EPROM. In order to verify that EPROM bank 2 exists within the system, the program could test various EPROM bank 2 memory locations before executing the branch instruction.

```
AND     #0B9h,P02D      ;Disable all EPROM banks (cannot
                        ;be done while executing from EPROM banks.)
OR      #004h,P02D      ;Enable EPROM bank 2. When turned off,
                        ;pin outputs a 1 because of the
BR      ROM2            ;initial set-up above, could be done
                        ;in 1 instruction if conditions of
                        ;other chip selects were known.
```

14.1.4.3 Changing to EPROM Bank 3 and RAM Bank 2

This routine provides switching from one EPROM bank to another while operating from an EPROM bank. Only one instruction in EPROM bank 2 is needed. The code within the EPROM banks must be synchronized, and the instruction at the address after the move instruction must be a valid instruction within the new EPROM bank.

```
GOROM3  MOV #003h,P02D  ;Enable ROM bank 3 and RAM bank 2.
ROM3                    ;This address must be the same
                        ;as the beginning routine address
                        ;in bank 3 if executing from EPROM.
```

14.1.4.4 Changing RAM Banks

This method demonstrates how to change RAM banks without affecting the execution from the current EPROM bank. The RAM banks are selected and deselected in the same manner as the EPROM banks. When you change RAM or EPROM banks, the software must ensure that only one bank is selected at a time. This example disables the $\overline{\text{CSE1}}$ and $\overline{\text{CSE2}}$ signals and enables the $\overline{\text{CSE2}}$ signal.

```
AND     #07Eh,P02D      ;Turn off all RAM banks (execute
                        ;from EPROM or on chip)
OR      #001h,P02D      ;Turn on RAM bank 2. When turned off,
                        ;pin outputs a 1 because of the
                        ;initial set-up above.
```

14.2 Programming With the TMS370 Family

The following example demonstrates the self-programming ability of the TMS370 family. This feature can program any byte of the onboard data EEPROM by passing the appropriate data and address to this routine.

The program consists of two major sections: the procedure that determines the bits that need to be changed (PROGRAM), and the procedure that changes these bits (EEPROG).

- PROGRAM attempts to save programming time by checking which portions of the two-step programming procedure must occur. If the data already in the array is the same as the new data, then no programming is necessary. By omitting a write ones or a write zeros operation, 10 ms is removed from the total 20-ms programming time; every programming step that this routine omits saves 10 ms.

  The address and data to program are passed to this routine in the register pair ADDR1–1:ADDR1 and in register A, respectively.

- EEPROG is the routine that initiates, times, and then stops the actual EEPROM programming. During this section of code, disable the interrupts to prevent data corruption. Corruption can occur when an interrupt routine accesses any EEPROM location, interrupting the EEPROG routine between writing to the EEPROM location and setting the EXE bit (DEECTL.0). (Refer to Section 6.3, page 6-6.)

You can program unprotected data EEPROM using only the V_{CC} power supply. Enter the WPO mode by placing 12 V on the MC pin when programming protected data EEPROM.

The following program is used to write to any location in the data EEPROM.

Parameters used:

ADDR1–1:ADDR1 = EEPROM address to program;
A = data to write to EEPROM address

PROGRAM Routine

```
TEMP1    .EQU  R3              ;General-purpose temporary register
TEMP2    .EQU  R4              ;General-purpose temporary register
ADDR1    .EQU  R6              ;Contains address for program
                               ;operation
ECOM     .EQU  R7              ;Command for DEECTL
DEECTL   .EQU  P01A            ;Address for data EEPROM control reg
;
PROGRAM  MOV   A,TEMP2         ;Save data
         MOV   @ADDR1,A        ;Read current data
         XOR   TEMP2,A         ;Different bits = 1
         JZ    EXITW           ;If byte is already equal then exit
         INV   A               ;Different bits = 0
         OR    TEMP2,A         ;Bits that change from 1 to 0 = 0
         BTJZ  #0FFh,A,WRITE0  ;Program 0s if any 0s
         JMP   ONES            ;If all 1s then go to WRITE1 part
WRITE0   MOV   #1,ECOM         ;Program to write 0s (DEECTL = 1)
         MOV   TEMP2,A
         CALL  EEPROG          ;Programming EEPROM
ONES     MOV   @ADDR1,A        ;Get the current data
         XOR   TEMP2,A         ;Bits that change = 1
         AND   TEMP2,A         ;Bits that change from 0 to 1 = 1
         JZ    LASTCHK         ;Are there any 1s to program?
WRITE1   MOV   #3,ECOM         ;DEECTL value=3 (program 1s)
         MOV   TEMP2,A
         CALL  EEPROG          ;Program 0s
                               ;Verify the programming operation
LASTCHK  MOV   @ADDR1,A        ;Check new memory against wanted
                               ;memory
         CMP   TEMP2,A         ;If equal then exit
         JEQ   EXITW
;
;        Error-handling routine here
;
EXITW    RTS
```

EEPROG Routine

```
;
EEPROG   DINT                  ;Disable interrupts
         MOV   A,@ADDR1        ;Move data to address
         MOV   ECOM,DEECTL     ;Load DEECTL register
         EINT                  ;Enable interrupts
         MOVW  #2778,TEMP1     ;Wait 10 ms for EEPROM write
                               ;(at 20MHz)
WAIT10   INCW  #-1,TEMP1
         JC    WAIT10
         MOV   #0,DEECTL       ;Clear EXE bit
         RTS                   ;Exit from internal RAM program
```

The following portion of code is the same as the PROGRAM routine above but provides actual values for each step. The values shown are the low nibble of a byte expressed in binary; these values are shown because they provide all possible bit combinations.

In this example, the memory address contains x1100, and x1010 is programmed to that address. Before calling the EEPROG routine, the program writes new data to the EEPROM address located in register ADDR1–1:ADDR1 and then passes data to register A that specifies either a write ones or a write zeros operation.

```
                                        A          @(ADDR1-1:ADDR1)

                                ;       x1010      x1100
PROGRAM   MOV   A,TEMP2         ;                  Save data
          MOV   @ADDR1,A        ;       x1100      Read current data
          XOR   TEMP2,A         ;       x0110      Different bits = 1
          JZ    EXITW           ;                  If byte is already equal then exit
          INV   A               ;       x1001      Different bits = 0
          OR    TEMP2,A         ;       x1011      Bits that change from 1 to 0 = 0
          BTJZ  #0FFH,A,WRITE0  ;                  Program 0s if any 0s
          JMP   ONES            ;                  If all 1s then go to WRITE1 part
WRITE0    MOV   #1,ECOM         ;                  Program to write 0s (DEECTL = 1)
          MOV   TEMP2,A         ;       x1010
          CALL  EEPROG          ;                  x1000      Programming EEPROM
ONES      MOV   @ADDR1,A        ;       x1000      Get the current data
          XOR   TEMP2,A         ;       x0010      Bits that change = 1
          AND   TEMP2,A         ;       x0010      Bits that change from 0
                                ;                  to 1 = 1
          JZ    LASTCHK         ;                  Are there any 1s to
                                ;                  program?
WRITE1    MOV   #3,ECOM         ;                  DEECTL value=3 (program 1s)
          MOV   TEMP2,A         ;       x1010
          CALL  EEPROG          ;                  x1010      Program 0s
                                ;                  Verify the programming
                                ;                  operation
LASTCHK   MOV   @ADDR1,A        ;       x1010      Check new memory against
                                ;                  wanted memory
          CMP   TEMP2,A         ;                  If equal then exit
          JEQ   EXITW           ;
;
;         Error-handling routine here
;
EXITW     RTS
```

14.3 Serial Communications

All devices in the TMS370 family provide serial communication capability with the peripheral devices.

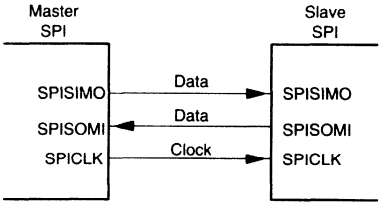
- ❑ The TMS370Cx1x series provides one serial port (SPI).
- ❑ The TMS370Cx2x series provides two serial ports (SPI and SCI).
- ❑ The TMS370Cx3x series provides one serial port that is part of the PACT module (PACT mini-SCI, see subsection 14.5.3).
- ❑ The TMS370Cx4x series provides one serial port (SCI).
- ❑ The TMS370Cx5x series provides two serial ports (SPI and SCI).

14.3.1 SPI Port Interfacing

The SPI port provides synchronous communication with peripherals such as shift registers, display drivers, A/D converters, and other microcomputers. Synchronous transmission is supported by programmable parameters such as the character length (one to eight bits) and the bit transfer rate (eight options).

In Figure 14–8, the SPI port is configured as a master/slave dual microcomputer interface. This full-duplex set-up has the master microcomputer initiating data transfer by sending the SPICLK signal to the slave. Data is then transmitted between the microcomputers simultaneously until the clock signal stops. Either or both of the data lines can send valid or dummy data, depending on the software.

Figure 14–8. Master/Slave SPI Interface Example



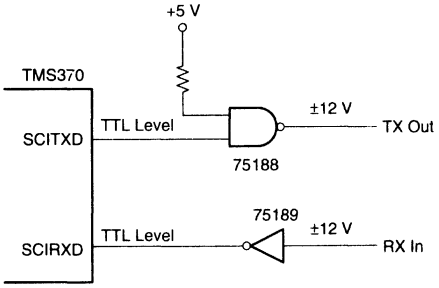
14.3.2 SCI Port Interfacing

The SCI port provides communication with a variety of peripheral devices in either asynchronous or isosynchronous mode. The format parameters of the SCI are programmable:

Parameter	Options
Mode	Asynchronous, Isosynchronous
Bit rate (baud)	64K possible bit rates
Character length	1 to 8 bits
Parity	Even, Odd, Off
No. of stop bits	1 or 2
Interrupt priorities	Receiver/transmitter

The SCI port is configured for an RS-232-C type interface in Figure 14–9. Since the TMS370 family uses TTL-level I/O, the transmit and receive data signals must be converted to RS-232 levels; the 75188 and 75189 devices provide this function. In the asynchronous mode, the clock signal does not need to be transmitted but is generated locally at both ends.

Figure 14–9. SCI/RS–232 Interface Example



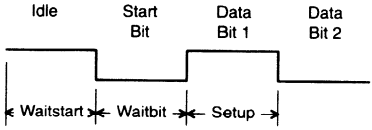
The routine in Example 14–1 automatically calculates the baud for the SCI port by timing the length of the start bit. This eliminates the need for external select switches, which can cause confusion.

The routine converts the SCIRXD pin to a general-purpose input pin and then samples this pin until it finds the start bit. Sampling is controlled by the baud counter, which takes 32 cycles for one complete count. At each count or every 32 cycles, the input pin is sampled. When the start bit is received, its low state is sampled until the high state of the first data bit (of an odd ASCII value) is detected. The baud register figures the bit rate according to the number of times the start bit is sampled. Refer to Figure 14–10 as you examine the routine.

Example 14–1. Calculating the SCI Baud

```
0050                        SCICCR   .EQU P050      ;SCI communication control register
0051                        SCICTL   .EQU P051      ;SCI control register
0052                        BAUDMSB  .EQU P052      ;Baud counter MSbyte
0053                        BAUDLSB  .EQU P053      ;Baud counter LSbyte
0054                        TXCTL    .EQU P054      ;Transmitter control
0055                        RXCTL    .EQU P055      ;Receiver control
0057                        RXBUF    .EQU P057      ;Receiver buffer
0057                        TXBUF    .EQU P059      ;Transmitter buffer
005D                        SCIPC1   .EQU P05D      ;Port control 1 (SCLK)
005E                        SCIPC2   .EQU P05E      ;Port control 2 (TXD,RXD)
005E                        SCIPRI   .EQU P05F      ;Priority register
005E                        COUNT    .EQU R04       ;Temporary counting register
0000
0000                        .TEXT    07000h         ;Initialize SCI port with a
0000                                                ;<CR> (return)
7000  AUTOBAUD                                       ;Baud automatically set on odd
7000                                                ;ASCII character
7000  D504                  CLR  COUNT              ;Clear count register
7002  D503                  CLR  COUNT-1            ;COUNT-1
7004  F7005E                MOV  #0,SCIPC2          ;Set RxD to general-purpose input pin
7007  A6085EFC  WAITSTRT    BTJO #8,SCIPC2,WAITSTRT ;Wait for a start bit to go low
700B
700B  B3        WAITBIT     INCA                    ;Dummy, gives 32 clock states
700C                                                ;(1 min baud)
700C  700104                INCW #1, COUNT          ;Increment counter
700F  A7085EF8              BTJZ #8,SCIPC2,WAITBIT  ;Wait until start bit ends
7013                                                ;(ASCII char=odd)
7013  70FF04    SETUP       INCW #-1,COUNT          ;One less than count into baud reg
7016  715304                MOV COUNT,BAUDLSB       ;since the SCI starts from count 0
7019  715203                MOV COUNT-1,BAUDMSB     ;Initialize baud registers
701C  F7225E                MOV #22h,SCIPC2         ;Enable Rx and Tx pins
701F  F7025D                MOV #2,SCIPC1           ;Enable SCLK pin (if needed)
7022  F77750                MOV #01110111b,SCICCR   ;8-bits length, even parity, 1 stop bit
7025                                                ;only even, odd, or none parity
7025                                                ;determined by SCICCR value
7025  F73351                MOV #00110011b,SCICTL   ;Enable Tx, Rx, SCLK = internal
702C                                                ;program after input character finishes
7028  F70154                MOV #1,TXCTL            ;Enable TX interrupts
702B  F70155                MOV #1,RXCTL            ;Enable RX interrupts
702E  8057                  MOV RXBUF,A             ;Clear out garbage from SCI (Place in
7032                                                ;program after input character finishes)
7032  F00C                  EINT
```

Figure 14–10. Autobaud Waveform



To increase flexibility and accuracy, you can improve the routine in Example 14–1 by using some of the following suggestions:

- For greater accuracy, time more than one bit and then divide by the number of bits. To do this, you must choose carefully the character to start the autobaud routine. The current routine can use 50% of the ASCII values (all odd ASCII values).
- Add a routine to check the parity of the incoming character and set the parity of the SCI port accordingly. Again, this means a limited number of characters will correctly autobaud the routine.
- As an accuracy check, add routines to compare the count of another bit in the character to the start bit count. Again, you must choose the correct character to start the autobaud routine.

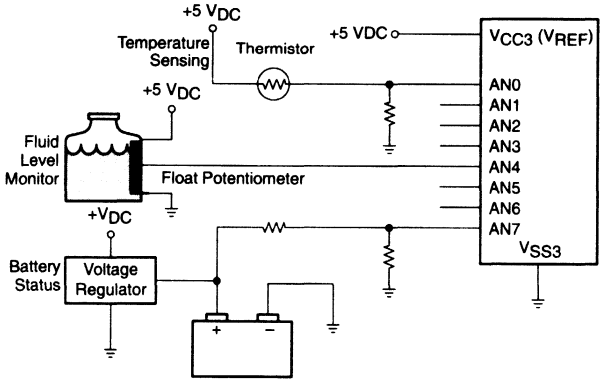
For a more in-depth discussion of the uses of the TMS370 SCI or SPI, refer to the *Using the TMS370 SPI and SCI Modules Application Report.*

14.4 Analog-to-Digital Converter

The A/D converter provides built-in data acquisition capability with 8-bit resolution. Any or all of the A/D pins can be used as a single-ended input with reference to the analog ground (V_{SS3}). Pins not used for A/D conversion can be software configured as standard digital input pins. The high-reference voltage (V_{REF}) can be either V_{CC3} or supplied by one of the inputs. If the sampled input is higher than V_{REF}, the conversion value placed in the A/D data register is FFh, indicating full scale. If the sampled input is lower than V_{SS3}, the value 00h is placed in the A/D data register.

The A/D converter enables the CPU to perform a variety of functions. Industrial applications include temperature sensing, fluid-level monitoring, and recharging circuit status as indicated in Figure 14–11. If the voltage of the sending units is greater than V_{REF}, a resistance network may be needed to keep the A/D input voltage within the meaningful range of V_{REF} to ground. This is especially true in the case of a fluid-level sensor, where the full linear range may be required.

Figure 14–11. A/D Converter Sample Applications



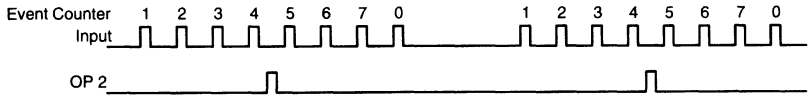
14.5 PACT Module

This section contains sample routines that show some of the capabilities of the PACT module. Section 12.6 discusses how to use the PACT module to produce a PWM signal. All of these routines use a macro header file (PACT.H) to facilitate the set-up of the PACT command/definition area. This header file is discussed in Appendix I.

14.5.1 Time After Event Example

The offset timer definition and the Conditional Compare command create an output signal that is delayed in time from an input signal. For example, in Figure 14–12, the input signal consists of a series of eight short pulses that are 100 ms apart. These pulses repeat after an unspecified time period. This example creates an output pulse of 10 ms duration centered between the fourth and fifth pulse of the series.

Figure 14–12. Time After Event—Example Waveforms



The input signal is connected to CP6 of the PACT module so that it will increment the event counter. Any of the output pins OP1–7 can be used for the output signal; OP2 has been chosen in this example.

The routine for this task consists of four parts:

- ❑ Defining the PACT command/definition area
- ❑ Disabling the PACT module and any currently enabled capture pins
- ❑ Transferring the new command/definition area into dual-port RAM
- ❑ Initializing the PACT peripheral frame

The following subsections describe these parts. Refer to Example 14–2 on page 14-25.

14.5.1.1 Defining the PACT Command/Definition Area

To establish a time base synchronized to the event counter, this routine uses the offset timer definition. Since the PACT command/definition area cannot start with a definition, a dummy standard command is used as the first entry.

- ❑ Line 14 of the example routine is the offset timer definition. The maximum event count is set to 7, allowing the event counter to increment from 0 through 7 (eight counts).

- Line 15 creates the Conditional Compare command that causes the output pin to go high. This command waits for the fourth rising edge on the input pin and then delays another 50 ms before setting the pin high. The macro file automatically subtracts 2 from the supplied number to create the value encoded in the command.

- Line 16 creates the Conditional Compare command that causes the output pin to go low. This happens 60 ms after the fourth rising edge on the input pin. A problem can occur if the fifth input pulse occurs less than 60 ms after the fourth pulse. Since the Conditional Compare command would not see the event count set to four and the offset time value equal to 60, the pin would never be cleared. By enabling the evt_plus1 action of this command, you avoid this problem; the output pin is cleared 60 ms after the fourth event or immediately after the fifth event, whichever comes first.

The actual time that it takes for the output pin to toggle is determined by three different delays: the specified delay, a synchronization delay, and a propagation delay.

- The specified delay is used in defining the Conditional Compare command. It is specified in PACT prescaled clock time periods and can be no less than two of these periods.

- The synchronization delay is one prescaled clock time period. Since the output must occur on a prescaled clock boundary, but the input edge can occur at any time, one prescaled clock period of jitter results.

- The propagation delay is three system clock periods (SYSCLK) plus an asynchronous delay of approximately 80 ns.

For a device operating with a 20-MHz crystal using a PACT prescale value of 5, with a Conditional Command delay of two, the delay from the input edge on pin CP6 to the output pin edge is:

$$\text{DELAY} = 2.5 \times \text{PS} + 3 \times \text{SYSCLK} + 80\ \text{ns} \pm 1/2\ \text{PS}$$

$$= 2.5 + 0.6 + 0.08 \pm 0.5 = 3.18 \pm 0.5\ \mu\text{s}$$

PS = PACT prescaled clock period

SYSCLK = System clock period

14.5.1.2 Disabling the PACT Module and Currently Enabled Capture Pins

When you load new information into the command/definition area, it is very important that you disable the PACT module. A half-modified command or definition could cause the PACT module to corrupt other parts of the dual-port RAM. The command/definition area is disabled by line 21 of the example.

To ensure that captures into the old circular buffer do not corrupt your new commands, disable all captures as well. Lines 22–24 disable all captures from the six input pins. This part of the code is not necessary if the PACT module is being set up immediately after reset.

14.5.1.3 Transferring the New Commands and Definitions Into Dual-Port RAM

Lines 27 through 32 of this routine copy the new commands and definitions into the dual-port memory. These instructions flip the table created by the macros end for end, making the table much easier to read because the first command listed will be the first one executed by the PACT module.

14.5.1.4 Initializing the PACT Peripheral Frame

Lines 35 through 40 initialize the peripheral frame. Mode A (the default mode) is used. Initializing the command start address to location 0EBh allows for two 32-bit entries in the circular buffer.

Lines 37 and 38 clear the event counter and set the event prescaler to no prescale. The rising edge of CP6 is enabled to perform captures to the circular buffer and increment the event counter. In a real application, you must ensure that the event counter is not initialized in the middle of a series of eight input pulses.

In line 40, the PACT prescale value is set to five, and the command/definition area is enabled.

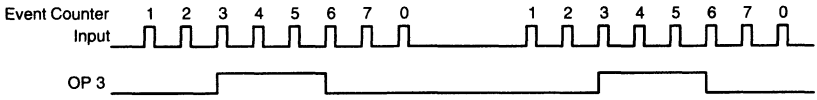
Example 14–2. Time After Event Routine

```
0001   ;This is an example program to do a time after event function
0002   ;
0003   ;MACRO DEFINITIONS
0004   ; STDCMP <compare value>,<pin>,<actions>
0005   ; OFSTMR <max event count>,<actions>,<initial value>
0006   ; CONCMP <evt cmp value>,<time cmp value>,<output pin>,<actions>
0007   ;
0008           .include "pact.h"
0009   cmd_st  .equ       0ebh
0010
0011   ;PACT commands
0012           .data 7800h
0013   table   stdcmp     00,00,nxt_def            ;Next line is definition
0001   #       .byte      0,0,1,0
0014           ofstmr     7,enable
0001   #       .byte      1,0,4,7
0015           concmp     4,50,op2,set_pin
0001   #       .byte      50,0,164,4
0016           concmp     4,60,op2,clr_pin|evt_plus1
0001   #       .byte      60,0,196,4
0017   len     .equ       $-table
0018           .text 6000h
0019   start   mov        #3,p04f                  ;Disable the watchdog
0020   ;Make sure that all captures are disabled
0021           mov        #0,p040                  ;Disable cmd/def area
0022           mov        #0,p04a                  ;Disable all capture pins
0023           mov        #0,p04b
0024           mov        #0,p04c
0025
0026   ;Copy PACT cmds/defs into RAM
0027           mov        #len,b
0028           movw       #(cmd_st-len+1),r3
0029   loop    mov        table-1(b),a
0030           mov        a,@r3
0031           inc        r3
0032           djnz       b,loop
0033
0034   ;Set up the peripheral file
0035           mov        #cmd_st,p041             ;Cmd/def start at 0ebh
0036           mov        #(cmd_st-len+1),p042     ;Cmd/def end at 0e0h
0037           mov        #2,p04d                  ;Clear the event counter
0038           mov        #0,p04d                  ;Set event prescale to no prescale
0039           mov        #20h,p04c                ;Select rising edge on CP6
0040           mov        #034h,p040               ;Set prescale to 5, 1 usec resolution
0041
0042   ;Signal running without processor intervention
0043           idle
0044           .end
```

14.5.2 Double Event Compare Command Example

If an output pin is to be set or cleared after an input event without an additional time delay, the Double Event Compare command should be used; this command can both set and clear an output pin with respect to two different values of the event counter. Figure 14–13 shows an input signal that consists of repeated series of eight short pulses. This signal is attached to the event counter input pin CP6. The desired output signal, shown as coming from output pin OP3, goes high on the leading edge of the third pulse and low on the leading edge of the sixth pulse of each series.

Figure 14–13. Double Event Compare—Example Waveforms



The routine for this task consists of four parts:

- ❑ Defining the PACT command/definition area
- ❑ Disabling the PACT module and any currently enabled capture pins
- ❑ Transferring the new command/definition area into dual-port RAM
- ❑ Initializing the PACT peripheral frame

The last three parts of this routine are exactly the same as in the previous example (subsections 14.5.1.2 through 14.5.1.4), so they will not be discussed again here.

Refer to Example 14–3 on the following page.

The PACT command/definition area in this example consists of one command and one definition. Line 10 is the Double Event Compare command. Since it does not refer to any timer, it can be the first entry in the command/definition area. Two event compare values are 3 and 6. OP3 is the specified output pin. This command sets the output pin when the event counter reaches 3 (the first event compare action) and does the opposite—that is, clears the output pin—when the event counter reaches six. In addition, this command specifies that the next block in the command/definition area is a definition.

Line 11 of this example routine is an offset timer definition. Its sole purpose is to set the maximum event count value to 7. The event counter will increment from one to 7 and then reset to 0 on the last edge of the input series.

The delay from the edge that increments the event counter until the output pin changes state is expressed by:

$$DELAY = 1.5 \times PS + 3 \times SYSCLK + 80\ ns \pm 1/2\ PS$$

PS = PACT prescaled clock period

SYSCLK = System clock period

This example uses a 20-MHz crystal and a PACT prescale value of 5; the delay is calculated as:

$$DELAY = 1.5 + 0.6 + 0.08 \pm 0.5 = 2.18 \pm 0.5\ \mu s$$

Example 14–3. Double Event Compare Command Routine

```
0001    ;This is an example program to do a Double Event Compare function
0002              .include "pact.h"
0003
0004    ;MACRO FORMATS
0005    ; DEVCMP <event value 1>,<event value 2>,<output pin>,<actions>
0006    ; OFSTMR <max event count>,<actions>,<initial value>
0007
0008    ;PACT commands and definitions
0009              .sect     "pact",7800h
0010    table     devcmp    3,6,op3,set_evt1|opp_act|enable|nxt_def
0001    #         .byte     3,6,37,11
0011              ofstmr    8,enable
0001    #         .byte     1,0,4,8
0012    len       .equ      $-table
0013              .text 6000h
0014    start     mov       #3,p04f           ;Disable the watchdog
0015    cmd_st    .equ      0ebh
0016              mov       #0,p040           ;Disable cmd/def area
0017              mov       #0,p04a           ;Disable all input captures
0018              mov       #0,p04b
0019              mov       #0,p04c
0020
0021    ;Copy PACT cmds/def into RAM
0022              mov       #len,b
0023              movw      #(cmd_st-len+1),r3
0024    loop      mov       table-1(b),a
0025              mov       a,@r3
0026              inc       r3
0027              djnz      b,loop
0028
0029    ;Set up the peripheral file
0030              mov       #cmd_st,p041      ;Cmd/def start at 0ebh
0031              mov       #(cmd_st-len+1),p042
0032              mov       #04,p04d          ;Event only & evt prescale to divide by 1
0033              mov       #034h,p040        ;Timer prescale to 5, 1 usec resolution
0034              mov       #2,p04d           ;Clear the event counter
0035              mov       #0,p04d           ;Set event prescale to no prescale
0036              mov       #20h,p04c         ;Select rising edge on CP6
0037
0038    ;PACT running without processor intervention
0039              idle
0040              .end
```

14.5.3 PACT SCI Example

This example routine is a simple test that shows how to use the PACT mini-SCI. The example consists of five parts:

- ❑ Defining the PACT command/definition area
- ❑ Disabling the PACT module and any currently enabled capture pins
- ❑ Transferring the new command/definition area into dual-port RAM
- ❑ Initializing the PACT peripheral frame
- ❑ Running the SCI test routine

The second, third, and fourth parts are the same as those used in the other PACT example routines and are explained fully in subsections 14.5.1.2 through 14.5.1.4; they will not be discussed here.

Refer to Example 14–4 on the following page.

The PACT command/definition area establishes the SCI transmit and receive bit rates. The baud timer definition defines these bit rates. The first entry in the command definition area, line 13, is a dummy Standard Compare command because this area cannot start with a definition. Line 14 of this routine is the actual baud timer definition. In this example, the transmit and receive rates are initialized to the same value. The maximum timer value is set by the equation found in subsection 12.9:

$$\text{Max Virtual Timer Value} = \frac{1}{(\text{Baud})\ (4)\ (\text{PACT Resolution})} - 2$$

For our example of 9600 baud with a PACT resolution of 1 ms, the required value is:

$$\text{Max Virtual Timer Value} = \frac{1}{(9600)\ (4)\ (10^{-6})} - 2$$

$$= 24$$

The actual SCI test is implemented in lines 39 through 48 of this routine. The 8-bit values are sent out on the transmit pin and received on the receive pin. In this simple example, software polling is used to determine when the transmit or receive buffers are ready. More sophisticated routines would use the interrupts for these two buffers.

The two idle instructions in lines 47 and 48 set breakpoints in a PACT XDS/22 emulator. If line 47 is reached, the routine completed without an error. If line 48 is reached, an error was detected. The value transmitted will be in register 2 and the value received will be in register 3.

Example 14–4. PACT Mini-SCI Routine

```
0001   ;This is an example program to verify the serial communications interface
0002   ;on the PACT module. The transmit and receive pins are shorted externally
0003   ;for this test. The timer is set up for 9600 baud.
0004
0005             .include "pact.h"
0006   cmd_st    .equ      0ebh
0007   rxreg     .equ      p046
0008   txreg     .equ      p047
0009   rxrdy     .dbit     7,p045
0010   txrdy     .dbit     6,p045
0011
0012             .sect     "pact",7800h
0013   table     stdcmp    00,00,nxt_def     ;Next line is definition
0001   #         .byte     0,0,1,0
0014             brtmr     24,rx!tx
0001   #         .byte     0,0,199,0
0015   len       .equ      $-table
0016             .text     6000h
0017   start     mov       #3,p04f           ;Disable the watchdog
0018             mov       #0,p040           ;Disable cmd/def area
0019             mov       #0,p04a           ;Disable all input captures
0020             mov       #0,p04b
0021             mov       #0,p04c
0022
0023   ;Copy PACT commands and definitions into RAM
0024             mov       #len,b
0025             movw      #(cmd_st-len+1),r3
0026   loop      mov       table-1(b),a
0027             mov       a,@r3
0028             inc       r3
0029             djnz      b,loop
0030
0031   ;Set up the peripheral file
0032             mov       #07,p04f          ;Set to mode B
0033             mov       #cmd_st,p041      ;Cmd/def start at 0ebh
0034             mov       #(cmd_st-len+1),p042
0035             mov       #04,p04d          ;Event only & no event prescale
0036             mov       #034h,p040        ;Set prescale = 5, 1 usec resolution
0037
0038   ;Send the numbers 0-255 and receive them on the loopback
0039             clr       r2
0040   loop1     jbit0     txrdy,loop1       ;Wait until transmit buffer is empty
0041             mov       r2,txreg
0042   rxwait    jbit0     rxrdy,rxwait      ;Wait until character is received
0043             mov       rxreg,r3
0044             cmp       r2,r3             ;Compare characters
0045             jne       error
0046             djnz      r2,loop1          ;Loop for all 256 characters
0047             idle                        ;Stop here if no errors
0048   error     idle                        ;Error condition occurred
0049             .end
```

14.6 Sample Routines

This section contains sample routines that show the various ways the TMS370 handles common software tasks.

14.6.1 T1PWM Pin Set-Up

- This example starts and stops the PWM function with a certain value on the PWM pin. Starting the T1PWM pin with a specific value can be done with one instruction as shown below. The value of the data out bit will become the initial value of the PWM pin.

```
MOV   #60h,P04E        ;Start with PWM pin high

MOV   #20h,P04E        ;Start with PWM pin low
```

- This example shows the two instructions needed to change the T1PWM pin from a PWM pin to a general-purpose output pin with a specific value. The first instruction changes the pin to a general-purpose output pin with the same value as the current PWM pin. The second instruction changes the pin to a particular value.

```
MOV   #50h,P04E        ;Stop with PWM pin high.
MOV   #50h,P04E        ;

MOV   #10h,P04E        ;Stop with PWM pin low.
MOV   #10h,P04E        ;
```

- The following example starts and stops the PWM function with the current value on the pin. Starting the function requires four instructions, while stopping the function takes only one.

```
      MOV   #20h,A           ;Start with PWM pin same as
      BTJZ  #80h,P04E,SKIP   ;current state.
      MOV   #60h,A           ;
SKIP  MOV   A,P04E           ;

      MOV   #10h,P04E        ;Stop with PWM pin same as
                             ;current state.
```

14.6.2 Clear RAM

This routine clears all of the internal RAM registers. It can be used at the beginning of a program to initialize the first 256 bytes of RAM to a known value. Registers A and B have the following functions in this routine:

- ❑ Register A holds the initialization value
- ❑ Register B serves as the index into the RAM

```
0000           0001          .TEXT 7000h   ;Absolute start address
7000  52FE      0002 CLEAR    MOV #254,B    ;Number of register to clear
               0003                        ;-2
7002  B5        0004          CLR A         ;Load the initialization
               0005                        ;value of zero
7003  AB0001    0006 LOOP     MOV A,1(B)    ;Clear the location indexed
               0007                        ;by B+1
7006  CAFB      0008          DJNZ B,LOOP   ;Loop until all RAM is
               0009                        ;cleared
7008           0010                        ;A and B end up as zeros.
```

14.6.3 RAM Self Test

This routine performs a simple alternating 0/1 test on the first 256 bytes of RAM by writing a AA,55 pattern to the entire RAM space and then checking the RAM for this pattern. The inverted pattern is then written to RAM and rechecked. Finally, the entire RAM is cleared. If an error is found, a bit is set in the flag register. The error flag bit should be cleared before the routine is started.

Register	Before	After No Error	After Error
A	XX	0	?
B	XX	0	?
Rn	XX	0	?
FLAG	XX	0	Bit 0=1

- ❑ Passing data: none
- ❑ Registers affected: all
- ❑ Ending data: all registers = 0; bit 0 in FLAG = 1 if error was found

```
0000           0001          .TEXT 7000H     ;Absolute start address
000A           0002 FLAG     .EQU  R10       ;Error register
7000 2255      0003          MOV   #55h,A    ;Start RAM fill with 55h
7002 52FD      0004 FILLR    MOV   #0FDh,B   ;Set RAM start address - 2
7004           0005                          ;(don't change register A or B)
7004 AB0002    0006 FILL1    MOV   A,2(B)    ;Fill RAM with aa to 55 pattern
7007 BC        0007          RR    A         ;Change to beginning number
7008 CAFA      0008          DJNZ  B,FILL1   ;Fill entire RAM with pattern
700A BC        0009          RR    A         ;Change to beginning number
700B 52FD      0010          MOV   #0FDh,B   ;Refresh index
700D AD0002    0011 COMPAR   CMP  2(B),A     ;Check for errors
7010 06**      0012          JNE   ERROR     ;Exit if values don't match
7012 BC        0013          RR    A         ;Change from 55 to AA to 55
7013 CAF8      0014          DJNZ  B,COMPAR  ;Check the entire RAM
7015 B0        0015          CLRC            ;Is reg A now 55, AA or 00?
7016 01EA      0016          JN    FILLR     ;=AA, change to opposite pattern
7018 02**      0017          JZ    EXIT      ;=00,
701A B5        0018 FILL0    CLR   A         ;=55,clear the ram now
701B 00E5      0019          JMP   FILLR     ;Repeat the fill and check routine
701D 74010A    0020 ERROR    OR    #1,FLAG   ;Set bit zero in the flag
7020           0021                          ;register
7020           0022 EXIT     .EQU $          ;Continue program here
```

14.6.4 ROM Checksum

This routine checks the integrity of a 4K-byte ROM by performing a checksum on the entire ROM. All ROM bytes from 7002h to 7FDFh are added together in a 16-bit word. The sum is checked against the value at the beginning of the ROM (7000h, 7001h). If the values don't match, then an error has occurred, and a bit is set in a register. The error flag bit should be cleared before the start of the routine. This routine can easily be modified for other ROM sizes.

Note:

Addresses 7FE0h through 7FEBh are reserved for TI use only and should not be used in a checksum calculation.

Register	Before	No Error	Error
A	XX	X	X
B	XX	X	X
R2	XX	CHKSUM MSbyte	CHKSUM MSbyte
R3	XX	CHKSUM LSbyte	CHKSUM LSbyte
R4	XX	70h	70h
R5	XX	01h	01h
R6	XX	FFh	FFh
R7	XX	FFh	FFh
FLAG	XX	Bit 1 = 0	Bit 1 = 1

```
0000            0001            .TEXT   7000h       ;Absolute start address
000F            0002    FLAG    .EQU    R15         ;Error status
3039            0003    CHECKSUM .EQU   12345       ;Value to be checked against
7000  3039      0004            .WORD   CHECKSUM    ;Put correct checksum into
7002            0005                                ;ROM
7002            0006                                ;Other initialization
7002            0007                                ;program here
7002  887FDF05 0008    ROMCHK  MOVW    #7FDFh,R5   ;Starting address (end of
7006            0009                                ;memory)
7006  880FDD07 0010            MOVW    #0FDDh,R7   ;Number of bytes to add + 1
700A  8000003  0011            MOVW    #0,R3       ;Reset summing register
700E            0012                                ;
700E  9A05      0013    ADDLOP  MOV     @R5,A       ;Get ROM byte
7010  480003    0014            ADD     A,R3        ;Add to 16-bit sum
7013  790002    0015            ADC     #0,R2       ;Add any carry
7016  70FF05    0016            INCW    #-1,R5      ;Decrement address
7019  70FF07    0017            INCW    #-1,R7      ;Decrement byte counter
701C  03F0      0018            JC      ADDLOP      ;Continue until byte count
701E            0019                                ;goes past 0
701E            0021                                ;
701E  8A7000    0022            MOV     7000h,A     ;Compare MSbyte stored to
7021            0023                                ;MSbyte sum
7021  4D0002    0024            CMP     A,R2        ;
7024  06**      0025            JNE     ERROR       ;Set error bit if different
```

```
7026   8A7001   0026            MOV     7001h,A    ;Compare LSbyte stored to
7029            0027                               ;LSbyte sum
7029   4D0003   0028            CMP     A,R3       ;
702C   02**     0029            JEQ     EXIT       ;Set error bit if different
702E   74020F   0030   ERROR    OR      #2,FLAG    ;Set bit 1 in the flag
7031            0031                               ;register
7031            0032   EXIT     .EQU               ;Continue program here
```

14.6.5 Binary-to-BCD Conversion

This program converts a 16-bit binary word to a packed six-nibble value.

Register	Before	After
A	XX	BCD MSbyte
B	XX	BCD
R2	XX	BCD LSbyte
R3	BINARY MSbyte	ZERO
R4	BINARY LSbyte	ZERO
R5	XX	ZERO

```
0000              0001            .TEXT 7000H    ;Absolute start address
7000 B5           0002 BN2BCD     CLR   A        ;Prepare answer registers
7001 C5           0003            CLR   B        ;
7002 D502         0004            CLR   R2       ;
7004 721005       0005            MOV   #16,R5   ;Move loop count to register
7007 DF04         0006 LOOP       RLC   R4       ;Shift higher binary bit out
7009 DF03         0007            RLC   R3       ;Carry contains higher bit
700B 4E0202       0008            DAC   R2,R2    ;Double the number then add
700E              0009                           ;the binary bit
700E 3E01         0010            DAC   R1,B     ;Binary bit (a 1 in carry on
7010              0011                           ;the 1st time is
7010 1E00         0012            DAC   R0,A     ;doubled 16 times).
7012 DA05F2       0013            DJNZ  R5,LOOP  ;Do this 16 times, once for
7015              0014                           ;each bit
7015 F9           0015            RTS            ;Back to calling routine
```

14.6.6 BCD-to-Binary Conversion

This routine converts a four-digit number to binary. The maximum BCD number is 9999 decimal. Operands originate and are stored in general-purpose RAM. The BCD number is composed of the four digits (D3, D2, D1, and D0) contained in the bytes DH and DL. The binary number is calculated by dividing the number into powers of ten (Binary = D3*1000 + D2*100 + D1*10 + D0*1). Multiplying by 10 is easier if the number is further broken up into other numbers so that D2*10 = D2*(8+2) = D2*8+D2*2. Likewise, multiplying by 1000 can be calculated by D3*(1000) = D3*(1024–24) = D3* (1024– (8+16)) = D3*1024– (D3*8 + D3*16). This may seem complex, but it works quickly and uses few bytes.

```
0000              0010       .TEXT 7000h
0002              0011 BH    .EQU  R2          ;Binary number MSbyte
0003              0012 BL    .EQU  R3          ;Binary number LSbyte
0004              0013 DH    .EQU  R4          ;Decimal number MSbyte
0005              0014 DL    .EQU  R5          ;Decimal number LSbyte
7000              0017                         ;D0=ones, D1=tens,
700C              0018                         ;D2=hundreds, D3=thousands
700C D502         0023 TOP   CLR   BH          ;Clear out binary MSbyte
700E 420503       0024       MOV   DL,BL       ;D0 to B0
7011 730F03       0025       AND   #0Fh, BL    ;Convert D0
7014              0026                         ;
7014 1205         0027       MOV   DL,A        ;D1*10=D1*8+D1*2
7016 23F0         0028       AND   #0F0h,A     ;Isolate D1
7018 C0           0029       MOV   A,B         ;B=D1*16
7019 D701         0030       SWAP  R1          ;B=D1
701B BC           0031       RR    A           ;A=D1*16/2=D1*8
701C CE           0032       RL    B           ;B=D1*2
701D 68           0033       ADD   B,A         ;A=D1*10   (D1*8+D1*2)
701E 480003       0034       ADD   R0,BL       ;D1:D0 converted
7021              0035                         ;
7021 3204         0036       MOV   DH,B        ;Get upper two digits
7023 530F         0037       AND   #0Fh,B      ;Isolate D2
7025 5C64         0038       MPY   #100,B      ;R0:R1=D2*100
7027 480103       0039       ADD   R1,BL       ;Add to current total
702A 490002       0040       ADC   R0,BH       ;D2:D1:D0 converted
702D              0041                         ;
702D 1204         0042       MOV   DH,A        ;Isolate D3
702F 23F0         0043       AND   #0F0h,A     ;A=D3 * 16
7031 C0           0044       MOV   A,B         ;B=D3 * 16
7032 CD           0045       RRC   B           ;B=D3 * 8
7033 68           0046       ADD   B,A         ;A=D3 * 24
7034 4A0003       0047       SUB   R0,BL       ;BH:BL=BH:BL-24*D3
7037 7B0002       0048       SBB   #0,BH       ;
703A B0           0049       CLRC              ;Setup for rotate
703B CD           0050       RRC   B           ;B=D3*4
703C 480102       0051       ADD   R1,BH       ;BH:BL=BH:BL+D3*4*256
703F              0052                         ;
```

14.6.7 BCD String Addition

The following routine uses the addition instruction to add two multidigit numbers together. Each number is a packed BCD string of less than 256 bytes (512 digits), stored at memory locations STR1 and STR2. This routine adds the two strings together and places the result in STR2. The strings must be stored with the most significant byte in the lowest numbered register. The TMS370 family instruction set encourages storing all numbers and addresses with the most significant byte in the lower numbered location.

Register	Before	After	Function
A	XX	??	Accumulator
B	XX	0	Length of string
R2	XX	??	Temporary save register
STR1	BINARY MSbyte	no change	BCD string
STR2	BINARY LSbyte	STR1 + STR2	Target string, 6 bytes max

```
0000 ;Decimal Addition Subroutine. Stack must have 3 available bytes.
0000 ;On output: STR2 = STR1 + STR2
0000          0001          .TEXT       7000h       ;Absolute start address
80E0          0002  STR1    .EQU        80E0h       ;Start of first string
80F0          0003  STR2    .EQU        80F0h       ;Start of second string
7000          0004                                  ;and result
7000 B0       0005  ADDBCD  CLRC                    ;Clear carry bit
7001 FB       0006          PUSH        ST          ;Save status to stack
7002 AA80DF   0007  LOOP    MOV         STR1-1(B),A ;Load current byte
7005 D002     0008          MOV         A,R2        ;Save it in R2
7007 AA80EF   0009          MOV         STR2-1(B),A ;Load next byte of STR2
700A FC       0010          POP         ST          ;Restore carry from last add
700B 1E02     0011          DAC         R2,A        ;Add decimal bytes
700D FB       0012          PUSH        ST          ;Save the carry from this add
700E AB80EF   0013          MOV         A,STR2-1(B) ;Store result
7011 CAEF     0014          DJNZ        B,LOOP      ;Loop until done
7013 FC       0015          POP         ST          ;Restore stack to starting
              0016                                  ;position
7014 F9       0017          RTS                     ;Back to calling routine
```

Notice the use of the indexed addressing mode to reference the bytes of the decimal strings. In addition, it is necessary to push the status register between decimal additions to save the decimal carry bit. Register B is used to keep count of the number of bytes that have been added.

14.6.8 Fast Parity

This routine presents a quick way to determine the parity of a byte. Exclusive ORing all the bits of the byte together derives a single bit that is the even parity of the word. With exclusive ORing, an even number of 1s combines to form a 0, leaving either an odd 1 or 0 bit. This routine keeps splitting the byte in half and exclusive ORing the two halves.

Register	Before	After	Function
A	TARGET	??	Passing byte from program
B	XX	??	
CARRY	XX	Parity	Status bit, result to calling routine

```
*************************************************************************
*       STEP 1                                          SUBROUTINE
*       Byte bits    7654 3210                          TO FIND
*                XOR      7654 [MSB above]             EVEN PARITY
*                  ===========
*                    xxxx ABCD
*       STEP 2                 ------>   AB CD
*                                  XOR     AB [MS bits above]
*                                    =======
*                                      xx ab
*       STEP 3                                ---->    a b
*                                                XOR    a [MS bit]
*                                                  =====
*                                                    x P    {answer}
*
*************************************************************************
0000          0001          .TEXT     7000h    ;Absolute start address
7000 C0       0002 PARITY   MOV       A,B      ;Duplicate the target byte
7001 B7       0003          SWAP      A        ;Line up the ms nibble with the ls
7002          0004                             ;nibble
7002 65       0005          XOR       B,A      ;Exclusive OR the nibbles to get a
7003          0006                             ;nibble answer
7003 C0       0007          MOV       A,B      ;Duplicate the nibble answer
7004 BC       0008          RR        A        ;Line up bits 0,1 of the answers to
7005          0009                             ;bits
7005 BC       0010          RR        A        ;2, 3 of the answer
7006 65       0011          XOR       B,A      ;XOR to get a new 2-bit answer
7007 C0       0012          MOV       A,B      ;Duplicate this 2 bit answer
7008 BC       0013          RR        A        ;Line up bit 0 with bit 1
7009 65       0014          XOR       B,A      ;XOR for final even parity answer
700A BC       0015          RR        A        ;Rotate answer into the carry bit
700B          0016                             ;and bit 7
700B F9       0017          RTS                ;Carry = 0 = even # of 1's
700C          0018                             ;Carry = 1 = odd # of 1's
700C          0019                             ;Use JC, JN, or JNC  in next
700C          0020                             ;executed instruction
```

14.6.9 Bubble Sort

This routine sorts up to 256 bytes using the bubble sort method. Longer tables can be sorted using the indirect addressing mode.

Register	Function
A	Temporary storage register
B	Index into the RAM
R2	Holds flag to indicate a byte swap has been made

```
0000             0001        .TEXT   7000h            ;Absolute start address
2000             0002 TABLE  .EQU    2000h            ;Start of data table in RAM
0002             0003 FLAG   .EQU    R2               ;'Swap has been made' flag
7000 D502        0004 SORT   CLR     FLAG             ;Reset swap flag
7002 52FF        0005        MOV     #0FFh,B          ;Load table offset value
7004 AA2000      0006 LOOP1  MOV     TABLE(B),A       ;Look at entry in table
7007 AD1FFF      0007        CMP     TABLE-1(B),A     ;Look at next lower byte
700A 0B**        0008        JHS     LOOP2            ;If higher or equal, skip to
700C             0009                                 ;next value
700C D302        0010        INC     FLAG             ;Entry is not lower, set swap
700E             0011                                 ;flag
700E B8          0012        PUSH    A                ;Store upper byte
700F AA1FFF      0013        MOV     TABLE-1(B),A     ;Take lower byte
7012 AB2000      0014        MOV     A,TABLE(B)       ;Put where upper was
7015 B9          0015        POP     A                ;Get the old upper byte
7016 AB1FFF      0016        MOV     A,TABLE-1(B)     ;Put where the lower byte was
7019 CAE9        0017 LOOP2  DJNZ    B,LOOP1          ;Loop until all the table
701B             0018                                 ;is looked at
701B 76FF02E1    0019        BTJO    #0FFh,FLAG,SORT  ;If swap was made, then
701F             0020                                 ;resweep table
701F F9          0021        RTS                      ;If no swap was made, then
                 0022                                 ;table is done
```

14.6.10 Table Search

The CMPA (Compare Register A Extended) instruction efficiently performs table searches. In the following example, a 150-byte table is searched for a match with a 6-byte string.

Register	Before	After	Function
A	XX	??	
B	XX	??	
R2	XX	??	Table length
TABLE	XX	no change	Long string in table
STRING	XX	no change	Target string, 6 bytes max

```
0000          0001          .TEXT  7000h          ;Absolute start address
2000          0002 TABLE    .EQU   2000           ;Start of data table in RAM
000A          0003 STRING   .EQU   R10            ;Start of target string,
7000          0004                                ;6 bytes max
7000 729602   0005 SEARCH   MOV    #150,R2        ;Table length = 150 bytes
7003 5206     0006 LOOP1    MOV    #6,B           ;String length = 6 bytes
4005 D602     0007 LOOP2    XCHB   R2             ;Swap pointers, long string in B
7007 C2       0008          DEC    B              ;Reduce index into table
7008 07**     0009          JNC    NOFIND         ;Table end? if so, no match found
700A AA2000   0010          MOV    TABLE(B),A     ;Load test character
700D D602     0011          XCHB   R2             ;Swap pointers, string pointer in
700F AD0009   0012          CMP    STRING-1(B),A;Match?
7012 06EF     0013          JNE    LOOP1          ;If not, reset string  pointer
7014                                              ;else test
7014 CAEF     0014          DJNZ   B,LOOP2        ;Next character
7016          0015 MATCH    .EQU   $              ;Match found
7016          0016 NOFIND   .EQU   $              ;No match found
7016          0017
```

The indexed addressing mode is used in this example and has the capability to search a 256-byte string, if needed. Register B alternates between a pointer into the 6-byte test string and a pointer into the longer table string.

14.6.11 16-by-16 (32-Bit) Multiplication

This example multiplies the 16-bit value in register pair R2, R3 by the value in register pair R4, R5. The results are stored in R6, R7, R8, R9; registers A and B are altered.

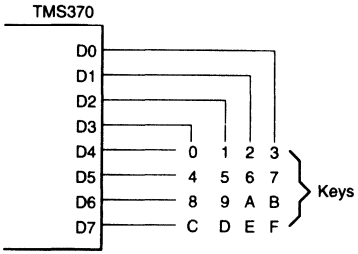
```
*******************************************************
*   16-BIT MPY:                  XH    XL       X VALUE
*                           X    YH    YL       Y VALUE
*                           ---------------
*                              XLYLm  XLYL1     1 = LSB
*                        XHYLm XHYL1           m = MSB
*                        XLYHm XHYH1
*           +  XHYHm     XHYH1
*           -------------------------------
*              RSLT3  RSLT2   RSLT1  RSLT0
*******************************************************
XH        .EQU      R2          ;Higher operand of X
XL        .EQU      R3          ;Lower operand of X
YH        .EQU      R4          ;Higher operand of Y
YL        .EQU      R5          ;Lower operand of Y
RSLT3     .EQU      R6          ;MSbyte of the final result
RSLT2     .EQU      R7
RSLT1     .EQU      R8
RSLT0     .EQU      R9          ;LSbyte of the final result

MPY32     CLR       RSLT2       ;Clear the present value
          CLR       RSLT3
          MPY       XL,YL       ;Multiply LSbytes
          MOVW      B,RSLT0     ;Store in result register 0
          MPY       XH,YL       ;Get XHYL
          ADD       R1,RSLT1    ;Add to existing result XLYL
          ADC       R0,RSLT2    ;Add carry if present
          ADC       #0,RSLT3    ;Add if carry present
          MPY       XL,YH       ;Multiply to get XLYH
          ADD       R1,RSLT1    ;Add to existing result XLYL+XHYL
          ADC       R0,RSLT2    ;Add to existing results and carry
          ADC       #0,RSLT3    ;Add if carry present
          MPY       XH,YH       ;Multiply MSbytes
          ADD       R1,RSLT2    ;Add once again to the result register
          ADC       R0,RSLT3    ;Do the final add to the result reg
          RTS                   ;Return to call subroutine
```

14.6.12 Keyboard Scan

This routine reads a 16-key keyboard, returns the hex digit of the key, and debounces the key to avoid noise. A valid key flag is set when a new key is found.

Figure 14–14. Keyboard Scan Values



Register	Before	After No Key	After New Key	Function
A	XX	0	COLUMN	Temporary
B	XX	0	ROW	Temporary
R2	XX	16	KEY #	Temporary store for key value
R3	OLD KEY	0FFh	KEY #	Holds key pressed no
R4	DEBOUNCED	0	0	Debounce counter, old key or new
R5	GENERAL BITS	?xxxxxxx0	?xxxxxxx1	One bit of register is 1 if new key

```
0000           0001         .TEXT07000h       ;
0002           0002 FLAG    .EQU R2           ;"Swap has been made" flag
002F           0003 DDIR    .EQU P02F         ;Port D data direction register
002E           0004 DDATA   .EQU P02E         ;Port D data register
7000           0005                           ;THESE ASSIGNMENTS NEED TO BE
7000           0006                           ;DONE IN THE MAIN INITIALIZATION
7000           0007                           ;
7000 F7002E    0008 START   MOV  #00,DDATA     ;Clear these registers
7003 720005    0009         MOV  #0,R5        ;Clear register that say key found
7006 F7F02F    0010         MOV  #0F0h,DDIR   ;Set data direction register 4
7009           0011                           ;output,
7009           0012                           ;4 input
7009           0013                           ;THIS IS THE BEGINNING OF THE
7009           0014                           ;KEYBOARD SCAN ROUTINE
7009           0015                           ;
7009 5208      0016 GETKEY  MOV  #8,B         ;Initialize row pointer
700B D502      0017         CLR  R2           ;
700D CF        0018 LOOP    RLC  B            ;Select next row
700E 03**      0019         JC   NOKEY        ;Last row? if so no key was found.
7010 780402    0020         ADD  #4,R2        ;Add number of keys/row to key
7013           0021                           ;accumulator
7013 512E      0022         MOV  B,DDATA      ;Activate row
7015 802E      0023         MOV  DDATA,A      ;Read columns
7017 F7002E    0024         MOV  #0,DDATA     ;Clear row
701A 230F      0025         AND  #0Fh,A       ;Isolate column data
701C 02EF      0026         JZ   LOOP         ;If no keys found then check next
701E           0027                           ;row
701E D202      0028 KEYLSB  DEC  R2           ;Decrement column offset
7020 BD        0029         RRC  A            ;Find column
7021 07FB      0030         JNC  KEYLSB       ;If not column then, try again
7023           0031                           ;
7023 4D0203    0032 NEWKEY  CMP  R2,R3        ;Is the new key the same as the old
7026           0033                           ;key
7026 02**      0034         JEQ  DEBONS       ;If it is then debounce it
7028 420203    0035         MOV  R2,R3        ;brand new key, move it to current
702B           0036                           ;key value
702B 720704    0037         MOV  #07,R4       ;Set up debounce count, debounce 7
702E           0038                           ;times
702E 7D0204    0039 DEBONS  CMP  #2,R4         ;Is the debounce count 1 or 0?
7031 09**      0040         JL   GOODKY       ;
7033 DA04D3    0041         DJNZ R4,GETKEY    ;If greater than 1 then debounce is
7036           0042                           ;not finished, go read key again
7036 770104**  0043 GOODKY  BTJZ #01,R4,NONEW;If debounce count = 0 then key
703A           0044                           ;was here last time
703A D204      0045         DEC  R4           ;If it was one this is a new valid
703C           0046                           ;key, make old key
703C           0047                           ;
703C 740105    0048         OR   #1,R5        ;Set new key flag in BIT register,
703F           0049                           ;the
703F F9        0050         RTS               ;found new key so return to main
7040           0051                           ;calling routine uses this flag
7040 72FF03    0052 NOKEY   MOV  #0FFh,R3     ;No key was found, set key value to
7043           0053                           ;unique
7043           0054                           ;value
7043 F9        0055 NONEW   RTS               ;If jumped to NONEW it is still the
7044           0056                           ;same key
7044           0057                           ;held down do nothing
```

14.6.13 Divide 1

This routine divides a 16-bit number by an 8-bit number to give a 16-bit quotient and an 8-bit remainder. The DIV instruction accomplishes this task.

```
0000          0021           .TEXT 7000h   ;
700B          0022 OVERFLOW  .EQU R7        ;
700B          0023                         ;Divisor -R3, quotient LSbyte-R5
700B          0024                         ;Dividend MSbyte-R1, quotient
700B          0024                         ;MSbyte-R4
700B          0025                         ;Dividend LSbyte-R2, remainder -B
700B          0026                         ;uses R0
700B          0027                         ;
700B B5       0028 DIVIDE8   CLR A         ;Clear MSbyte of first dividend
700C F4F803   0029           DIV R3,A      ;Divide MSbyte of dividend to get MSbyte
700F 08**     0030           JV OVERF      ;Exit if overflow
7011 D004     0031           MOV A,R4      ;quotient. Move MSbyte of quotient
7013          0032                         ;to storage.
7013 62       0033           MOV B,A       ;Move remainder to MSbyte of dividend
7014 3202     0034           MOV R2,B      ;Move dividend LSbyte to LSbyte position
7016 F4F803   0035           DIV R3,A      ;Divide to get quotient LSbyte and
7019          0036                         ;remainder
7019 08**     0037           JV OVERF      ;Exit if overflow
701B D005     0038           MOV A,R5      ;Store the quotient LSbyte next to MSbyte
701D F9       0039           RTS           ;Remainder in B
701E          0040                         ;
701E D311     0041 OVERF     INC OVERFLOW  ;Set bit 0 if overflow
7020 F9       0042           RTS           ;
```

14.6.14 Divide 2

This program divides a 16-bit dividend by a 16-bit divisor and produces a 16-bit quotient with a 16-bit remainder. All numbers are unsigned positive integers. All values can range from 0 to FFFFh. The same principle can be applied to larger or smaller divide routines to allow different sizes of quotients, dividends, divisors, and remainders.

```
0000          0026            .TEXT 7000h
700B          0027 ;        Before                  After
700B          0028 ;     A =                     Remainder MSbyte
700B          0029 ;     B =                     Remainder LSbyte
700B          0030 ;     R2= Dividend MSbyte     Quotient  MSbyte
700B          0031 ;     R3= Dividend LSbyte     Quotient  LSbyte
700B          0032 ;     R4= Divisor  MSbyte     Divisor   MSbyte
700B          0033 ;     R5= Divisor  LSbyte     Divisor   LSbyte
700B          0034 ;     R6= XXX                 Zero
700B          0035 ;
700B          0036
700B 721006   0037 DIV16  MOV #16,R6      ;Set loop counter to 16,
700E          0038                        ;one for each quotient bit
700E B5       0039        CLR A           ;
700F C5       0040        CLR B           ;Initialize result register
7010 DF03     0041 DIVLOP RLC R3          ;Multiply dividend by 2
7012 DF02     0042        RLC R2          ;
7014 CF       0043        RLC B           ;Shift dividend into A:B for
7015 BF       0044        RLC A           ;comparison to divisor
7016 07**     0045        JNC SKIP1       ;Check for possible error
7018 3A05     0046        SUB R5,B        ;condition that results when
701A 1B04     0047        SBB R4,A        ;a 1 is shifted past the MSbyte,
701C F8       0048        SETC            ;Correct by subtracting
701D          0049                        ;divisor and setting carry.
701D 00**     0050        JMP DIVEND      ;If MSB=1 then subtract is
701F          0051                        ;possible
701F 1D04     0052 SKIP1  CMP R4,A        ;Compare MSbytes of dividend
7021          0053                        ;and divisor
7021 07**     0054        JNC DIVEND      ;Jump if divisor is bigger
7023 06**     0055        JNE MSBNE       ;If equal compare LSbytes.
7025 3D05     0056        CMP R5,B        ;Compare LSbytes.
7027 07**     0057        JNC DIVEND      ;Jump if divisor is bigger
7029 3A05     0058 MSBNE  SUB R5,B        ;If smaller, subtract divisor
702B 1B04     0059        SBB R4,A        ;from dividend. Carry gets
702D          0060                        ;folded into next rotate and
702D          0061                        ;gets doubled each time.
702D DA06E0   0062 DIVEND DJNZ R6,DIVLOP  ;Next bit, is divide done?
7030 DF03     0063        RLC  R3         ;Finish last rotate.
7032 DF02     0064        RLC  R2         ;
7034          0065
```

Chapter 15

Development Support

This chapter discusses the key features of the TMS370 development tools. These tools are currently available for PC-DOS or MS-DOS (version 3.0 and up) systems. For a detailed description of system components, refer to the documents listed in the preface.

The topics in this chapter include the following:

Topic **Page**

15.1 TMS370 Development Tools

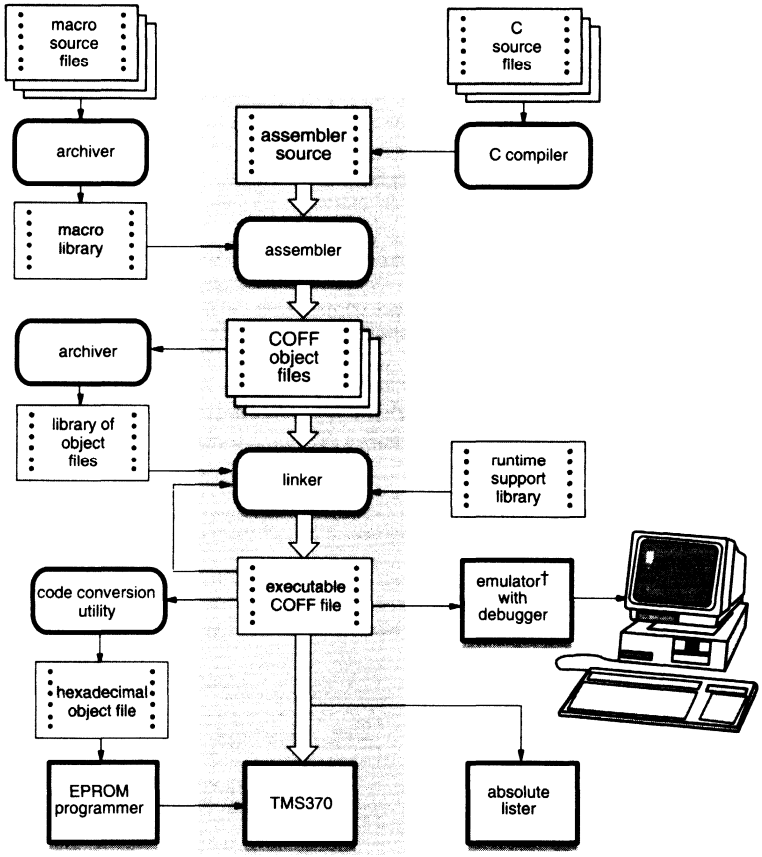
Texas Instruments provides extensive development support for the TMS370 family:

- ❑ Assembly language tools that convert assembly language into executable object code
- ❑ Additional software support tools such as an archiver and code conversion utility
- ❑ An optimizing C compiler that supports high-level language programming and is a full implementation of the standard ANSI C language
- ❑ Development support tools that offer full-speed emulation for testing your object files:
  - XDS/22 (extended development support system) that provides realtime breakpoint/trace/timing functions to facilitate hardware and software integration during system development
  - PACT XDS/22 (available in Europe only) that provides realtime breakpoint/trace/timing functions to facilitate hardware and software integration during system development for devices with the PACT module (available in Europe only)
  - CDT370 (compact development tool) that supports realtime in-circuit emulation of the TMS370 family
  - Design kit for evaluation of the TMS370 family
- ❑ A C source debugger—an advanced software interface to the TMS370 XDS/22 emulator, CDT370, and design kit
- ❑ TMS370 microcontroller programmer to program the programmable memory of any TMS370 device
- ❑ TMS370 gang programmer for programming up to sixteen devices at once.
- ❑ EPROM devices for prototype and small production runs

These development tools are designed to work with an IBM-compatible PC. Additionally, the TMS370C7xx devices prototype and emulate masked ROM parts and also act as a medium for submitting the program to TI for mask-ROM production.

Figure 15–1 shows the software development flow. The shaded portion highlights the most common development path; the other portions are optional.

Figure 15–1. Software Development Flow



† The emulator can be the XDS/22, the CDT370, or the design kit.

15.2 The Assembler

The TMS370 assembler translates assembly language source files into machine language object files. Source files can contain instructions, assembler directives, and macro directives. The assembler directives control various aspects of the assembly process, such as the source listing format, symbol definition, conditional assembly blocks, macro library definition, and the way the machine code is placed into the TMS370 memory space.

The format of the object files created by the assembler and linker is called *common object file format* (COFF). COFF encourages and facilitates modular programming. It allows the assembler to maintain a section program counter (SPC) for each section of object code generated. The SPC defines the virtual program memory addresses assigned to the associated object code. The assembler uses the SPC while it builds the symbol table.

The symbol tables contained in the COFF object files allow the C source debugger to provide you with *symbolic debugging*. The debugger also provides for direct referencing of any assembler label and arithmetic expressions involving assembler labels when the labels are part of the downloaded COFF object file. The COFF object files are also used by the TMS370 microcontroller programmer to form a PC memory image of the data loaded for programming.

15.3 The Linker

The TMS370 linker creates executable modules by combining COFF object files. The concept of user-definable COFF *sections* is basic to the linker operation. The linker accepts several types of files as input:

- Relocatable COFF object files produced by the TMS370 assembler
- Command files
- Archive object libraries
- Output modules created by a previous linker run (these are referred to as partially linked files)

As the linker combines object files, it performs the following tasks:

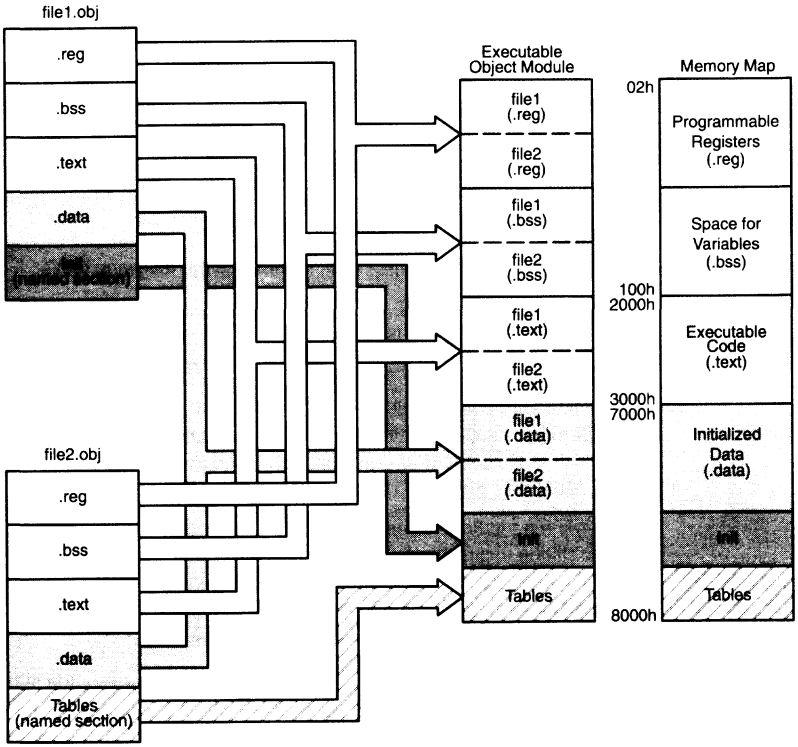
- Allocates sections into the target system's configured memory
- Relocates symbols and sections to assign them to final addresses
- Resolves undefined external references between input files

The linker supports a command language similar to C that controls memory configuration, section definition, and address binding. The language supports expression assignment and evaluation. It also provides two powerful directives, MEMORY and SECTIONS, that allow you to:

- Define a memory model that conforms to target system memory
- Combine object file sections
- Allocate sections into specific areas of memory
- Define overlayed memory structures
- Define or redefine global symbols at link time

Figure 15–2 shows the operation of the linker on two source code files. Each file has been assembled and contains four default sections and one named section. The various sections are arranged in the order dictated either by the linker's default method or by a user-supplied control file. The executable object module shows the combined sections, and the memory map indicates the location of the sections in memory.

Figure 15–2. Linker Output Generation



15.4 Additional Software Support

Included with the TMS370 assembly language tools are two addition software utilities:

- An archiver
- A code conversion utility

15.4.1 The Archiver

The archiver provides file management by allowing a group of files to be collected into a single library. For example, macros can be collected by the archiver, then fetched by the assembler as directed by the source file. In addition, object modules can be collected into a library for convenient access by the linker. While not necessary for program development, the archiver can provide valuable organization in the building of the executable COFF object file.

15.4.2 The Code Conversion Utility

The code conversion utility converts COFF object files to Tektronix, Intel, Motorola, or TI-tagged object formats. Code conversion is necessary when you are not using the TI microcontroller programmer, because some of the non-TI EPROM programmers do not accept COFF object files as input. Code in the Intel hex object format can be downloaded to most EPROM programmers.

15.5 The Optimizing C Compiler

The TMS370 optimizing C compiler translates the widely used ANSI C language directly into highly optimized assembly language, enhancing productivity by enabling you to program in C. C code is easier to prototype, debug, and benchmark than assembly language. Also, it produces a rich set of information that is used by the debugger, which allows source-level debugging in C and assembly. This shortens the development cycle for TMS370 applications.

Key features of the C compiler include:

- ❑ Conformance with the ANSI C specification
- ❑ Highly efficient code—the C compiler incorporates state-of-the-art generic and target-specific optimizations
- ❑ ANSI standard runtime-support library
- ❑ A C shell program that facilitates one-step translation from C source to executable code
- ❑ ROM-able, relocatable, and re-entrant code
- ❑ A source interlist utility that can interlist your original C source statements into assembly language output of the compiler
- ❑ A utility for building object libraries from source libraries
- ❑ Fast compilation to increase productivity

The following is a list of key optimizations by the compiler:

- ❑ Performs control-flow graph simplification
- ❑ Allocates variables to registers
- ❑ Performs loop rotation
- ❑ Eliminates dead code
- ❑ Simplifies expressions and statements
- ❑ Performs local copy/constant propagations
- ❑ Removes dead assignments
- ❑ Eliminates local common expressions
- ❑ Performs loop optimizations expressions
- ❑ Eliminates global common expressions
- ❑ Eliminates global dead assignments
- ❑ Performs loop unrolling

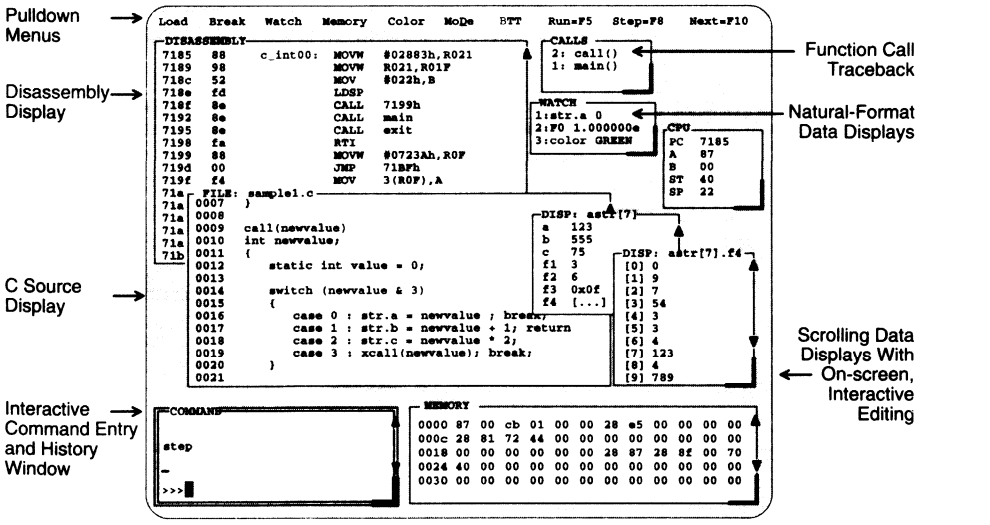
15.6 The C Source Debugger

The C source debugger is an advanced software interface that runs on an IBM-compatible PC and helps you to develop, test, and refine TMS370 C programs (compiled with the TMS370 optimizing ANSI C compiler) and assembly language programs. The debugger is the interface to the TMS370 in-circuit XDS/22 emulator and design kit. The debugger has a standard screen appearance for the supported products, allowing you to move from one tool to the next without having to learn a new interface—only the new features.

The C source debugger improves productivity by enabling you to debug a program in the language it was written in. You can choose to debug your programs in C, assembly language, or both. Also, the debugger is easy to learn and use. Its friendly window-, mouse-, and menu-oriented interface reduces learning times and eliminates the need to memorize complex commands.

Figure 15–3 identifies several features of the debugger display.

Figure 15–3. The Basic Debugger Display

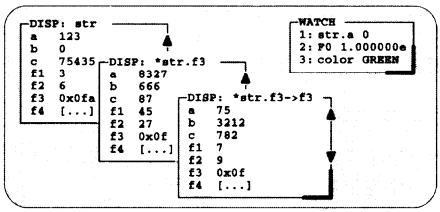


Key features of the C source debugger include:

- **Multilevel debugging**. The debugger allows you to debug both C and assembly language code. If you're debugging a C program, you can choose to view just the C source, the disassembly of the object code created from the C source, or both. You can also use the debugger as an assembly language debugger.

- **Fully configurable, state-of-the-art, window-oriented interface.** The C source debugger separates code, data, and commands into manageable portions. Use any of the default displays. Or select the windows you want to display, size them, and move them where you want them.

- **Comprehensive data displays.** You can easily create windows for displaying *and editing* the values of variables, arrays, structures, pointers—any kind of data—in their natural format (*float*, *int*, *char*, *enum*, or *pointer*). You can even display entire linked lists.



- **On-screen editing.** Change any data value displayed in any window—just point the mouse, click, and type.

- **Continuous update.** The debugger continuously updates information on the screen, highlighting changed values.

- **Powerful command set.** The C source debugger supports a small but powerful command set that makes full use of C expressions. One debugger command performs actions that would require several commands in another system.

- **Flexible command entry.** There are a variety of ways to enter commands. You can type commands or use a mouse, function keys, or the pulldown menus; choose the method that you like best.

- **Create your own debugger.** The debugger display is completely configurable, allowing you to create the interface that is best suited for your use.

- **Variety of screen sizes.** The debugger's default configuration is set up for a typical PC display, with 25 lines by 80 characters. If you use a sophisticated graphics card, you can take advantage of the debugger's additional screen sizes. A larger screen size allows you to display more information and provides you with more screen space for organizing the display—bringing the benefits of workstation displays to your PC.

- **All the standard features you expect in a world-class debugger.** The debugger provides you with complete control over program execution with features like conditional execution and single-stepping (including single-stepping into or over function calls). You can set or clear a breakpoint with a click of the mouse or by typing commands. You can define a memory map that identifies the portions of target memory that the debugger can access. You can choose to load only the symbol table portion of an object file to work with systems that have code in ROM. The debugger can execute commands from a batch file, providing you with an easy method for entering often-used command sequences.

15.7 Breakpoint, Trace, and Timing Functions

Included with the XDS/22 emulation system is a separate board called the BTT board, which provides breakpoint, trace, and timing features. The BTT monitors the TMS370 CPU; when a preselected pattern of bus activity is detected, the BTT performs an action such as executing a hardware breakpoint or storing information in the trace buffer.

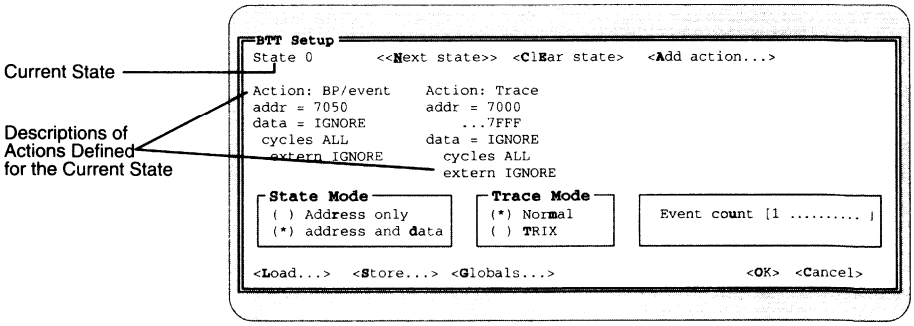
The BTT supports a rich set of features:

- **Full range of actions.** The BTT allows you to set hardware breakpoints, to count event occurrences, to collect trace samples, to jump to a BTT state, or to start/stop timers. These actions occur when they are *qualified*—that is, when bus activity matches conditions that you have defined.

- **Four separate states.** The BTT supports four separate states, called state 0–state 3. Each state can be associated with up to four actions. You can define actions for as many states as you need. By default, the BTT will cycle through the states, beginning with state 0 and ending with the last state that you defined actions for. You can control this sequencing by jumping to another state or by using counters to loop through a sequence of states.

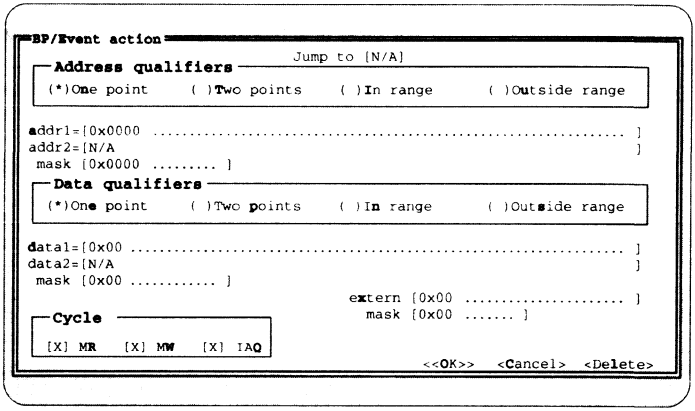
  To view information about specific states, add an action, delete an action, or access global settings, use the BTT Setup dialog box (see Figure 15–4).

Figure 15–4. The BTT Set-Up Dialog Box



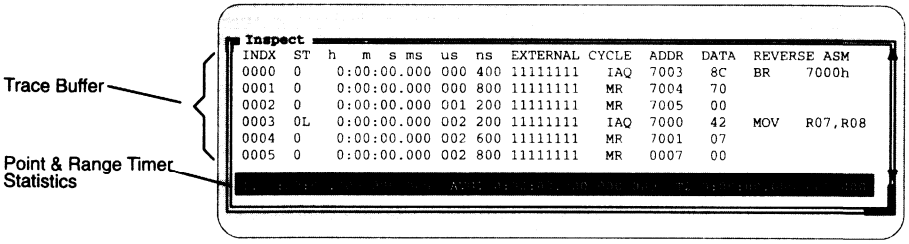
- **Flexible qualification for actions.** You can qualify actions according to address or data values (either singly or in relation to ranges) and to the memory-cycle type. You can combine these conditions; for example, you could qualify an action whenever a certain data value is accessed during an instruction acquisition cycle. You can define conditions or qualify actions by using a dialog box (refer to Figure 15–5).

Figure 15–5. The Dialog Box for Defining Conditions



- **Informative trace reporting.** The BTT can store up to 2047 trace samples in the trace buffer. You can display the trace samples and associated information by opening the INSPECT window (see Figure 15–6).

Figure 15–6. An Example of the INSPECT Window



- **External signal access.** The BTT has eight external probes that can be connected to eight signals. You can qualify actions by looking for a particular pattern of activity on these probes. Additionally, the trace buffer reports the values that were on the signals when each trace sample was collected.

- **Complete timing analysis.** The BTT supports two timers that you can start and stop on a variety of conditions. The BTT reports the total time for each of these timers; it also reports the average for one of the timers. Additionally, the BTT collects timing information that relates specifically to the samples in the trace buffer.

- **External filing.** Once you have defined a complex BTT set-up, you may want to reuse the set-up. The BTT allows you to save the set-up to a file and then load it again for a later session. You can also save the contents of the trace buffer to a file for later use or to compare to another trace collection.

15.8 The XDS/22 System

The XDS/22 system is a self-contained package that provides full-speed, in-circuit emulation and debugging functions required for program development of the TMS370 family devices. These are key features of the XDS/22 emulation function:

- 2 to 20-MHz full-speed in-circuit emulation of all TMS370 family members
- Realtime hardware breakpoint/trace/timing analysis capabilities
- Execution of programs from internal XDS/22 memory (64K byte) or target memory
- Support of both microcomputer and microprocessor modes
- Large trace buffer, 2047 samples
- Full logic tracing with logic analyzer interface cable

The XDS/22 system set of boards consists of an emulator, communications board, and a breakpoint/trace/timing board. At the heart of the XDS/22 system is a special system emulator chip containing all of the peripheral modules and I/O line circuits that precisely duplicate the TMS370's logic and performance. You can use the internal XDS/22 memory to emulate on-chip ROM and/or external memory.

The target cables that are necessary for using the XDS/22 are sold separately. See subsection 17.4.4, page 17-20, for ordering information.

The XDS/22 is supported by a debugger, as described in Section 15.6.

15.8.1 The PACT XDS/22 System (Available in Europe Only)

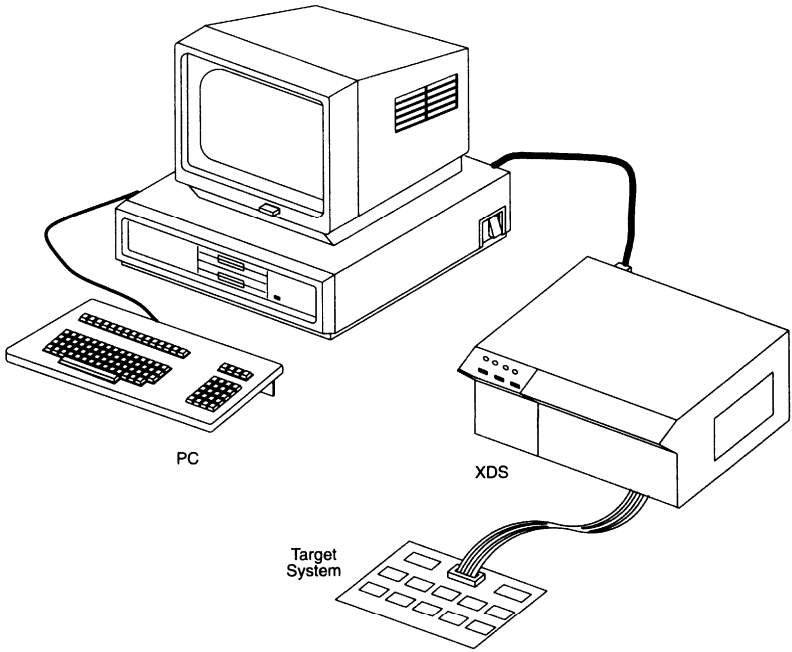
The PACT XDS/22 system is specifically developed for emulating and debugging the TMS370Cx3x devices with the PACT module.

15.8.2 XDS System Configuration Requirements

A functional XDS system configuration consists of the XDS system and the following user-supplied components:

- ❑ IBM-compatible PC with a minimum of 640K bytes, a serial communication port, and a 5-1/4-inch high-density disk drive
- ❑ MS-DOS or PC-DOS, version 3.0 or later
- ❑ Monitor (preferably color, to better highlight field and value changes)

Figure 15–7. Typical XDS System Configuration



15.8.3 XDS System Operating Considerations

The emulation hardware of the XDS systems (XDS/22 and PACT XDS/22) generally exhibits the same characteristics as the actual TMS370 devices. There are, however, a few subtle differences that you should be aware of when building a prototype circuit for use with the XDS system.

- **Mode Control pin.** To allow an XDS system to function without being attached to a target system, a 20-kΩ pulldown resistor is connected to the mode control line in the XDS unit. This increases the minimum input current needed to drive this line high (I_I) from 100 μA to 300 μA. If you are using a pull-up resistor to put the device in the microprocessor mode, then its value should be no greater than 1 kΩ.

- **Reset.** The XDS systems add an analog switch and a 51-kΩ pull-up resistor to the reset line. This increases the current necessary to pull this line to a logic low from 10 μA to 100 μA.

- **Clock in.** The XDS systems cannot drive a crystal located on the target system. Therefore, either the crystal must be moved to socket Y1 of the emulator board, or an external clock signal must be connected to the XTAL2/CLKIN pin. The external clock signal must meet the V_{IH} specifications for the XTAL2/CLKIN described in Chapter 16.

15.9 The CDT370 (Compact Development Tool)

The CDT370 supports real-time in-circuit emulation of most of the TMS370 family devices. It offers a low-cost, highly efficient route to TMS370 family development of software and hardware with the target system.

The CDT370 contains a single board, called an emulator, which can be plugged into the expansion chassis of any IBM-compatible PC or connected through the RS-232 serial communication link. Attached to the emulator is a target cable with the same pinout as the system's circuit board; the cable uses the same socket that would normally hold the TMS370 microcontroller.

The CDT370 uses a debugger interface that is similar to that of the XDS/22 systems.

With the CDT370, you can:

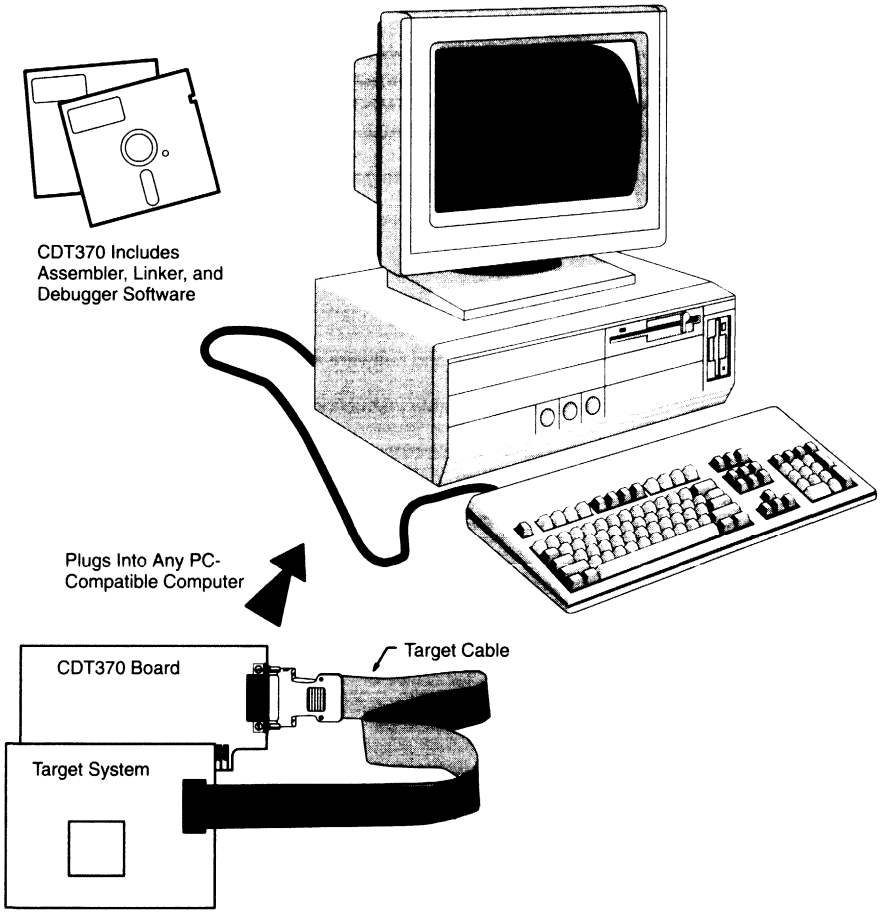
- ❑ Program the data EEPROM and program EPROM contained within the TMS370Cx1x, TMS370Cx2x, TMS370Cx4x, and TMS370Cx5x devices
- ❑ Inspect and modify memory locations
- ❑ Upload/download program and data memory
- ❑ Execute programs and software routines
- ❑ Use a large trace buffer with 1,024 samples
- ❑ Single-step executable instructions
- ❑ Use software breakpoints to halt program execution at selected addresses

The CDT package is shown in Figure 15–8 and comes complete with:

- ❑ A CDT370 emulator board
- ❑ An assembler and linker
- ❑ The C source debugger

The cables that you need to use the CDT370 are sold separately. See subsection 17.4.5, page 17-20, for ordering information.

Figure 15–8. CDT370 Configuration



15.10 The Design Kit

The TMS370 design kit helps you to quickly assess the feasibility of using a member of the TMS370 family for your application. This low-cost evaluation tool lets you analyze the hardware and software capabilities of the TMS370 family by actually using the TMS370 devices. However, the design kit cannot be used for evaluating the PACT module on TMS370Cx3x parts.

The design kit package allows you to:

- ❑ Upload and download code
- ❑ Access to any register or memory location
- ❑ Read and modify memory locations
- ❑ Execute programs and software routines
- ❑ Single-step executable instructions
- ❑ Use software breakpoints to halt program execution at selected addresses
- ❑ Program the EEPROM and EPROM contained within the any of the TMS370 family devices, except TMS370Cx58 devices, TMS370Cx3x devices, and any devices that have the hard watchdog option. For devices other than the TMS370Cx1x and TMS370Cx5x devices, you must wire-wrap your own socket to the board.
- ❑ Use the assembler on a PC
- ❑ Use a wire-wrap protoarea
- ❑ Use a patch assembler in the debugger mode
- ❑ Use a reverse assembler

15.10.1 Operating Modes

The application board operates in one of three modes:

- **Debugger mode.** The debugger mode is the standard TI programmer's interface. There are many advantages to using the debugger mode of operation:

  - Multilevel debugging
  - Powerful command set
  - Window-oriented interface
  - Comprehensive data displays
  - On-screen editing
  - Patch assembly
  - Direct COFF download

  To use the debugger mode of the design kit, the PC must meet the XDS system configuration requirements as discussed in subsection 15.8.2. For more information on the debugger function, refer to Section 15.6.

- **TTY mode.** The TTY mode allows you to communicate using an ASCII serial protocol with equipment such as "dumb" terminals, PCs running terminal interface programs (for example, CROSSTALK or PROCOMM), or computers that are not PC-compatible. This mode is most useful when you don't have access to a PC or if you need to do only simple operations or programming. The TTY mode has several two-letter commands that allow you to check the operation of an application board.

  Additionally, you can perform the following memory operations in the TTY mode; these operations aren't available in the debugger mode:

  - Run system tests
  - Upload memory
  - Verify memory
  - Find bytes
  - Move memory
  - Compare memory
  - Block program
  - Directly read the analog pin
  - Load on-chip EEPROM from on-board UVEPROM
  - Use a computer in "dumb" terminal mode
  - Reverse assemble with cycle times

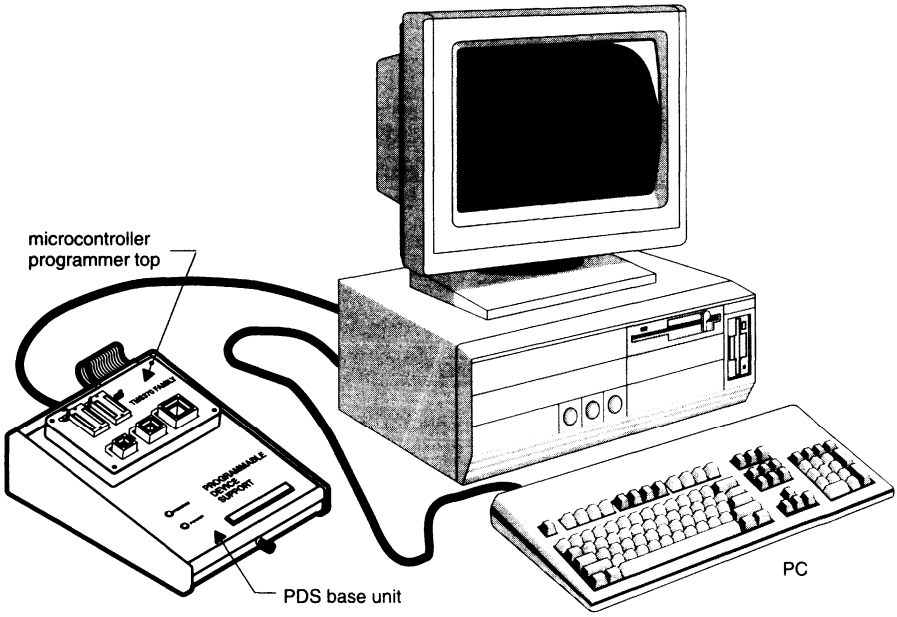
- **Isolated mode.** After you have written and debugged a program for a slave, you can check the accuracy of that program by using the isolated mode. This mode "turns off" all debugger software and allows the slave device to run on its own.

15.11 The Microcontroller Programmer

The TMS370 microcontroller programmer is an interactive, menu-driven system that provides a method of programming TMS370 family devices and EPROMs directly or through an XDS. The microcontroller programmer is currently capable of programming the TMS370, TMS77C82, TMS27C128, and TMS27C256 devices. To program the TMS77C82 devices, you will need a 40-pin to 28-pin converter, which is sold separately. To program 64-pin SDIP devices, you will need a 64-pin to 28-pin adapter, which is also sold separately.

The TMS370 microcontroller programmer system (as shown in Figure 15–9) consists of a PDS (programmable device support) base unit, a microcontroller programmer top, and an IBM-compatible PC running microcontroller programmer software under MS/PC-DOS.

Figure 15–9. Typical TMS370 Microcontroller Programmer Configuration



The programmer top and PDS base unit are sold separately. Two types of single-unit tops are available for the packages listed:

TMS370 Single-Unit DIP Top	TMS370 Single-Unit PLCC Top
28-Pin PDIP	28-Pin PDIP
40-Pin PDIP	28-Pin PLCC
40-Pin SDIP	44-Pin PLCC
	68-Pin PLCC

The programmer software provides both interactive and limited batch control with the following features:

- ❑ Window-oriented screens with a menu-driven command structure
- ❑ Intermediate PC memory, which provides a storage area for downloading a COFF file for uploading from device. This allows you to inspect and patch loaded data.
- ❑ Programming mode bit selection for TMS370 family devices only
- ❑ Relocatable programming capability, which allows source data bytes within a certain address range to be programmed at a specified address
- ❑ Reverse assembly code display
- ❑ Ability to generate a COFF file from PC memory content
- ❑ User-defined device types that allow new family members
- ❑ Ability to save or load programmer configuration to or from configuration/ batch files

The TMS370 microcontroller programmer, unlike most other EPROM programmers, can use COFF object files developed by the assembler/linker as input for programming the TMS370 devices; most other EPROM programmers require that the object files be converted into some other object format before programming.

15.12 The Gang Programmer

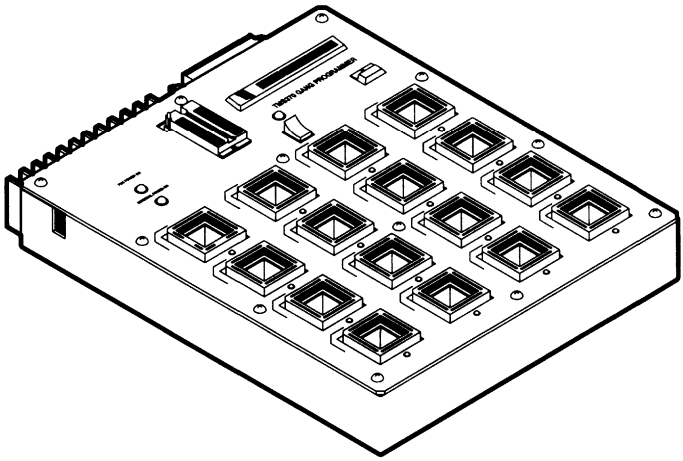
The TMS370 gang programmer is an interactive, menu-driven system that provides programming support for on-chip EEPROM or EPROM of TMS370 microcontrollers in a production environments.

The gang programmer has the following features:

- ❑ Two modes of operation—PC mode and standalone mode
- ❑ Ability to program up to 16 devices
- ❑ LEDs that indicate programming or verification failure
- ❑ A buzzer that indicates programming completion

Figure 15–10 illustrates a gang programmer.

Figure 15–10. Typical TMS370 Gang Programmer Board



The gang programmer consists of the standard programmer base, a gang programmer top, and the standard programmer software. If you already have a standard TMS370 microcontroller programmer, you can purchase the gang programmer top separately.

The gang programmer top and base unit are sold separately. Individual tops are available for the packages listed:

28-Pin PDIP	40-Pin SDIP	44-Pin PLCC
40-Pin PDIP	28-Pin PLCC	68-Pin PLCC

15.13 Reprogrammable EPROM and OTP Devices

In the TMS370 one-time programmable (OTP) and SE370 reprogrammable EPROMs devices, the program ROM has been replaced by a programmable program memory, such as EPROM.

- The TMS370 OTP devices are in plastic packages and can be programmed one time. The OTP device is an effective microcomputer to use for immediate production updates for other members of the TMS370 family or for low-volume production runs that cannot satisfy minimum volume or cycle times for low-cost mask-ROM devices.
- The SE370 reprogrammable EPROM devices are in windowed ceramic packages to allow reprogramming of the EPROM memory during the prototype development phase of design. This capability provides form factor preproduction parts with zero lead time for field testing and production qualifications, thereby reducing the overall time to market. This also supports applications with small production runs. You can program all TMS370 devices directly from the assembler or linker output file with the TMS370 microcontroller programmer. The application board can program the corresponding devices for which it demonstrates the capabilities.

Table 15–1 shows the TMS370 ROM devices and their corresponding OTP devices and reprogrammable EPROM devices.

Table 15–1. OTP and Reprogrammable EPROM Support of ROM Devices

TMS370 ROM Device	OTP EPROMs	Reprogrammable EPROMs
TMS370C010 TMS370C310 TMS370C311	TMS370C710 TMS370C610 TMS370C610	SE370C710
TMS370C020 TMS370C320 TMS370C022 TMS370C322	TMS370C722 TMS370C622 TMS370C722 TMS370C622	SE370C722
TMS370C032 TMS370C332	TMS370C732 TMS370C732	SE370C732
TMS370C040 TMS370C340 TMS370C042 TMS370C342	TMS370C742 TMS370C642 TMS370C742 TMS370C642	SE370C742
TMS370C050 TMS370C350 TMS370C052 TMS370C352 TMS370C056 TMS370C356	TMS370C756 TMS370C756 TMS370C756 TMS370C756 TMS370C756 TMS370C756	SE370C756
TMS370C058 TMS370C358	TMS370C758 TMS370C758	SE370C758

Note: ROM-less devices (TMS370C15x, TMS370C25x) do not need reprogrammable EPROM devices

Chapter 16

Electrical Specifications and Timings

This chapter contains electrical and timing information for the TMS370 family devices. This information is presented according to device category.

Topic **Page**

16.1 Timing Parameter Symbols

Throughout this chapter, timing parameter symbols have been created in accordance with JEDEC standard 100. In order to shorten the symbols, some of the pin names and other related terminology have been abbreviated as follows:

A	Address	R	Read
AR	Array	RXD	SCIRXD
B	Byte	S	Slave mode
CI	XTAL2/CLKIN	SCC	SCICLK
CO	CLKOUT	SIMO	SPISIMO
D	Data	SOMI	SPISOMI
E	$\overline{EDS}$	SPC	SPICLK
FE	Final	TXD	SCITXD
IE	Initial	W	Write
PGM	Program	WT	$\overline{WAIT}$

Lowercase subscripts and their meanings are:

c	Cycle time (period)	su	Setup time
d	Delay time	v	Valid time
f	Fall time	w	Pulse duration (width)
r	Rise time	x	Oscillator
h	Hold time		

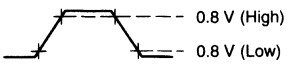
The following additional letters are used with these meanings:

H	High	V	Valid
L	Low	Z	High impedance

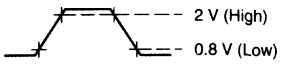
16.2 Parameter Measurements

All timings are calculated between high and low measurement points as indicated in Figure 16–1.

Figure 16–1. Measurement Points for Timings



XTAL2/CLKIN Measurement Points



General Measurement Points

16.3 Absolute Maximum Ratings for All TMS370 Devices

For all TMS370 devices, Table 16–1 provides the absolute maximum ratings over an operational free-air temperature range, unless otherwise noted.

Stresses beyond those listed in Table 16–1 may cause permanent damage to the device. This is a stress rating only.

Table 16–1. Absolute Maximum Ratings Over Operational Free-Air Temperature Range

Parameter	Maximum Rating
Supply voltage range, V_{CC} (see Note 1)	–0.6 V to 7 V
Input voltage range: All pins except MC MC	–0.6 V to 7 V –0.6 V to 14 V
Input clamp current, I_{IK} ($V_I < 0$ or $V_I > V_{CC}$)	± 20 mA
Output clamp current, I_{OK} ($V_O < 0$ or $V_O > V_{CC}$)	± 20 mA
Continuous output current per buffer, I_O ($V_O = 0$ to V_{CC}) (see Note 2)	± 10 mA
Maximum I_{CC} current	170 mA
Maximum I_{SS} current	–170 mA
Continuous power dissipation for TMS370Cx1xA devices	500 mW
Continuous power dissipation for TMS370Cx3xA devices	800 mW
Continuous power dissipation for TMS370Cx2xA, TMS370Cx4xA, and TMS370Cx5xA devices	1 W
Storage temperature range	–65 °C to 150 °C

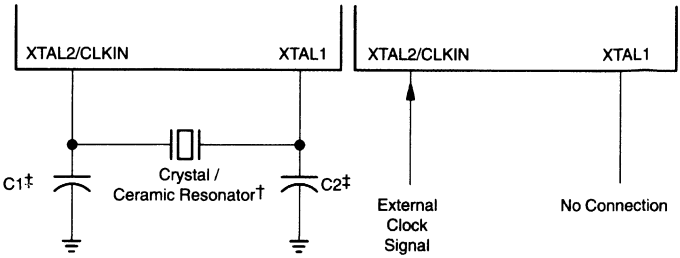
Notes: 1) Unless otherwise noted, all voltage values are with respect to V_{SS}.

2) Electrical characteristics are specified with all output buffers loaded with the specified I_O current. Exceeding the specified I_O current in any buffer may affect the levels on other buffers.

16

Figure 16–2 illustrates how to connect the crystal/ceramic resonator and the external clock signal. This figure is valid for all TMS370 family devices.

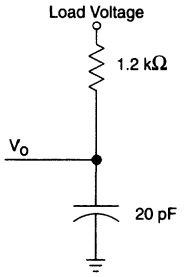
Figure 16–2. Recommended Crystal/Clock Connections



† The crystal/ceramic resonator frequency is four times the reciprocal of the system clock period.
‡ The values of C1 and C2 are typically 15 pF. See the manufacturer's recommendations for ceramic resonators.

Figure 16–3 illustrates an output load circuit that you can use for any TMS370 device.

Figure 16–3. Typical Output Load Circuit†



Case 1: $V_O = V_{OH} = 2.4$ V; Load Voltage = 0 V
Case 2: $V_O = V_{OL} = 0.4$ V; Load Voltage = 2.1 V

† All measurements are made with the pin loading as shown unless otherwise noted. All measurements are made with XTAL2/CLKIN driven by an external square wave signal with a 50% duty cycle and rise and fall times less than 10 ns unless otherwise stated.

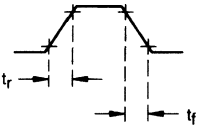
16.4 General-Purpose Output Signal Timings

Refer to Sections 16.1 and 16.2 for timing symbol definitions and parameter measurement points. The timings shown in this section are valid for all TMS370 family devices.

Table 16–2. General-Purpose Output Signal Switching Time Requirements

Parameter		Min	Nom	Max	Unit
t_r	Rise time		30		ns
t_f	Fall time		30		ns

Figure 16–4. Switching Time Measurement Points



16.5 EPROM/EEPROM Specifications

Refer to Sections 16.1 and 16.2 for timing symbol definitions and parameter measurement points. The timings shown in this section are valid for all TMS370CxxxA family devices, except the TMS370C3xxA devices.

Table 16–3. Recommended EEPROM Timing Requirements for Programming

Parameter		Min	Nom	Max	Unit
$t_{w(PGM)B}$	Programming signal pulse duration to insure valid data is stored (byte mode)	10			ms
$t_{w(PGM)AR}$	Programming signal pulse duration to insure valid data is stored (array mode)	20			ms

Table 16–4. Recommended EPROM Operating Conditions for Programming

Parameter		Min	Nom	Max	Unit
V_{CC}	Supply voltage	4.75	5.5	6	V
V_{PP}	Supply voltage at MC pin	12	12.5	13	V
I_{PP}	Supply current at MC pin during programming (V_{PP} = 13 V)		30	50	mA
CLKIN	Operating crystal frequency	2		20	MHz

Table 16–5. Recommended EPROM Timing Requirements for Programming

Parameter		Min	Nom	Max	Unit
$t_{w(EPGM)}$	Programming signal pulse duration†	0.95	2	30	ms

† Programming signal is active when both EXE (EPCTL.0) and VPPS (EPCTL.6) are set.

16.6 TMS370Cx1xA Specifications

16

The tables in this section give specifications that apply to the devices in the TMS370Cx1xA category. These devices include the TMS370C010A, TMS370C310A, TMS370C311A, TMS370C610A, TMS370C710A, and SE370C710A.

16.6.1 TMS370Cx1xA Electrical Specifications

Functional operation of the device at maximum or any other conditions beyond those indicated in Table 16–6 is not implied. Exposure to absolute maximum rated conditions for extended periods may affect device reliability.

Table 16–6. Recommended Operating Conditions

Parameter			Min	Nom	Max	Unit
V_{CC}	Supply voltage (see Note 1)		4.5	5	5.5	V
V_{CC}	RAM data retention supply voltage (see Note 2)		3		5.5	V
V_{IL}	Low-level input voltage	All pins except MC	V_{SS}		0.8	V
		MC, normal operation	V_{SS}		0.3	
V_{IH}	High-level input voltage	All pins except MC, XTAL2/CLKIN, and $\overline{RESET}$	2		V_{CC}	V
		XTAL2/CLKIN	0.8 V_{CC}		V_{CC}	
		$\overline{RESET}$	0.7 V_{CC}		V_{CC}	
V_{MC}	MC (mode control) voltage	EEPROM write protect override (WPO)	11.7	12	13	V
		EPROM programming voltage (V_{PP})	12	12.5	13	
		Microcomputer	V_{SS}		0.3	
T_A	Operating free-air temperature	A version	–40		85	°C
		L version	0		70	°C

Notes: 1) Unless otherwise noted, all voltages are with respect to V_{SS}.

2) To guarantee RAM data retention from 3.0 V to 4.5 V, $\overline{RESET}$ must be externally asserted and released only while V_{CC} is within the recommended operating range of 4.5 V to 5.5 V.

Table 16–7. Electrical Characteristics Over Full Ranges of Recommended Operating Conditions

Parameter			Test Conditions	Min	Typ	Max	Unit
V_{OL}	Low-level output voltage		I_{OL} = 1.4 mA			0.4	V
V_{OH}	High-level output voltage		I_{OH} = –50 µA	0.9 V_{CC}			V
			I_{OH} = –2 mA	2.4			
I_I	Input current	MC	0 V ≤ V_I ≤ 0.3 V			10	µA
			0.3 V < V_I ≤ 13 V			650	
			Note 1 12 V ≤ V_I ≤ 13 V			50	mA
		I/O pins	0 V ≤ V_I ≤ V_{CC}			± 10	µA
I_{OL}	Low-level output current		V_{OL} = 0.4 V	1.4			mA
I_{OH}	High-level output current		V_{OH} = 0.9 V_{CC}	–50			µA
			V_{OH} = 2.4 V	–2			mA
I_{CC}	Supply current (operating mode) OSC POWER bit = 0 (see Note 4)		Notes 2 and 3 CLKIN = 20 MHz		20	36	mA
			Notes 2 and 3 CLKIN = 12 MHz		13	25	
			Notes 2 and 3 CLKIN = 2 MHz		5	11	
I_{CC}	Supply current (standby mode) OSC POWER bit = 0 (see Note 5)		Notes 2 and 3 CLKIN = 20 MHz		10	17	mA
			Notes 2 and 3 CLKIN = 12 MHz		6.5	11	
			Notes 2 and 3 CLKIN = 2 MHz		2	3.5	
I_{CC}	Supply current (standby mode) OSC POWER bit = 1 (see Note 6)		Notes 2 and 3 CLKIN = 12 MHz		4.5	8.6	mA
			Notes 2 and 3 CLKIN = 2 MHz		1.5	3.0	
I_{CC}	Supply current (halt mode)		Note 2 XTAL2/CLKIN < 0.2 V		1	30	µA

Notes: 1) Input current I_{PP} will be a maximum of 50 mA only when you are programming EPROM.

2) Single-chip mode, ports configured as inputs or outputs with no load. All inputs ≤ 0.2 V or ≥ V_{CC} – 0.2 V.

3) XTAL2/CLKIN is driven with an external square wave signal with 50% duty cycle and rise and fall times less than 10 ns. Currents may be higher with a crystal oscillator. At 20 MHz, this extra current = .01 mA × (total load capacitance + crystal capacitance in pF).

4) Maximum operating current for TMS370Cx1xA = 1.4 (CLKIN) + 8 mA.

5) When OSC POWER bit = 0, maximum standby current for TMS370Cx1xA = 0.75 (CLKIN) + 2 mA.

6) When OSC POWER bit = 1, maximum standby current for TMS370Cx1xA = 0.56 (CLKIN) + 1.9 mA. Bit is valid only from 2 MHz to 12 MHz.

16.6.2 TMS370Cx1xA Timings

Refer to Sections 16.1 and 16.2 for timing symbol definitions and parameter measurement points.

Table 16–8. External Clocking Requirements†

No.	Parameter		Min	Nom	Max	Unit
1	$t_{w(CI)}$	XTAL2/CLKIN pulse duration‡	20			ns
2	$t_{r(CI)}$	XTAL2/CLKIN rise time			30	ns
3	$t_{f(CI)}$	XTAL2/CLKIN fall time			30	ns
	CLKIN	Crystal operating frequency	2		20	MHz

† For V_{IL} and V_{IH}, refer to Table 16–6, page 16-6.
‡ This pulse may be either a high pulse, as illustrated, which extends from the earliest valid high to the final valid high in an XTAL2/CLKIN cycle, or a low pulse, which extends from the earliest valid low to the final valid low in an XTAL2/CLKIN cycle.

Figure 16–5. External Clock Timing

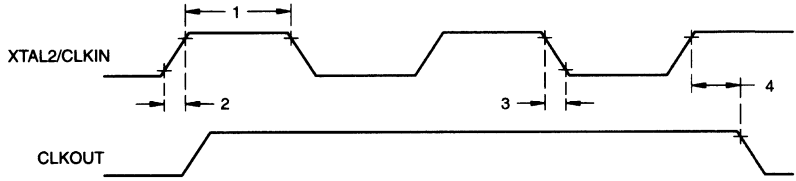
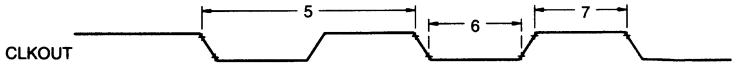


Table 16–9. Switching Characteristics and Timing Requirements†

No.	Parameter		Min	Max	Unit
4	$t_{d(CIH-COL)}$	Delay time, XTAL2/CLKIN rise to CLKOUT fall		100	ns
5	t_c	CLKOUT (system clock) cycle time	200	2000	ns
6	$t_{w(COL)}$	CLKOUT low pulse duration	$0.5t_c$–20	$0.5t_c$	ns
7	$t_{w(COH)}$	CLKOUT high pulse duration	$0.5t_c$	$0.5t_c$+20	ns

† t_c = system clock cycle time = 4/CLKIN.

Figure 16–6. CLKOUT Timing



16.7 TMS370Cx2xA Specifications

The tables in this section give specifications that apply to the devices in the TMS370Cx2xA category. These devices include the TMS370C020A, TMS370C022A, TMS370C320A, TMS370C322A, TMS370C622A, TMS370C722A, and SE370C722A.

16.7.1 TMS370Cx2xA Electrical Specifications

Functional operation of the device at maximum or any other conditions beyond those indicated in Table 16–10 is not implied. Exposure to absolute maximum rated conditions for extended periods may affect device reliability.

Table 16–10. Recommended Operating Conditions

Parameter			Min	Nom	Max	Unit
V_{CC}	Supply voltage (see Note 1)		4.5	5	5.5	V
V_{CC}	RAM data retention supply voltage (see Note 2)		3		5.5	V
V_{IL}	Low-level input voltage	All pins except MC	V_{SS}		0.8	V
		MC, normal operation	V_{SS}		0.3	
V_{IH}	High-level input voltage	All pins except MC, XTAL2/CLKIN, and $\overline{RESET}$	2		V_{CC}	V
		XTAL2/CLKIN	0.8 V_{CC}		V_{CC}	
		$\overline{RESET}$	0.7 V_{CC}		V_{CC}	
V_{MC}	MC (mode control) voltage	EEPROM write protect override (WPO)	11.7	12	13	V
		EPROM programming voltage (V_{PP})	12	12.5	13	
		Microcomputer	V_{SS}		0.3	
T_A	Operating free-air temperature	A version	–40		85	°C
		L version	0		70	°C

Notes: 1) Unless otherwise noted, all voltages are with respect to V_{SS}.

2) $\overline{RESET}$ is externally released while V_{CC} is within the recommended operating range of 4.5 V to 5.5 V and is externally activated when V_{CC} <4.5 or V_{CC} > 5.5 V. RAM data retention is valid in the 2-MHz to 20-MHz frequency range. An active $\overline{RESET}$ initializes (clears) RAM location 0000h and 0001h.

16

Table 16–11. Electrical Characteristics Over Full Ranges of Recommended Operating Conditions

Parameter			Test Conditions	Min	Typ	Max	Unit
V_{OL}	Low-level output voltage		I_{OL} = 1.4 mA			0.4	V
V_{OH}	High-level output voltage		I_{OH} = –50 µA	0.9 V_{CC}			V
			I_{OH} = –2 mA	2.4			
I_I	Input current	MC	0 V < V_I ≤ 0.3 V			10	µA
			0.3 V < V_I ≤ 13 V			650	
			Note 1 12 V ≤ V_I ≤ 13 V			50	mA
		I/O pins	0 V ≤ V_I ≤ V_{CC}			± 10	µA
I_{OL}	Low-level output current		V_{OL} = 0.4 V	1.4			mA
I_{OH}	High-level output current		V_{OH} = 0.9 V_{CC}	–50			µA
			V_{OH} = 2.4 V	–2			mA
I_{CC}	Supply current (operating mode) OSC POWER bit = 0 (see Note 4)		Notes 2 and 3 CLKIN = 20 MHz		30	45	mA
			Notes 2 and 3 CLKIN = 12 MHz		20	30	
			Notes 2 and 3 CLKIN = 2 MHz		7	11	
I_{CC}	Supply current (standby mode) OSC POWER bit = 0 (see Note 5)		Notes 2 and 3 CLKIN = 20 MHz		10	17	mA
			Notes 2 and 3 CLKIN = 12 MHz		8	11	
			Notes 2 and 3 CLKIN = 2 MHz		2	3.5	
I_{CC}	Supply current (standby mode) OSC POWER bit = 1 (see Note 6)		Notes 2 and 3 CLKIN = 12 MHz		6	8.6	mA
			Notes 2 and 3 CLKIN = 2 MHz		2	3.0	
I_{CC}	Supply current (halt mode)		Note 2 XTAL2/CLKIN < 0.2 V		2	30	µA

Notes: 1) Input current I_{PP} will be a maximum of 50 mA only when you are programming EPROM.

2) Single-chip mode, ports configured as inputs or outputs with no load. All inputs ≤ 0.2 V or ≥ V_{CC} – 0.2 V.

3) XTAL2/CLKIN is driven with an external square wave signal with 50% duty cycle and rise and fall times less than 10 ns. Currents may be higher with a crystal oscillator. At 20 MHz, this extra current = .01 mA × (total load capacitance + crystal capacitance in pF).

4) Maximum operating current for TMS370Cx2xA = 1.90 (CLKIN) + 7 mA.

5) When OSC POWER bit = 0, maximum standby current for TMS370Cx2xA = 0.75 (CLKIN) + 2 mA.

6) When OSC POWER bit = 1, maximum standby current for TMS370Cx2xA = 0.56 (CLKIN) + 1.9 mA. Bit is valid only from 2 MHz to 12 MHz.

16.7.2 TMS370Cx2xA Timings

Refer to Sections 16.1 and 16.2 for timing symbol definitions and parameter measurement points.

Table 16–12. External Clocking Requirements†

No.	Parameter		Min	Nom	Max	Unit
1	$t_{w(CI)}$	XTAL2/CLKIN pulse duration‡	20			ns
2	$t_{r(CI)}$	XTAL2/CLKIN rise time			30	ns
3	$t_{f(CI)}$	XTAL2/CLKIN fall time			30	ns
	CLKIN	Crystal operating frequency	2		20	MHz

† For V_{IL} and V_{IH}, refer to Table 16–10, page 16-9.
‡ This pulse may be either a high pulse, as illustrated, which extends from the earliest valid high to the final valid high in an XTAL2/CLKIN cycle, or a low pulse, which extends from the earliest valid low to the final valid low in an XTAL2/CLKIN cycle.

Figure 16–7. External Clock Timing

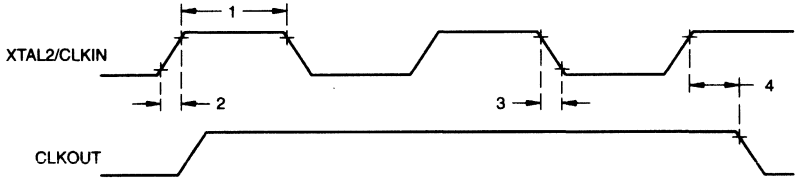
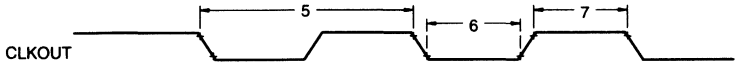


Table 16–13. Switching Characteristics and Timing Requirements†

No.	Parameter		Min	Max	Unit
4	$t_{d(CIH-COL)}$	Delay time, XTAL2/CLKIN rise to CLKOUT fall		100	ns
5	t_c	CLKOUT (system clock) cycle time	200	2000	ns
6	$t_{w(COL)}$	CLKOUT low pulse duration	$0.5t_c$–20	$0.5t_c$	ns
7	$t_{w(COH)}$	CLKOUT high pulse duration	$0.5t_c$	$0.5t_c$+20	ns

† t_c = system clock cycle time = 4/CLKIN.

Figure 16–8. CLKOUT Timing



16.8 TMS370Cx3x Specifications

The tables in this section give specifications that apply to the devices in the TMS370Cx3x category. These devices include the TMS370C032, TMS370C332, and TMS370C732.

16.8.1 TMS370Cx3x Electrical Specifications

Functional operation of the device at maximum or any other conditions beyond those indicated in Table 16–14 is not implied. Exposure to absolute maximum rated conditions for extended periods may affect device reliability.

Table 16–14. Recommended Operating Conditions

Parameter			Min	Nom	Max	Unit
V_{CC1}	Supply voltage (see Note 1)		4.5	5	5.5	V
V_{CC1}	RAM data retention supply voltage (see Note 2)		3		5.5	V
V_{CC3}	Analog supply voltage (see Note 1)		4.5	5	5.5	V
V_{IL}	Low-level input voltage	All pins except MC	V_{SS}		0.8	V
		MC, normal operation	V_{SS}		0.3	
V_{IH}	High-level input voltage	All pins except MC, XTAL2/CLKIN, and $\overline{RESET}$	2		V_{CC}	V
		XTAL2/CLKIN	0.8 V_{CC}		V_{CC}	
		$\overline{RESET}$	0.7 V_{CC}		V_{CC}	
V_{MC}	MC (mode control) voltage (see Note 3)	EEPROM write protect override (WPO)	11.7	12	13	V
		EPROM programming voltage (V_{PP})	12	12.5	13	
		Microcomputer	V_{SS}		0.3	
T_A	Operating free-air temperature	A version	–40		85	°C
		L version	0		70	°C

Notes: 1) Unless otherwise noted, all voltages are with respect to V_{SS}.

2) To guarantee RAM data retention from 3.0 V to 4.5 V, $\overline{RESET}$ must be externally asserted and released only while V_{CC} is within the recommended operating range of 4.5 V to 5.5 V.

3) The WPO mode may be selected anytime a sufficient voltage is present on the MC pin.

Table 16–15. Electrical Characteristics Over Full Ranges of Recommended Operating Conditions

Parameter			Test Conditions	Min	Typ	Max	Unit
V_{OL}	Low-level output voltage		I_{OL} = 1.4 mA			0.4	V
V_{OH}	High-level output voltage		I_{OH} = –50 µA	0.9 V_{CC}			V
			I_{OH} = –2 mA	2.4			
I_I	Input current	MC	0 V < V_I ≤ 0.3 V			10	µA
			0.3 V < V_I ≤ 13 V			650	
			Note 1 12 V ≤ V_I ≤ 13 V			50	mA
		I/O pins	0 V ≤ V_I ≤ V_{CC}			± 10	µA
I_{OL}	Low-level output current		V_{OL} = 0.4 V	1.4			mA
I_{OH}	High-level output current		V_{OH} = 0.9 V_{CC}	–50			µA
			V_{OH} = 2.4 V	–2			mA
I_{CC}	Supply current (operating mode) OSC POWER bit = 0		Notes 2 and 3 CLKIN = 20 MHz		35	45	mA
			Notes 2 and 3 CLKIN = 12 MHz		25	35	
			Notes 2 and 3 CLKIN = 2 MHz		10	14	
I_{CC}	Supply current (standby mode) OSC POWER bit = 0		Notes 2 and 3 CLKIN = 20 MHz		12	17	mA
			Notes 2 and 3 CLKIN = 12 MHz		8	13	
			Notes 2 and 3 CLKIN = 2 MHz		3	4	
I_{CC}	Supply current (standby mode) OSC POWER bit = 1		Notes 2 and 3 CLKIN = 12 MHz		6	8.6	mA
			Notes 2 and 3 CLKIN = 2 MHz		2	3.0	
I_{CC}	Supply current (halt mode)		Note 2 XTAL2/CLKIN < 0.2 V		15	40	µA

Notes: 1) Input current I_{PP} will be a maximum of 50 mA only when you are programming EPROM.

2) Single-chip mode, ports configured as inputs or outputs with no load. All inputs ≤ 0.2 V or ≥ V_{CC} – 0.2 V.

3) XTAL2/CLKIN is driven with an external square wave signal with 50% duty cycle and rise and fall times less than 10 ns. Currents may be higher with a crystal oscillator. At 20 MHz, this extra current = .01 mA × (total load capacitance + crystal capacitance in pF).

16.8.2 TMS370Cx3x Timings

Refer to Sections 16.1 and 16.2 for timing symbol definitions and parameter measurement points.

Table 16–16. External Clocking Requirements†

No.	Parameter		Min	Nom	Max	Unit
1	$t_{w(CI)}$	XTAL2/CLKIN pulse duration‡	20			ns
2	$t_{r(CI)}$	XTAL2/CLKIN rise time			30	ns
3	$t_{f(CI)}$	XTAL2/CLKIN fall time			30	ns
	CLKIN	Crystal operating frequency	2		20	MHz

† For V_{IL} and V_{IH}, refer to Table 16–14, page 16-12.
‡ This pulse may be either a high pulse, as illustrated, which extends from the earliest valid high to the final valid high in an XTAL2/CLKIN cycle, or a low pulse, which extends from the earliest valid low to the final valid low in an XTAL2/CLKIN cycle.

Figure 16–9. External Clock Timing

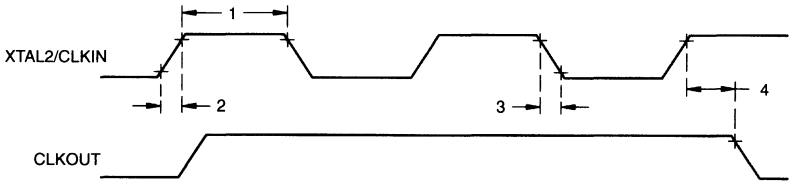
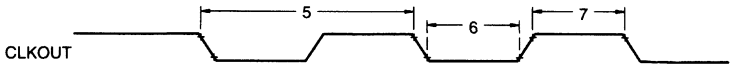


Table 16–17. Switching Characteristics and Timing Requirements†

No.	Parameter		Min	Max	Unit
4	$t_{d(CIH-COL)}$	Delay time, XTAL2/CLKIN rise to CLKOUT fall		100	ns
5	t_c	CLKOUT (system clock) cycle time	200	2000	ns
6	$t_{w(COL)}$	CLKOUT low pulse duration	$0.5t_c$–20	$0.5t_c$	ns
7	$t_{w(COH)}$	CLKOUT high pulse duration	$0.5t_c$	$0.5t_c$+20	ns

† t_c = system clock cycle time = 4/CLKIN.

Figure 16–10. CLKOUT Timing



16.9 TMS370Cx4xA Specifications

The tables in this section give specifications that apply to the devices in the TMS370Cx4xA category. These devices include the TMS370C040A, TMS370C042A, TMS370C340A, TMS370C342A, TMS370C642A, TMS370C742A, and SE370C742A.

16.9.1 TMS370Cx4xA Electrical Specifications

Functional operation of the device at maximum or any other conditions beyond those indicated in Table 16–18 is not implied. Exposure to absolute maximum rated conditions for extended periods may affect device reliability.

Table 16–18. Recommended Operating Conditions

Parameter			Min	Nom	Max	Unit
V_{CC1}	Supply voltage (see Note 1)		4.5	5	5.5	V
V_{CC1}	RAM data retention supply voltage (see Note 2)		3		5.5	V
V_{CC3}	Analog supply voltage (see Note 1)		3		5.5	V
V_{SS3}	Analog supply ground		–0.3	0	0.3	V
V_{IL}	Low-level input voltage	All pins except MC	V_{SS}		0.8	V
		MC, normal operation	V_{SS}		0.3	
V_{IH}	High-level input voltage	All pins except MC, XTAL2/CLKIN, and $\overline{RESET}$	2		V_{CC}	V
		XTAL2/CLKIN	0.8 V_{CC}		V_{CC}	
		$\overline{RESET}$	0.7 V_{CC}		V_{CC}	
V_{MC}	MC (mode control) voltage	EEPROM write protect override (WPO)	11.7	12	13	V
		EPROM programming voltage (V_{PP})	12	12.5	13	
		Microcomputer	V_{SS}		0.3	
T_A	Operating free-air temperature	A version	–40		85	°C
		L version	0		70	°C

Notes: 1) Unless otherwise noted, all voltages are with respect to V_{SS}.

2) $\overline{RESET}$ is externally released while V_{CC} is within the recommended operating range of 4.5 V to 5.5 V and is externally activated when V_{CC} <4.5 or V_{CC} > 5.5 V. RAM data retention is valid in the 2-MHz to 20-MHz frequency range. An active $\overline{RESET}$ initializes (clears) RAM location 0000h and 0001h.

16

Table 16–19. Electrical Characteristics Over Full Ranges of Recommended Operating Conditions

Parameter			Test Conditions	Min	Typ	Max	Unit
V_{OL}	Low-level output voltage		I_{OL} = 1.4 mA			0.4	V
V_{OH}	High-level output voltage		I_{OH} = –50 µA	0.9 V_{CC}			V
			I_{OH} = –2 mA	2.4			
I_I	Input current	MC	0 V < V_I ≤ 0.3 V			10	µA
			0.3 V < V_I ≤ 13 V			650	
			Note 1 12 V ≤ V_I ≤ 13 V			50	mA
		I/O pins	0 V ≤ V_I ≤ V_{CC}			± 10	µA
I_{OL}	Low-level output current		V_{OL} = 0.4 V	1.4			mA
I_{OH}	High-level output current		V_{OH} = 0.9 V_{CC}	–50			µA
			V_{OH} = 2.4 V	–2			mA
I_{CC}	Supply current (operating mode) OSC POWER bit = 0 (see Note 4)		Notes 2 and 3 CLKIN = 20 MHz		30	45	mA
			Notes 2 and 3 CLKIN = 12 MHz		20	30	
			Notes 2 and 3 CLKIN = 2 MHz		7	11	
I_{CC}	Supply current (standby mode) OSC POWER bit = 0 (see Note 5)		Notes 2 and 3 CLKIN = 20 MHz		10	17	mA
			Notes 2 and 3 CLKIN = 12 MHz		8	11	
			Notes 2 and 3 CLKIN = 2 MHz		2	3.5	
I_{CC}	Supply current (standby mode) OSC POWER bit = 1 (see Note 6)		Notes 2 and 3 CLKIN = 12 MHz		6	8.6	mA
			Notes 2 and 3 CLKIN = 2 MHz		2	3.0	
I_{CC}	Supply current (halt mode)		Note 2 XTAL2/CLKIN < 0.2 V		2	30	µA

Notes: 1) Input current I_{PP} will be a maximum of 50 mA only when you are programming EPROM.

2) Single-chip mode, ports configured as inputs or outputs with no load. All inputs ≤ 0.2 V or ≥ V_{CC} – 0.2 V.

3) XTAL2/CLKIN is driven with an external square wave signal with 50% duty cycle and rise and fall times less than 10 ns. Currents may be higher with a crystal oscillator. At 20 MHz, this extra current = .01 mA × (total load capacitance + crystal capacitance in pF).

4) Maximum operating current for TMS370Cx4xA = 1.90 (CLKIN) + 7 mA.

5) When OSC POWER bit = 0, maximum standby current for TMS370Cx4xA = 0.75 (CLKIN) + 2 mA.

6) When OSC POWER bit = 1, maximum standby current for TMS370Cx4xA = 0.56 (CLKIN) + 1.9 mA. Bit is valid only from 2 MHz to 12 MHz.

16.9.2 TMS370Cx4xA Timings

Refer to Sections 16.1 and 16.2 for timing symbol definitions and parameter measurement points.

Table 16–20. External Clocking Requirements†

No.	Parameter		Min	Nom	Max	Unit
1	$t_{w(CI)}$	XTAL2/CLKIN pulse duration‡	20			ns
2	$t_{r(CI)}$	XTAL2/CLKIN rise time			30	ns
3	$t_{f(CI)}$	XTAL2/CLKIN fall time			30	ns
	CLKIN	Crystal operating frequency	2		20	MHz

† For V_{IL} and V_{IH}, refer to Table 16–18, page 16-15.
‡ This pulse may be either a high pulse, as illustrated, which extends from the earliest valid high to the final valid high in an XTAL2/CLKIN cycle, or a low pulse, which extends from the earliest valid low to the final valid low in an XTAL2/CLKIN cycle.

Figure 16–11. External Clock Timing

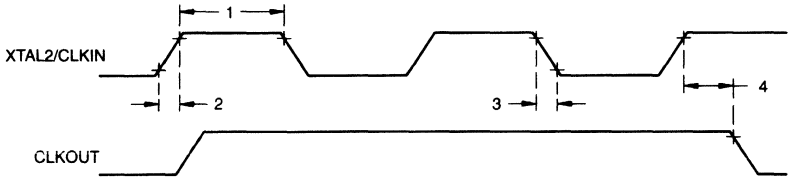
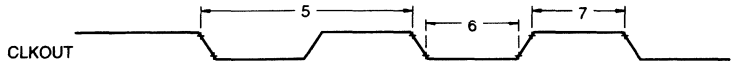


Table 16–21. Switching Characteristics and Timing Requirements†

No.	Parameter		Min	Max	Unit
4	$t_{d(CIH-COL)}$	Delay time, XTAL2/CLKIN rise to CLKOUT fall		100	ns
5	t_c	CLKOUT (system clock) cycle time	200	2000	ns
6	$t_{w(COL)}$	CLKOUT low pulse duration	$0.5t_c$–20	$0.5t_c$	ns
7	$t_{w(COH)}$	CLKOUT high pulse duration	$0.5t_c$	$0.5t_c$+20	ns

† t_c = system clock cycle time = 4/CLKIN.

Figure 16–12. CLKOUT Timing



16.10 TMS370Cx5xA Specifications

16

The tables in this section give specifications that apply to the devices in the TMS370Cx5xA category. These devices include the TMS370C050A, TMS370C150A, TMS370C250A, TMS370C350A, TMS370C052A, TMS370C352A, TMS370C056A, TMS370C156A, TMS370C256A, TMS370C356A, TMS370C756A, SE370C756A, TMS370C058A, TMS370C758A, and SE370C758A.

Note:

Some electrical specifications and timings differ for TMS370Cx5x devices. Refer to Appendix A.

16.10.1 TMS370Cx5xA Electrical Specifications

Functional operation of the device at maximum or any other conditions beyond those indicated in Table 16–22 is not implied. Exposure to absolute maximum rated conditions for extended periods may affect device reliability.

Table 16–22. Recommended Operating Conditions

Parameter			Min	Nom	Max	Unit
V_{CC1}	Supply voltage (see Note 1)		4.5	5	5.5	V
V_{CC1}	RAM data retention supply voltage (see Note 2)		3		5.5	V
V_{CC2}	Digital I/O supply voltage (see Note 1)		4.5	5	5.5	V
V_{CC3}	Analog supply voltage (see Note 1)		4.5	5	5.5	V
V_{SS2}	Digital I/O supply ground		–0.3	0	0.3	V
V_{SS3}	Analog supply supply ground		–0.3	0	0.3	V
V_{IL}	Low-level input voltage	All pins except MC	V_{SS}		0.8	V
		MC, normal operation	V_{SS}		0.3	
V_{IH}	High-level input voltage	All pins except MC, XTAL2/CLKIN, and $\overline{RESET}$	2		V_{CC}	V
		MC (non-WPO mode)	V_{CC}–0.3		V_{CC}+0.3	
		XTAL2/CLKIN	0.8 V_{CC}		V_{CC}	
		$\overline{RESET}$	0.7 V_{CC}		V_{CC}	

Notes: 1) Unless otherwise noted, all voltages are with respect to V_{SS1} (V_{SS1} = V_{SS}).

2) $\overline{RESET}$ is externally released while V_{CC} is within the recommended operating range of 4.5 V to 5.5 V and is externally activated when V_{CC} <4.5 or V_{CC} > 5.5 V. RAM data retention is valid in the 2-MHz to 20-MHz frequency range. An active $\overline{RESET}$ initializes (clears) RAM location 0000h and 0001h.

Table 16–22. Recommended Operating Conditions (Continued)

Parameter			Min	Nom	Max	Unit
V_{MC}	MC (mode control) voltage (see Note 3)	EEPROM write protect override (WPO)	11.7	12	13	V
		EPROM programming voltage (V_{PP})	12	12.5	13	
		Microprocessor	V_{CC}–0.3		V_{CC}+0.3	
		Microcomputer	V_{SS}		0.3	
T_A	Operating free-air temperature	A version	–40		85	°C
		L version	0		70	°C

Notes: 3) The basic microcomputer and microprocessor operating modes are selected by the voltage level applied to the dedicated MC pin 2 system clock cycles (t_c) before the $\overline{\text{RESET}}$ pin goes inactive (high). The WPO mode can be selected anytime a sufficient voltage is present on the MC pin.

You cannot use the internal connections between pins (for example, the connection between V_{SS1} and V_{SS2}) for a jumper from one side of the chip to the other.

16

Table 16–23. Electrical Characteristics Over Full Ranges of Recommended Operating Conditions

Parameter			Test Conditions	Min	Typ	Max	Unit
V_{OL}	Low-level output voltage (see Note 1)		I_{OL} = 1.4 mA			0.4	V
V_{OH}	High-level output voltage		I_{OH} = -50 µA	0.9 V_{CC}			V
			I_{OH} = –2 mA	2.4			
I_I	Input current	MC	0 V < V_I ≤ 0.3 V			10	µA
			0.3 V < V_I < V_{CC}–0.3			50	
			V_{CC}–0.3 V ≤ V_I ≤ V_{CC}+0.3 V			10	
			V_{CC}+0.3 V< V_I ≤13 V			650	
			Note 2; 12 V ≤ V_I ≤ 13 V			50	mA
		I/O pins	0 V ≤ V_I ≤ V_{CC}			± 10	µA
I_{OL}	Low-level output current (see Note 1)		V_{OL} = 0.4 V	1.4			mA
I_{OH}	High-level output current		V_{OH} = 0.9 V_{CC}	–50			µA
			V_{OH} = 2.4 V	–2			mA
I_{CC}	Supply current (operating mode) OSC POWER bit = 0 (see Note 5)	TMS370Cx50A, TMS370Cx52A	Notes 3 and 4; CLKIN = 20 MHz		30	45	mA
		TMS370Cx56A, TMS370Cx58A			35	56	
		TMS370Cx50A, TMS370Cx52A	Notes 3 and 4; CLKIN = 12 MHz		20	30	
		TMS370Cx56A, TMS370Cx58A			25	36	
		TMS370Cx50A, TMS370Cx52A	Notes 3 and 4; CLKIN = 2 MHz		5	11	mA
		TMS370Cx56A, TMS370Cx58A			13	18	
I_{CC}	Supply current (standby mode) OSC POWER bit = 0 (see Note 6)		Notes 3 and 4; CLKIN = 20 MHz		12	17	mA
			Notes 3 and 4; CLKIN = 12 MHz		8	11	
			Notes 3 and 4; CLKIN = 2 MHz		2.5	3.5	
I_{CC}	Supply current (standby mode) OSC POWER bit = 1 (see Note 7)		Notes 3 and 4; CLKIN = 12 MHz		6	8.6	mA
			Notes 3 and 4; CLKIN = 2 MHz		2	3.0	
I_{CC}	Supply current (halt mode)		Notes 3; XTAL2/CLKIN < 0.2 V		2	30	µA

Notes:
1) In prior versions of the TMS370 family, the I_{OL} current was equal to 2 mA for ports A, B, C, and D and the $\overline{\text{RESET}}$ pin.
2) Input current I_{pp} will be a maximum of 50 mA only when you are programming EPROM.
3) Single-chip mode, ports configured as inputs or outputs with no load. All inputs ≤ 0.2 V or ≥ V_{CC} – 0.2 V.
4) XTAL2/CLKIN is driven with an external square wave signal with 50% duty cycle and rise and fall times less than 10 ns. Currents may be higher with a crystal oscillator. At 20 MHz, this extra current = .01 mA × (total load capacitance + crystal capacitance in pF).
5) Maximum operating current for TMS370Cx50A and TMS370Cx52A = 1.9 (CLKIN) + 7 mA. Maximum operating current for TMS370Cx56A = 2.5 (CLKIN) + 5.8 mA.
6) When OSC POWER bit = 0, maximum standby current for TMS370Cx5xA = 0.75 (CLKIN) + 2 mA.
7) When OSC POWER bit = 1, maximum standby current for TMS370Cx5xA = 0.56 (CLKIN) + 1.9 mA. Bit is valid only from 2 MHz to 12 MHz.

16.10.2 TMS370Cx5xA Timings

Refer to Sections 16.1 and 16.2 for timing symbol definitions and parameter measurement points.

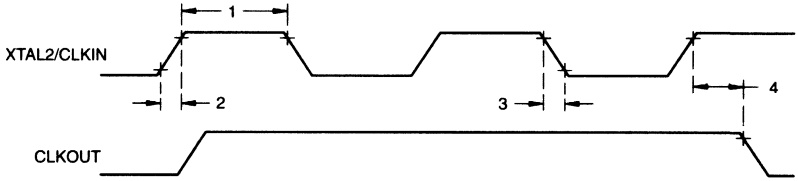
Table 16–24. External Clocking Requirements†

No.	Parameter		Min	Nom	Max	Unit
1	$t_{w(CI)}$	XTAL2/CLKIN pulse duration‡	20			ns
2	$t_{r(CI)}$	XTAL2/CLKIN rise time			30	ns
3	$t_{f(CI)}$	XTAL2/CLKIN fall time			30	ns
4	$t_{d(CIH-COL)}$	Delay time, XTAL2/CLKIN rise to CLKOUT fall			100	ns
	CLKIN	Crystal operating frequency	2		20	MHz

† For V_{IL} and V_{IH}, refer to Table 16–22, page 16-18.

‡ This pulse may be either a high pulse, as illustrated, which extends from the earliest valid high to the final valid high in an XTAL2/CLKIN cycle, or a low pulse, which extends from the earliest valid low to the final valid low in an XTAL2/CLKIN cycle.

Figure 16–13. External Clock Timing



16

Table 16–25. Switching Characteristics and Timing Requirements for External Read and Write†

No.	Parameter		Min	Max	Unit
5	t_c	CLKOUT (system clock) cycle time	200	2000	ns
6	$t_{w(COL)}$	CLKOUT low pulse duration	0.5 t_c – 25	0.5 t_c	ns
7	$t_{w(COH)}$	CLKOUT high pulse duration	0.5 t_c	0.5t_c + 20	ns
8	$t_{d(COL-A)}$	Delay time, CLKOUT low to address R/$\overline{W}$, and $\overline{OCF}$ valid		0.25t_c + 75	ns
9	$t_{v(A)}$	Address valid to $\overline{EDS}$, $\overline{CSE1}$, $\overline{CSE2}$, $\overline{CSH1}$, $\overline{CSH2}$, $\overline{CSH3}$, and $\overline{CSPF}$ low	0.5t_c – 90		ns
10	$t_{su(D)}$	Write data set-up time to $\overline{EDS}$ high	0.75 t_c – 80‡		ns
11	$t_{h(EH-A)}$	Address, R/$\overline{W}$, and $\overline{OCF}$ hold time from $\overline{EDS}$,$\overline{CSE1}$, $\overline{CSE2}$, $\overline{CSH1}$, $\overline{CSH2}$, $\overline{SH3}$, and $\overline{CSPF}$ high	0.5 t_c – 60		ns
12	$t_{h(Eh-D)W}$	Write data hold time from $\overline{EDS}$ high	0.75 t_c + 15		ns
13	$t_{d(DZ-EL)}$	Delay time, data bus high impedance to $\overline{EDS}$ low (read cycle)	0.25 t_c – 35		ns
14	$t_{d(EH-D)}$	Delay time, $\overline{EDS}$ high to data bus enable (read cycle)	1.25 t_c – 40		ns
15	$t_{d(EL-DV)}$	Delay time, $\overline{EDS}$ low to read data valid		t_c – 95‡	ns
16	$t_{h(EH-D)R}$	Read data hold time from $\overline{EDS}$ high	0		ns
17	$t_{su(WT-COH)}$	$\overline{WAIT}$ set-up time to CLKOUT high	0.25 t_c + 70§		ns
18	$t_{h(COH-WT)}$	$\overline{WAIT}$ hold time from CLKOUT high	0		ns
19	$t_{d(ED-WTV)}$	Delay time, $\overline{EDS}$ low to $\overline{WAIT}$ valid		0.5 t_c – 60	ns
20	t_w	Pulse duration, $\overline{EDS}$, $\overline{CSE1}$, $\overline{CSE2}$, $\overline{CSH1}$, $\overline{CSH2}$, $\overline{CSH3}$, and $\overline{CSPF}$ low	t_c – 80‡	t_c + 40‡	ns
21	$t_{d(AV-DV)R}$	Delay time, address valid to read data valid		1.5 t_c – 115‡	ns
22	$t_{d(AV-WTV)}$	Delay time, address valid to $\overline{WAIT}$ valid		t_c – 115	ns
23	$t_{d(AV-EH)}$	Delay time, address valid to $\overline{EDS}$ high (end of write)	1.5t_c – 85‡		ns

† t_c = system clock cycle time = 4/CLKIN.
‡ If wait states, PFWait, or the autowait feature is used, add t_c to this value for each wait state invoked.
§ If the autowait feature is enabled, the $\overline{WAIT}$ input may assume a "don't care" condition until the third cycle of the access. The $\overline{WAIT}$ signal must be synchronized with the high pulse of the CLKOUT signal while still conforming to the minimum set-up time.

Figure 16–14. External Read Timing

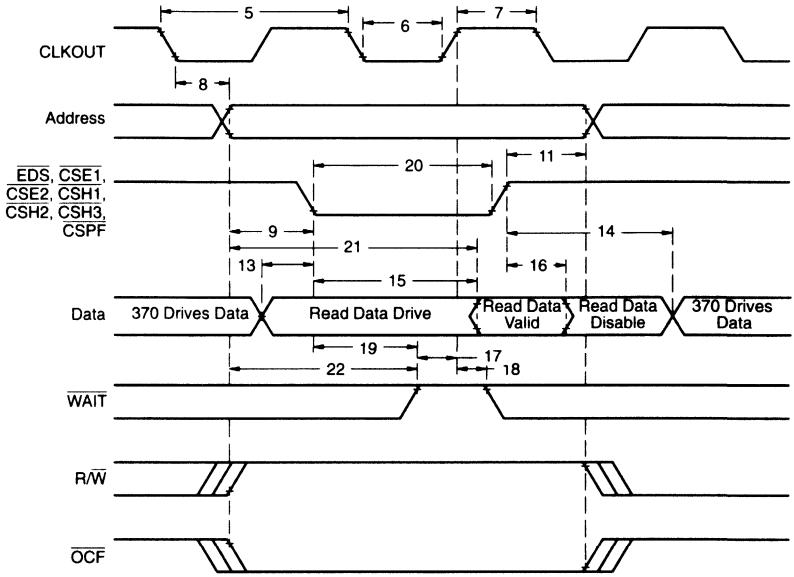
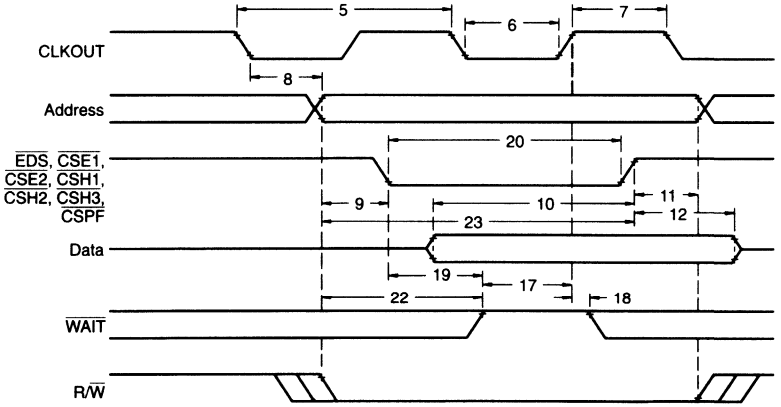


Figure 16–15. External Write Timing



16.11 SCI Timings

16

This section contains timing tables and figures for devices that have the serial communications interface (SCI) module.

Table 16–26. SCI Isosynchronous Mode Timing Characteristics and Requirements for Internal Clock†

No.	Parameter		Min	Max	Unit
24	$t_{c(SCC)}$	SCICLK cycle time	$2\ t_c$	$131{,}072\ t_c$	ns
25	$t_{w(SCCL)}$	SCICLK low pulse duration	$t_c - 45$	$0.5\ t_{c(SCC)} + 45$	ns
26	$t_{w(SCCH)}$	SCICLK high pulse duration	$t_c - 45$	$0.5\ t_{c(SCC)} + 45$	ns
27	$t_{d(SCCL-TXDV)}$	Delay time, SCITXD valid after SCICLK low	−50	60	ns
28	$t_{v(SCCH-TXD)}$	SCITXD data valid after SCICLK high	$t_{w(SCCH)} - 50$		ns
29	$t_{su(RXD-SCCH)}$	SCIRXD set-up time to SCICLK high	$0.25\ t_c + 145$		ns
30	$t_{v(SCCH-RXD)}$	SCIRXD data valid after SCICLK high	0		ns

† t_c = system clock cycle time = 4/CLKIN.

Figure 16–16. SCI Isosynchronous Mode Timing Diagram for Internal Clock

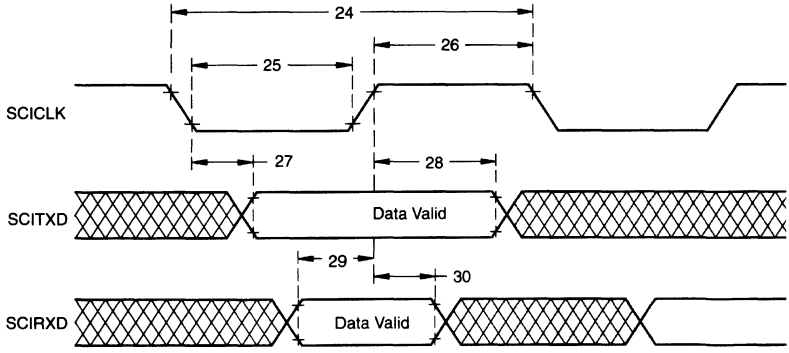
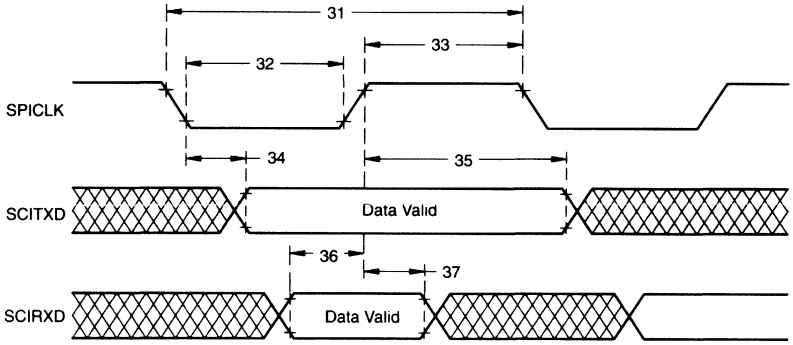


Table 16–27. SCI Isosynchronous Mode Timing Characteristics and Requirements for External Clock†

No.	Parameter		Min	Max	Unit
31	$t_{c(SCC)}$	SCICLK cycle time	10 t_c		ns
32	$t_{w(SCCL)}$	SCICLK low pulse duration	4.25 t_c + 120		ns
33	$t_{w(SCCH)}$	SCICLK high pulse duration	t_c + 120		ns
34	$t_{d(SCCL-TXDV)}$	Delay time, SCITXD valid after SCICLK low		4.25 t_c + 145	ns
35	$t_{v(SCCH-TXD)}$	SCITXD data valid after SCICLK high	$t_{w(SCCH)}$		ns
36	$t_{su(RXD-SCCH)}$	SCIRXD set-up time to SCICLK high	40		ns
37	$t_{v(SCCH-RXD)}$	SCIRXD data valid after SCICLK high	2 t_c		ns

† t_c = system clock cycle time = 4/CLKIN.

Figure 16–17. SCI Isosynchronous Mode Timing Diagram for External Clock



16.12 SPI Timings

16

This section contains timing tables and figures for devices that have the serial peripheral interface (SPI) module.

Note:

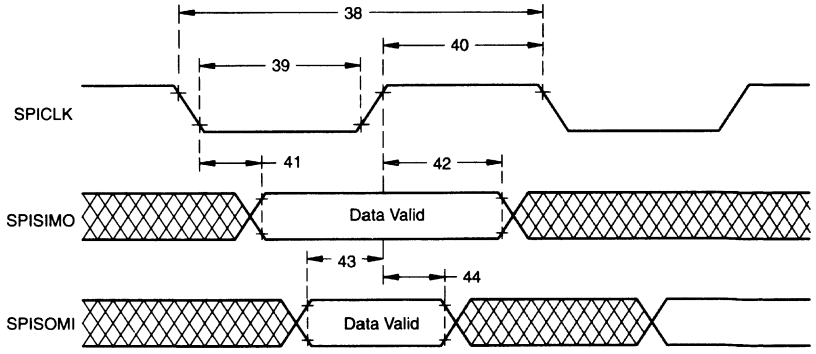
Some SPI electrical specifications and timings differ for TMS370Cxxx devices. Refer to subsection A.6.2, page A-6.

Table 16–28. SPI Master External Timing Characteristics and Requirements†

No.	Parameter		Min	Max	Unit
38	$t_{c(SPC)}$	SPICLK cycle time	$2\ t_c$	$256\ t_c$	ns
39	$t_{w(SPCL)}$	SPICLK low pulse duration	$t_c - 45$	$0.5\ t_{c(SPC)} + 45$	ns
40	$t_{w(SPCH)}$	SPICLK high pulse duration	$t_c - 55$	$0.5\ t_{c(SPC)} + 45$	ns
41	$t_{d(SPCL-SIMOV)}$	Delay time, SPISIMO valid after SPICLK low (polarity = 1)	–65	50	ns
42	$t_{v(SPCH-SIMO)}$	SPISIMO data valid after SPICLK high (polarity = 1)	$t_{w(SPCH)} - 50$		ns
43	$t_{su(SOMI-SPCH)}$	SPISOMI set-up time to SPICLK high (polarity = 1)	$0.25\ t_c + 150$		ns
44	$t_{v(SPCH-SOMI)}$	SPISOMI data valid after SPICLK high (polarity = 1)	0		ns

† t_c = system clock cycle time = 4/CLKIN.

Figure 16–18. SPI Master External Timing



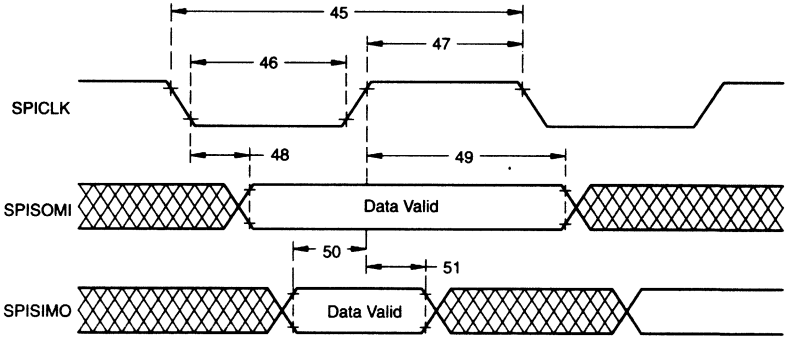
Note: In this figure, polarity = 1. SPICLK is inverted when polarity = 0.

Table 16–29. SPI Slave External Timing Characteristics and Requirements†

No.	Parameter		Min	Max	Unit
45	$t_{c(SPC)S}$	SPICLK cycle time	8 t_c		ns
46	$t_{w(SPCL)S}$	SPICLK low pulse duration	4 t_c – 45	0.5 $t_{c(SPC)S}$ + 45	ns
47	$t_{w(SPCH)S}$	SPICLK high pulse duration	4 t_c – 45	0.5 $t_{c(SPC)S}$ + 45	ns
48	$t_{d(SPCL–SOMIV)S}$	Delay time, SPISOMI valid after SPICLK low (polarity = 1)		3.25 t_c + 130	ns
49	$t_{v(SPCH–SOMI)S}$	SPISOMI data valid after SPICLK high (polarity = 1)	$t_{w(SPCH)S}$		ns
50	$t_{su(SIMO–SPCH)S}$	SPISIMO set-up time to SPICLK high (polarity = 1)	0		ns
51	$t_{v(SPCH–SIMO)S}$	SPISIMO data valid after SPICLK high (polarity = 1)	3 t_c + 100		ns

† t_c = system clock cycle time = 4/CLKIN.

Figure 16–19. SPI Slave External Timing



Notes: 1) In this figure, polarity = 1. SPICLK is inverted when polarity = 0.

2) As a slave, the SPICLK pin is used as the input for the serial clock, which is supplied from the network master.

16.13 A/D Converter Specifications

16

This section contains timing tables and figures for devices that have the analog-to-digital (A/D) converter module.

The A/D converter has a separate power bus for its analog circuitry. These pins are referred to as V_{CC3} and V_{SS3}. The purpose is to enhance A/D performance by preventing digital switching noise of the logic circuitry that may be present on V_{SS} and V_{CC} from coupling into the A/D analog stage. All A/D specifications are given with respect to V_{SS3} unless otherwise noted.

Resolution ..... 8 bits (256 values)

Monotonic ..... Yes

Output conversion code ..... 00h to FFh (00 for $V_I \leq V_{SS3}$; FF for $V_I \geq V_{ref}$)

Conversion time (excluding sample time) ..... 164 t_c

Table 16–30. Recommended Operating Conditions

Parameter		Min	Nom	Max	Unit
V_{CC3}	Analog supply voltage	4.5	5	5.5	V
		$V_{CC} - 0.3$		$V_{CC} + 0.3$	V
V_{SS3}	Analog ground	$V_{SS} - 0.3$		$V_{SS} + 0.3$	V
V_{ref}	Non-V_{CC3} reference†	2.5	V_{CC3}	$V_{CC3} + 0.1$	V
	Analog input for conversion	V_{SS3}		V_{ref}	V

† V_{ref} must be stable, within ± 1/2 LSB of the required resolution, during the entire conversion time.

Table 16–31. A/D Converter Operating Characteristics Over Full Range of Operating Conditions

Parameter		Test Conditions	Min	Nom	Max	Unit
	Absolute accuracy†	V_{CC3} = 5.5 V, V_{ref} = 5.1 V			±1.5	LSB
	Differential/integral linearity error†‡	V_{CC3} = 5.5 V, V_{ref} = 5.1 V			±0.9	LSB
I_{CC3}	Analog supply current	Converting			2	mA
		Nonconverting			5	µA
I_I	Input current, AN0–AN7	$0 V \leq V_I \leq 5.5 V$			2	µA
	V_{ref} input charge current				1	mA
Z_{ref}	Source impedance of V_{ref}	XTAL2/CLKIN ≤ 12 MHz			24	kΩ
		12 MHz < XTAL2/CLKIN ≤ 20 MHz			10	kΩ

† Absolute resolution = 20 mV. At V_{ref} = 5 V, this is 1 LSB. As V_{ref} decreases, LSB size decreases; thus, absolute accuracy and differential/integral linearity errors in terms of LSBs increase.
‡ Excluding quantization error of 1/2 LSB.

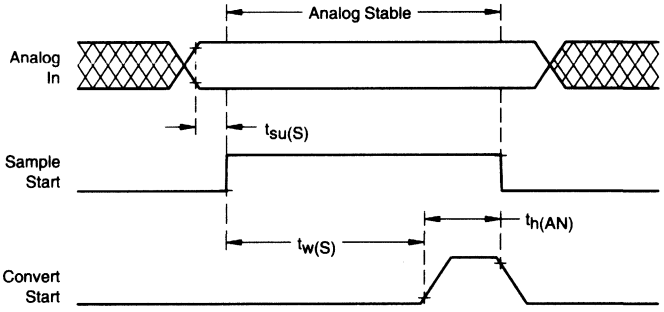
The A/D module allows complete freedom in design of the sources for the analog inputs. The period of the sample time is user-defined so that high-impedance sources can be accommodated without penalty to low-impedance sources. The sample period begins when the SAMPLE START bit of the A/D control register (ADCTL.6) is set to 1. The end of the signal sample period occurs when the conversion bit (CONVERT START, ADCTL.7) is set to 1. After a hold time, the converter will reset the SAMPLE START and CONVERT START bits, signaling that a conversion has started and that the analog signal can be removed.

Table 16–32. Analog Timing Requirements

	Parameter	Min	Nom	Max	Unit
$t_{su(S)}$	Analog input set-up to sample command	0			ns
$t_{h(AN)}$	Analog input hold from start of conversion	18 t_c			ns
$t_{w(S)}$	Duration of sample time per kilohm of source impedance	1			µs/kΩ

† The value given is valid for a signal with a source impedance greater than 1 kΩ. If the source impedance is less than 1 kΩ, use a minimum sampling time of 1 µs.

Figure 16–20. Analog Timing



Chapter 17

Customer Information

This chapter includes general information on mask-ROM prototyping, TMS370 physical characteristics, and parts ordering.

Topic **Page**

17.1 Mask-ROM Prototype and Production Flow

The TMS370 family includes many mask-ROM microcontrollers. The ROM is manufactured containing your application code. The custom-programmed nature of these devices requires a standard, defined interface between you and the factory during production. The prototype and production flow is described in the following steps.

1) **Customer-required information.** For TI to accept the receipt of your ROM algorithm, you must follow these steps:

   a) Complete and submit a new code release form (NCRF—available from a TI Field Sales Office and shown in Example 17–1, page 17-5) describing the custom features of the device (for example, customer information, prototype and production quantities and dates, any exceptions to standard electrical specifications, part numbers and symbols, package type, etc.).

   b) If you requested nonstandard specifications on the NCRF, submit a copy of the description of the microcomputer that includes the functional description and electrical specifications (including absolute maximum ratings, recommended operating conditions, and timing values). TI will then respond to the requested specification changes.

   c) When you have developed and verified your code with the development system, submit the object file in one of the following formats: Intel Hex, TI-Tagged, or COFF (generated by TMS370 development system). Acceptable media include the following:

      - Modem transfer: PC-to-PC via Xmodem, Ymodem, or Zmodem protocol or Microstuf's CROSSTALK XVI protocol
      - DOS-formatted 5-1/4-inch or 3-1/2-inch high-density disk
      - EPROM devices (currently supported: TMS27C128 and TMS27C256)
      - TMS370 EPROM devices (OTP and reprogrammable—refer to Table 15–1 on page 15-25)

   Send the completed NCRF, customer specification (if required), and ROM code to the local representative or to the nearest field sales office.

2) **TI performs ROM receipt.** Code review and ROM receipt is performed on your code, and a unique manufacturing ROM code number (such as R1501234FN) is assigned to your algorithm. All future correspondence should indicate this number. During the ROM receipt procedure, TI:

   ❑ Reads the ROM code information.

   ❑ Processes the ROM code information.

- ❑ Reproduces your ROM object code on the media that you requested on the NCRF. (Note: You must provide the EPROM device if that is the type of media that you have requested on the NCRF.)
- ❑ Returns the processed code and the original code for your verification.

Notes:

- ❑ All TMS370 family devices contain mask-ROM space reserved for TI use only. This space includes addresses 7FE0h through 7FEBh. This reserved area should not be used in your software algorithm, nor should it be used during mask-ROM/firmware development. **The reserved location contents are changed only by TI.**
- ❑ Data EEPROM locations are not programmed in the standard production flow.

3) **Customer ROM receipt approval.** Verify that the ROM code received and processed by TI is correct and that no information was misinterpreted in the transfer. Next, return an algorithm approval form (available from the field sales office) to verify the correct ROM receipt or resubmit the code for processing. This written verification constitutes the contractual agreement for creating the custom mask and manufacturing the ROM verification prototype units.

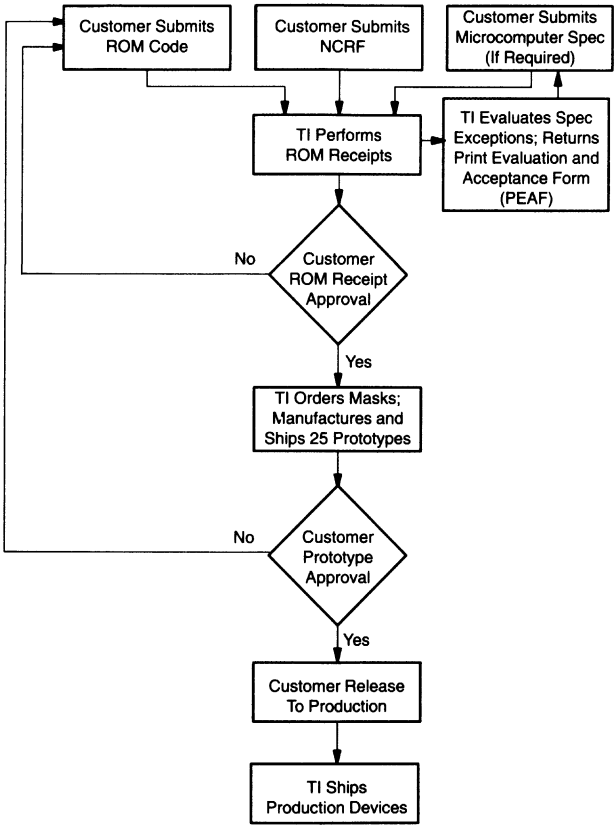
4) **TI orders masks; manufactures and ships 25 prototypes.** TI generates the prototype photomasks and processes, manufactures, and tests 25 microcomputer prototypes containing your ROM pattern. TI then ships the prototypes to you for ROM code verification; these microcomputer devices have been made using the custom mask but are for the purposes of ROM verification only. Prototype devices are identified with a **P** preceding the manufacturing ROM code number (for example, PR1501234FN) to differentiate them from production devices.

5) **Customer prototype approval.** Verify the operation of these prototypes in the system and respond with written approval or disapproval. Written approval constitutes the contractual agreement to initiate volume microcomputer production using the verified prototype ROM code.

6) **Customer release to production.** With your algorithm approval, the ROM code is released to production. TI begins shipping production devices (with production lead time) according to your final specification and order requirements.

Figure 17–1 illustrates the standard of prototype/production.

Figure 17- -1. Prototype and Production Flow



Two lead times are quoted in reference to the preceding flow. For the latest lead times, contact the nearest TI field sales office.

- **Prototype lead time** is the elapsed time from the receipt of written ROM receipt verification to the delivery of 25 prototype devices.
- **Production lead time** is the elapsed time from the receipt of written customer prototype approval to delivery of production devices.

Example 17–1 shows a sample new code release form.

Example 17–1. New Code Release Form

NEW CODE RELEASE FORM
TEXAS INSTRUMENTS
TMS370 MICROCONTROLLER PRODUCTS

DATE: _____

To release a new custom algorithm to TI for incorporation into a TMS370 family microcontroller, complete this form and submit with the following information:

1. A ROM description in object form on Floppy Disk, Modem XFR, or EPROM (Verification file will be returned via same media)
2. An attached specification if not using TI standard specification as incorporated in TI's applicable device data book.

Company Name: _____
Street Address: _____
Street Address: _____
City: _____ State _____ Zip _____

Contact Mr./Ms. _____
Phone (_____) _____ Ext _____

Customer Purchase Order Number: _____

Customer Part Number: _____

Customer Print Number Yes ___ # _____
No ___ (Std. spec to be followed)

TMS370 Device: _____

TI Custom ROM Number: _____

OSCILLATOR FREQUENCY

	min	typ	max
[ ] External Drive (CLKIN)	___	___	___
[ ] Crystal	___	___	___
[ ] Ceramic Resonator	___	___	___

POWER SOURCE
[ ] Supply Voltage min ____ max ____
(std range: 4.5V to 5.5V)

TEMPERATURE RANGE
[ ] 'L': 0 to 70 Deg C (standard)
[ ] 'A': -40 to 85 Deg C
[ ] Other: min _____ max _____

SYMBOLIZATION
[ ] TI standard symbolization
[ ] TI standard w/customer part number
[ ] Custom symbolization
(per attached spec, subject to approval)

. CONTACT OPTIONS FOR THE 'A' VERSION TMS370 MICROCONTROLLERS .

. Low Power Modes
. [ ] Enabled
. [ ] Disabled

Watchdog counter
[ ] Standard
[ ] Hard Enabled
[ ] Simple Counter

.NOTE: Non 'A' versions of the TMS370 microcontrollers will have the "Low power modes Enabled" and "Standard" Watchdog options. See the TMS370 Family Data Manual Appendix A for more details.

PACKAGE TYPE	'X1X	'X2X	'X3X	'X4X	'X5X
[ ] "N" 28-pin PDIP	*				
[ ] "FN" 28-pin PLCC	*				
[ ] "N" 40-pin PDIP		*	*	*	
[ ] "N2" 40-pin SDIP		*	*	*	
[ ] "FN" 44-pin PLCC		*	*	*	
[ ] "FN" 68-pin PLCC					*
[ ] "NM" 64-pin SDIP					*
[ ] Other: _____					

NON-STANDARD SPECIFICATIONS:
ALL NON-STANDARD SPECIFICATIONS MUST BE APPROVED BY THE TI ENGINEERING STAFF: If the customer requires expedited production material (i.e., product which must be started in process prior to prototype approval and full production release) and non-standard spec issues are not resolved to the satisfaction of both the customer and TI in time for a scheduled shipment, the specification parameters in question will be processed/tested to the standard TI spec. Any such devices which are shipped without conformance to a mutually approved spec, will be identified by a 'P' in the symbolization preceding the TI part number.

RELEASE AUTHORIZATION:
This document, including any referenced attachments, is and will be the controlling document for all orders placed for this TI custom device. Any changes must be in writing and mutually agreed to by both the customer and TI. The prototype cycle time commences when this document is signed off and the verification code is approved by the customer.

1. Customer: _____ Date: _____

2. TI: Field Sales: _____
Marketing: _____
Prod Eng: _____
Proto Release: _____

17.2 Mechanical Package Information

The TMS370 microcontroller family devices are assembled in six package types according to the type of material and outline used for the package. These package types are:

- ❑ Plastic dual-inline package (PDIP)
- ❑ Plastic dual-inline shrink package (SDIP)
- ❑ Plastic leaded chip carrier (PLCC)
- ❑ Ceramic dual-inline package (CDIP)
- ❑ Ceramic dual-inline shrink package (CSDIP)
- ❑ Ceramic leaded chip carrier (CLCC)

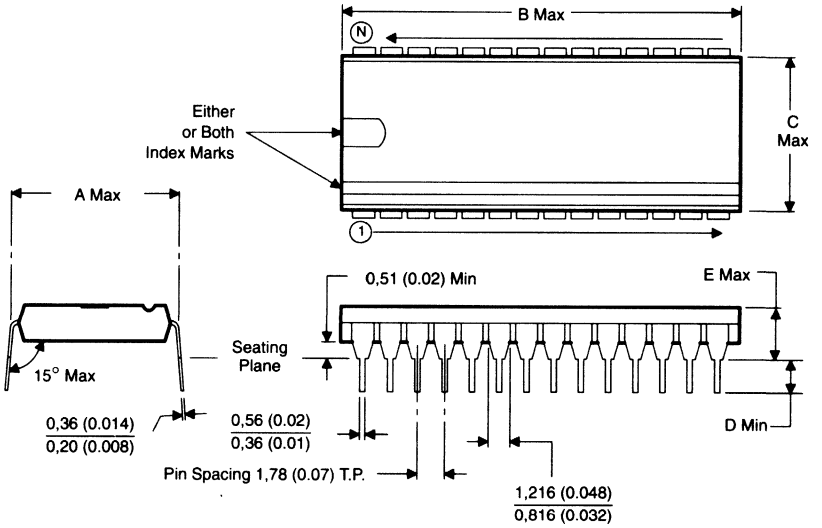
Package types are designated by the suffix on the ROM code number for devices manufactured with your ROM code (for example, R1501234FN) and by the suffix of the standard device number for devices with EPROM. Table 17–1 indicates the package type, suffix indicator, and family members supported on that package type.

Table 17–1. Package Types

Package Type	Suffix Indicator	Family Members
28-pin plastic DIP (100-mil pin spacing)	N	TMS370Cx1x
28-pin ceramic DIP (100-mil pin spacing)	JD	SE370C710
28-pin PLCC (50-mil pin spacing)	FN	TMS370Cx1x
40-pin plastic DIP (100-mil pin spacing)	N	TMS370Cx2x or TMS370Cx4x
40-pin plastic shrink DIP (70-mil pin spacing)	N2	TMS370Cx2x or TMS370Cx4x
40-pin ceramic DIP (100-mil pin spacing)	JD	SE370C722 or SE370C742
40-pin ceramic shrink DIP (70-mil pin spacing)	JC	SE370C722 or SE370C742
44-pin PLCC (50-mil pin spacing)	FN	TMS370Cx2x, TMS370Cx3x, or TMS370Cx4x
44-pin CLCC (50-mil pin spacing)	FZ	SE370C722, SE370C732, or SE370C742
64-pin plastic shrink DIP (70-mil pin spacing)	NM	TMS370Cx5x
64-pin ceramic shrink DIP (70-mil pin spacing)	JN	SE370C756 or SE370C758
68-pin PLCC (50-mil pin spacing)	FN	TMS370Cx5x
68-pin CLCC (50-mil pin spacing)	FZ	SE370C756 or SE370C758

The various package types are shown in the following figures.

Figure 17–2. Plastic Dual-Inline Package (N Suffix)

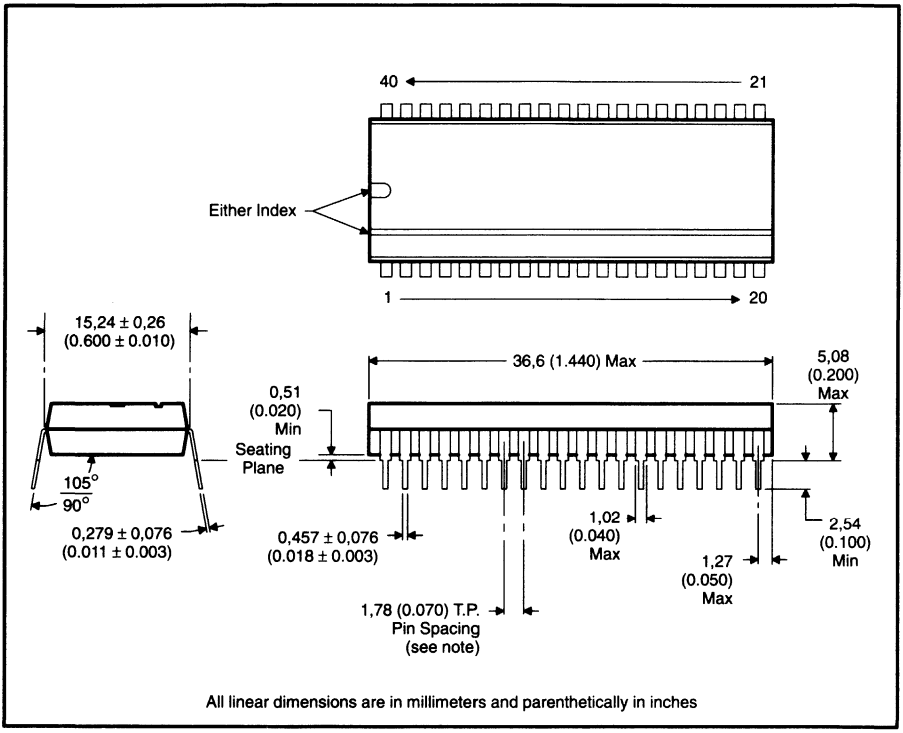


DIM \ PIN	28	40
A (Max)	10,16 (0.400)	15,24 (0.60)
B (Max)	25,79 (1.015)	36,20 (1.425)
C (Max)	9,14 (0.36)	14,22 (0.56)
D (Min)	2,54 (0.10)	3,18 (0.13)
E (Max)	3,30 (0.13)	5,08 (0.20)

All linear dimensions are in millimeters and parenthetically in inches

17

Figure 17–3. 40-Pin Plastic Dual-Inline Shrink Package (N2 Suffix)



Note: Each pin centerline is located within 0,26 (0.010) of its true longitudinal position.

Figure 17–4. 64-Pin Plastic Dual-Inline Shrink Package, 70-mil Pin Spacing (NM Suffix)

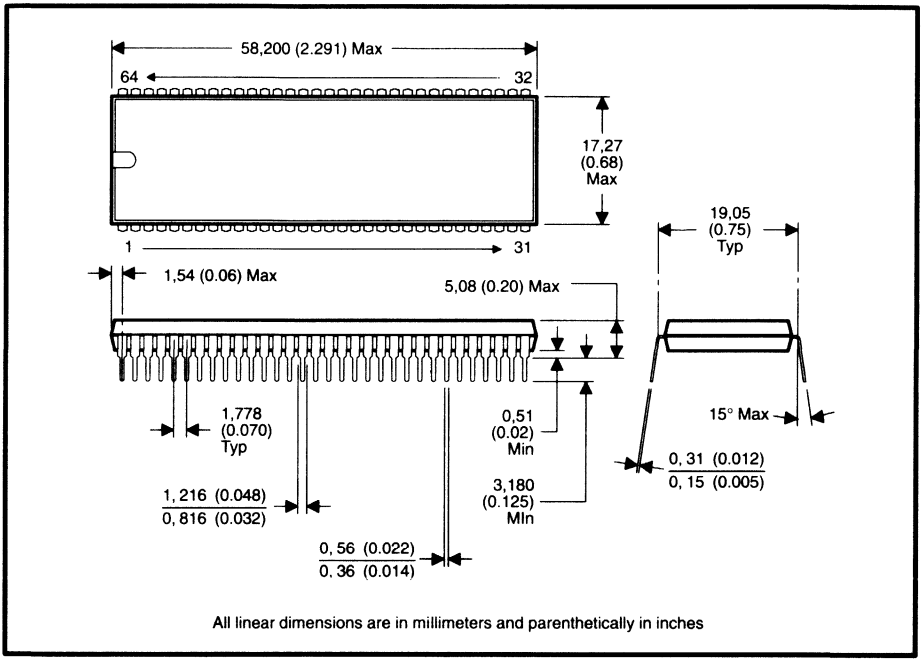
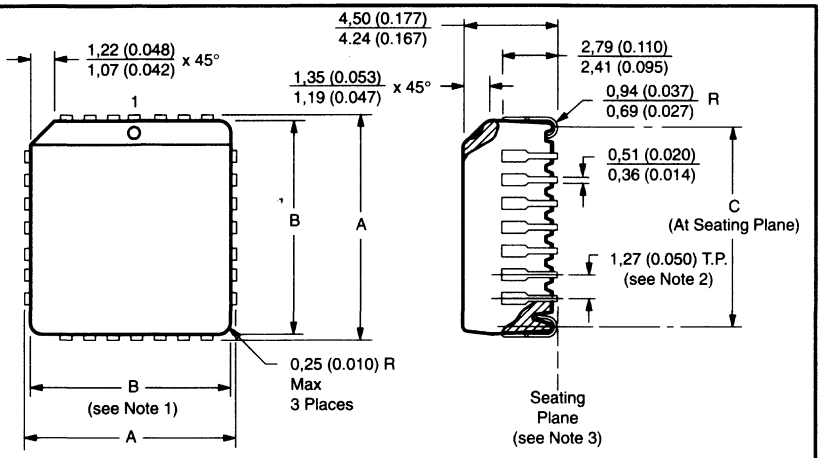


Figure 17–5. Plastic-Leaded Chip Carrier Package (FN Suffix)



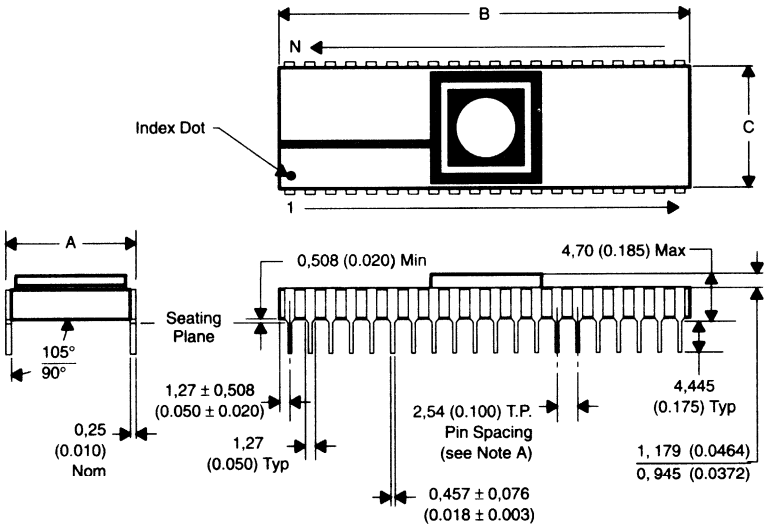
JEDEC Outline	No. of Pins	A		B		C	
		Min	Max	Min	Max	Min	Max
MO-047AB	28	12,32 (0.485)	12,57 (0.495)	11,43 (0.450)	11,58 (0.456)	9,91 (0.390)	10,92 (0.430)
MO-047AC	44	17,40 (0.685)	17,65 (0.695)	16,51 (0.650)	16,66 (0.656)	14,99 (0.590)	16,00 (0.630)
MO-047AE	68	25,02 (0.985)	25,27 (0.995)	24,13 (0.950)	24,33 (0.956)	22,61 (0.840)	23,62 (0.930)

All dimensions and notes to the specified JEDEC outline apply

All linear dimensions are in millimeters and parenthetically in inches

Notes: 1) Centerline of center pin on each side is within 0,10 (0.004) of package centerline as determined by dimension B.
2) Location of each pin is within 0,127 (0.005) of true position with respect to center pin on each side.
3) The lead contact points are planar within 0,10 (0.004).

Figure 17–6. Ceramic Dual-Inline Package (JD Suffix)

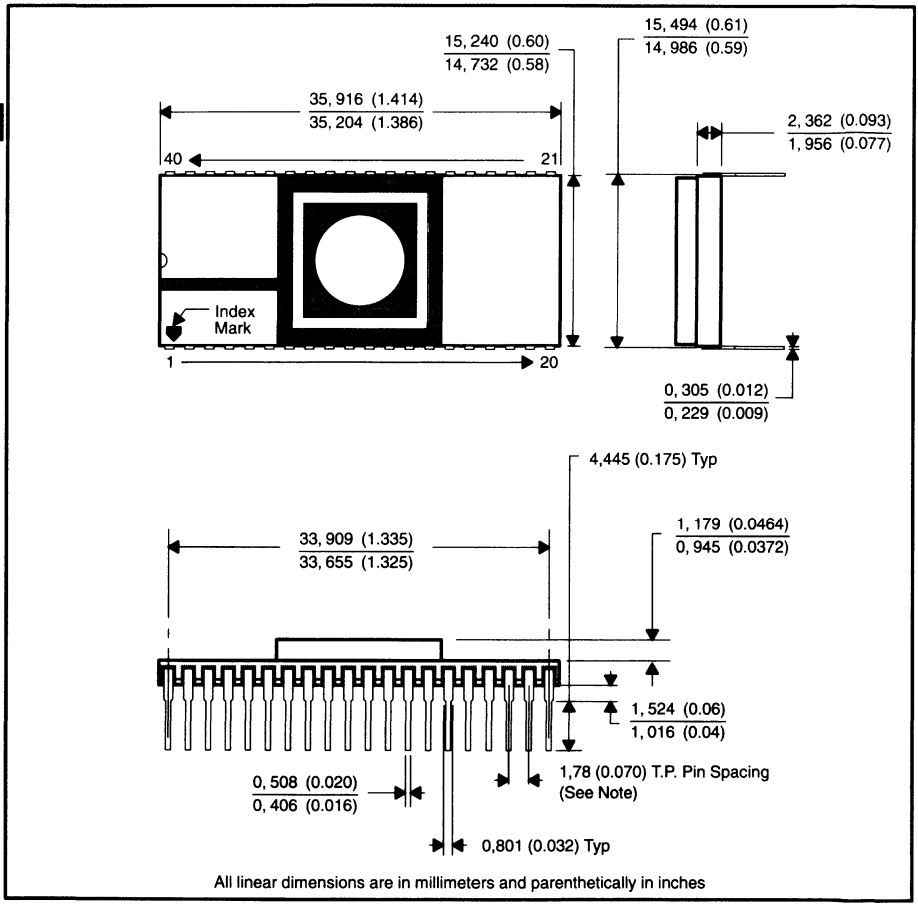


PIN / DIM	28	40
A (Max)	15,44 (0.620)	15,44 (0.620)
B (Max)	35,92 (1.414)	51,31 (2.020)
C (Max)	15,37 (0.605)	15,37 (0.605)

All linear dimensions are in millimeters and parenthetically in inches

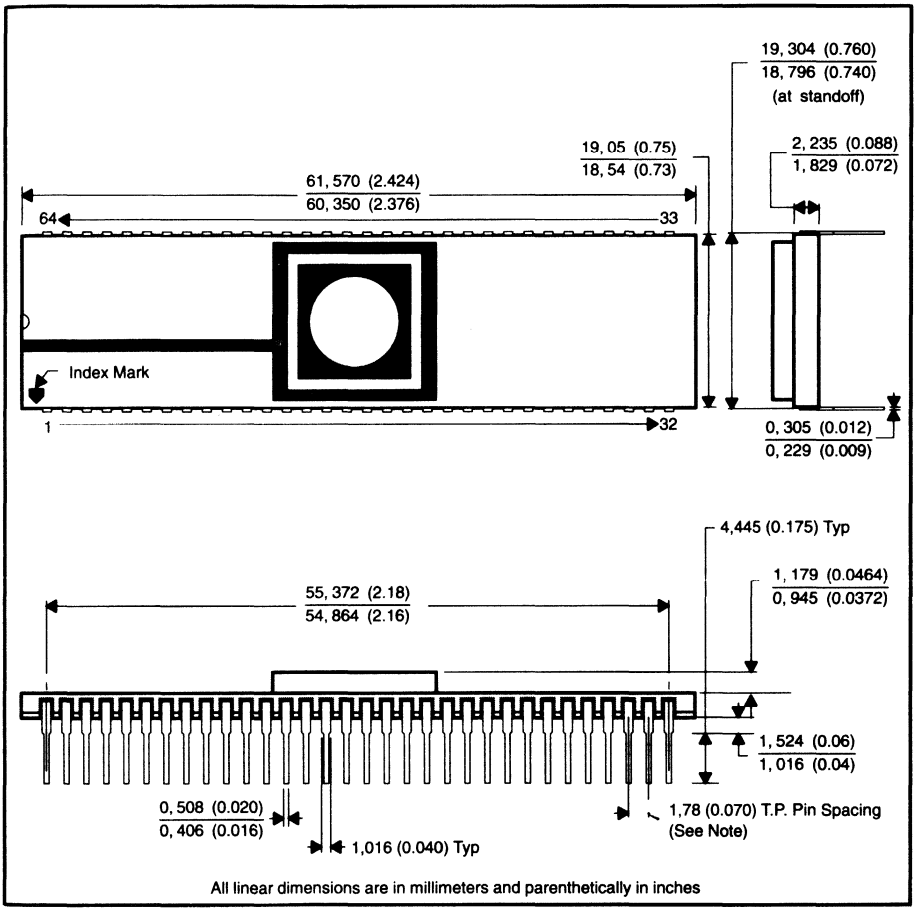
17

Figure 17–7. 40-Pin Ceramic Dual-Inline Shrink Package (JC Suffix)



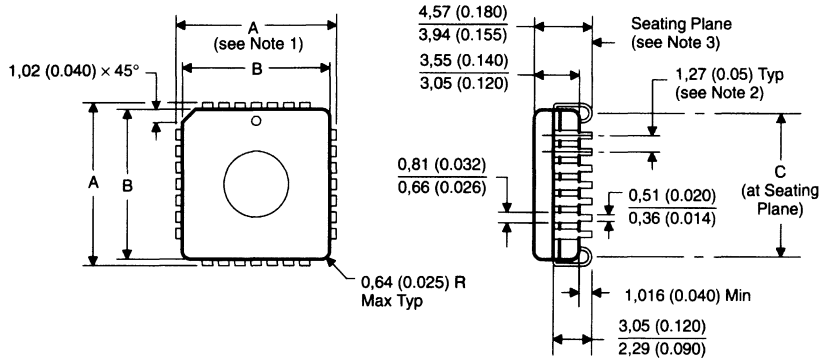
Note: Each pin centerline located within 0,26 (0.010) of its true longitudinal position.

Figure 17–8. 64-Pin Ceramic Dual-Inline Shrink Package, 70-mil Pin Spacing (JN Suffix)



Note: Each pin center line located within 0,26 (0.010) of its true longitudinal position.

Figure 17–9. Ceramic-Leaded Chip Carrier Package (FZ Suffix)



JEDEC Outline	No. of Terminals	A		B		C	
		Min	Max	Min	Max	Min	Max
M0-087AA	28	12,32 (0.485)	12,57 (0.495)	10,92 (0.430)	11,56 (0.455)	10,41 (0.410)	10,92 (0.430)
M0-087AB	44	17,40 (0.685)	17,65 (0.695)	16,00 (0.630)	16,64 (0.655)	15,49 (0.610)	16,00 (0.630)
M0-087AD	68	25,02 (0.985)	25,27 (0.995)	23,62 (0.930)	24,26 (0.955)	23,11 (0.910)	23,62 (0.930)

All linear dimensions are in millimeters and parenthetically in inches

Notes: 1) Centerline of center pin on each side is within 0,10 (0.004) of package centerline as determined by dimension B.
2) Location of each pin is within 0,127 (0.005) of true position with respect to center pin on each side.
3) The lead contact points are planar within 0,15 (0.006).

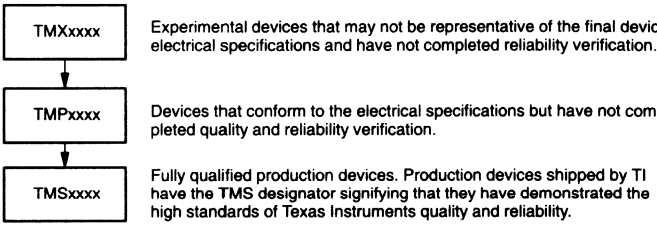
17.3 TMS370 Family Numbering and Symbol Conventions

All TMS370 devices are marked with information as to the type, package, copyright date(s), place of manufacture, and manufacturing data.

17.3.1 Production Device Prefix Designators

To help you identify the development stage of a product, Texas Instruments assigns a prefix designator: TMX, TMP, or TMS. These prefixes represent the evolutionary stages of product development from engineering prototypes through fully qualified production devices. Figure 17–10 depicts this evolutionary development flowchart.

Figure 17–10. Development Flowchart



- TMX devices are shipped against the following disclaimer:
  - Experimental product and its reliability has not been characterized.
  - Product is sold "as is".
  - Product is not warranted to be exemplary of final production version if or when released by Texas Instruments.
- TMP devices are shipped against the following disclaimer:
  - Customer understands that the product purchased here under has not been fully characterized and the expectation of reliability cannot be defined; therefore, Texas Instruments standard warranty refers only to the device's specifications.
  - No warranty of merchantability or fitness is expressed or implied.
- TMS devices have been fully characterized and the quality and reliability of the device has been fully demonstrated. Texas Instruments standard warranty applies.

17.3.2 Support Device Prefix Designators

The SE prefix designation is given to the system evaluator devices that are used for prototyping purposes. This designation applies only to the prototype members of the TMS370 family (SE370C710, SE370C722, SE370C732, SE370C742, SE370C756, and SE370C758). The SE devices are shipped against the following disclaimer:

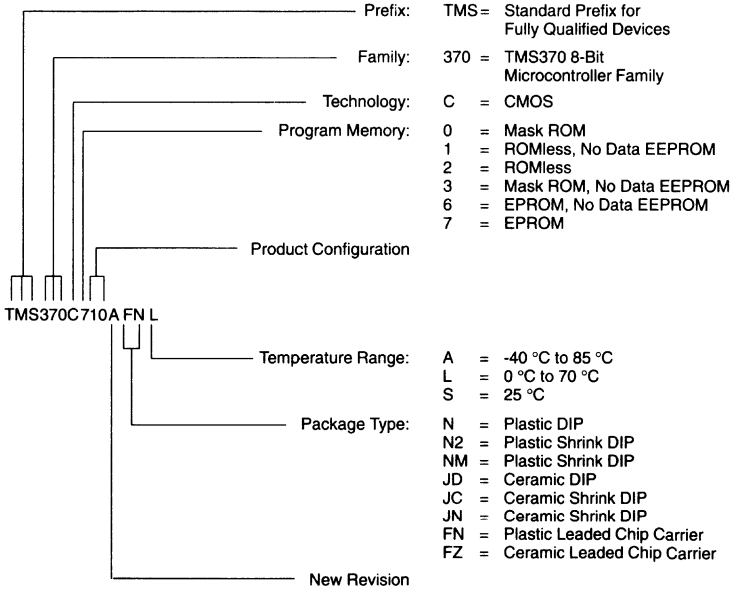
17

Development products are intended for internal evaluation purposes only.

17.3.3 Device Numbering Conventions

Figure 17–11 illustrates the numbering and symbol nomenclature for the TMS370 family.

Figure 17–11. TMS370 Family Nomenclature



17.3.4 Device Symbols

The device symbolization of the TMS370 family members can be divided into two categories: those with factory-programmed mask-ROM, and those with user-programmed memory (OTP and reprogrammable EPROM).

Table 17–2 identifies the designators for the figures in this section.

Table 17–2. Symbolization Designators

Designation	Description	Example
a) Texas Instruments trademark	This is the standard TI signature.	
b) Optional customer part number	You can specify a one-line number that will appear on your mask-ROM device: ❑ On a 28-pin PLCC, you can have 11 characters. ❑ On a 44-pin PLCC, you can have 12 characters. ❑ On a 68-pin PLCC, you can have 15 characters. ❑ On a 28-, 40-, or 64-pin PDIP, you can have 23 characters.	12345678901
c) Customer's ROM code and package type	On mask-ROM devices, you will find a nine-digit number that TI assigns. This number is unique to the device that you ordered.	R1X00XXXN
d) Standard device part number	This is the TMS- part number on OTP devices.	TMS370CXXXN
e) Evaluation device part number	This is the SE- evaluation part number on reprogrammable EPROM devices.	SE370C7XXFZX
f) Tracking mark and date code	This six-character code denotes the date and place of the device assembly. **W** stands for the wafer fabrication plant, **A** represents the device revision, and **YY** and **WW** stand for the year and week the device was assembled.	WAYYWW
g) TI microcode copyright	If your device has only one copyright date, it is that of the TI microcode.	©1986TI
h) ROM code copyright	If your device has more than one copyright date, the second copyright date is that of the ROM code.	©1989TI
i) Lot code	This eight-digit number designates the wafer lot.	12345678
j) EIA code	This code is an alternate to the TI signature. It denotes that the device is a TI device. You **must** use either the EIA code or the TI trademark on your custom-programmed devices.	980
k) Assembly site	This the TI site at which the part was assembled. This designation may appear on the backside of the part.	Philippines

- **TMS370 family members with mask-ROM** are custom-programmed devices in which the ROM is mask programmed according to your application code. These devices follow the prototyping and production flow outlined in Section 17.1. Since they are semicustom devices, they receive a unique ROM code identification number.

  Figure 17–12 illustrates the typical label for mask-ROM devices. Refer to Table 17–2 for the key to the lettered designators.

Figure 17–12. Typical Symbolization for Mask-ROM Devices

- **TMS370 family members with program EPROM (OTP)** are standard device types and have a standard identification. The TMS370 family members with program EPROM include the TMS370C6xx and TMS370C7xx.

  Figure 17–13 illustrates the typical label for OTP devices. Refer to Table 17–2 for the key to the lettered designators.

Figure 17–13. Typical Symbolization for Program EPROM Devices (OTP)

- **TMS370 family members with reprogrammable EPROM** are standard device types and have a standard identification. The TMS370 family members with program EPROM include the SE370C7xx.

  Figure 17–14 illustrates the typical label for reprogrammable EPROM devices. Refer to Table 17–2 for the key to the lettered designators.

Figure 17–14. Typical Symbolization for Reprogrammable EPROM Devices

17.4 Ordering Information for Development Support Tools

All of the necessary development support tools (excluding a PC) for the TMS370 family are available from TI separately or as a complete package. The development tools are designed to work with a PC with a minimum of 512K bytes of memory and a 5-1/4-inch floppy disk drive.

17.4.1 TMS370 Macro Assembler, Linker, C Compiler, and Utilities

These software packages include all of the utilities required for developing object code for the TMS370 devices.

Part Number	Description
TMDS3740810-02	Assembler/Linker (DOS version)
TMDS3740815-02	C Compiler with Assembler/Linker (DOS version)
TMDS3740515-09	C Compiler with Assembler/Linker (Sun version)

17.4.2 TMS370 Design Kit

The TMS370 design kit contains the necessary software and hardware for you to evaluate TMS370Cx1x and TMS370Cx5x devices; however, no linker is included in the kit.

Part Number	Description
TMDS3770110	TMS370 Design Kit

17.4.3 TMS370 Microcontroller Programmer

With the TMS370 microcontroller programmer and gang programmer, you can program the TMS370 prototype devices. Each programmer comes with the necessary cables and control software for interfacing with a PC.

Part Number	Description
TMDS3760500	TMS370 Programmer Base Unit (Requires Top)
TMDS3780510	TMS370 Single-Unit PLCC (Top only)
TMDS3780511	TMS370 Single-Unit DIP (Top only)
TMDS3780521	TMS370 68-Pin PLCC Gang Programmer (Top only)
TMDS3780522	TMS370 28-Pin PLCC Gang Programmer (Top only)
TMDS3780523	TMS370 28-Pin DIP Gang Programmer (Top only)
TMDS3780524	TMS370 44-Pin PLCC Gang Programmer (Top only)
TMDS3780525	TMS370 40-Pin DIP Gang Programmer (Top only)
TMDS3780526	TMS370 40-Pin SDIP Gang Programmer (Top only)
TMDS3778064	TMS370 64/28-Pin DIP Programmer Adapter

17.4.4 TMS370 XDS Systems

The XDS system provides software debugging and overall evaluation of a TMS370-based system. The XDS comes complete with necessary cables and the debugging program.

Part Number	Description
TMDS3762210	TMS370 XDS/22 Emulator—110 V_{AC} (target cables sold separately)
TMDS3762281	TMS370 XDS/22 Emulator—220 V_{AC} (target cables sold separately)
TMDS3788828	28-Pin DIP/PLCC Target Cable
TMDS3788844	40/44-Pin SDIP/PLCC Target Cable
TMDS3788868	64/68-Pin SDIP/PLCC Target Cable

17.4.5 TMS370 Compact Development Tool

The CDT370 includes a debugger, an assembler/linker, and the CDT board.

Part Number	Description
EDSCDT370	TMS370 Compact Development Tool (target cables sold separately)
EDSTRG28DIL	28-Pin DIP Target Cable
EDSTRG28PLCC	28-Pin PLCC Target Cable
EDSTRG40DIL	40-Pin DIP Target Cable ('Cx4x devices only)
EDSTRG44PLCC	44-Pin PLCC Target Cable ('Cx4x devices only)
EDSTRG2XDIL	40-Pin DIP Target Cable ('Cx2x devices only)
EDSTRG2XPLCC	44-Pin PLCC Target Cable ('Cx2x devices only)
EDSTRG68PLCC	68-Pin PLCC Target Cable

17.4.6 XDS Upgrade (Available in Europe Only)

XDS/22 systems can be upgraded to a PACT XDS/22 system. This upgrade is handled through factory repair and requires the XDS/22 system to be shipped to the factory. Please contact factory repair for further information.

Part Number	Description
2563975–0001	XDS/22 upgrade to PACT XDS/22

17.4.7 XDS Target Connectors

For additional or replacement XDS target connectors, contact the TI Factory Repair.

Appendix A

Differences Between a TMS370CxxxA Device and a TMS370Cxxx Device

This manual describes the TMS370 family of microcontroller devices. These devices have been revised to include more robust features that enhance performance and enable new application technologies. The specifications and descriptions included in this manual apply to the TMS370CxxxA devices. This appendix describes how the following features differ for the TMS370Cxxx devices:

Topic **Page**

A.1 Watchdog Options

As described in Section 7.7, you can configure the watchdog timer for the TMS370CxxxA devices as one of three mask options to accommodate various applications:

- A **standard watchdog** for ROMless, EPROM, and mask-ROM devices. This option lets you configure the watchdog timer as a nonwatchdog or as a watchdog.
- A **hard watchdog** for mask-ROM devices. In this configuration, you can operate the watchdog timer only as a watchdog, providing additional system integrity.
- A **simple counter** for mask-ROM devices. In this configuration, the watchdog timer can be used as an event counter, a pulse accumulator, or an interval timer (similar to the nonwatchdog mode in the standard watchdog configuration).

These watchdog options are summarized in Table A–1.

Table A–1. Watchdog Option Summary for TMS370CxxxA Devices

	Standard Watchdog		Hard Watchdog	Simple Counter
	Watchdog	**Nonwatchdog**		
WD OVRFL TAP SEL bit and WD INPUT SELECT0–2 bits	Once the WD OVRFL RST ENA is set, the values of these bits can be changed only after any system reset.	These bits can be changed at any time, as long as WD OVRFL RST ENA is not set.	The values of these WD bits can be changed at any time, even if WD OVRFL RST ENA bit is set. However, the WD INPUT SELECT2 bit is not available.	Once the WD OVRFL RST ENA is set, the values of these bits can be changed only after any system reset.
Generates an interrupt when the watchdog counter overflows?	No	Yes	No	Yes
Generates a system reset?	Yes	No	Yes	No
WD OVRFL RST ENA bit	Select to be a watchdog. If bit=1, watchdog counter does initiate a reset upon overflow. This bit is cleared by any system reset.	Select to be a nonwatchdog. If bit = 0, watchdog counter does not initiate reset upon overflow.	This bit is ignored.	If bit=0, WD bits and WD OVRFL TAP SELECT are not locked. If bit=1, WD bits and WD OVRFL TAP SELECT are locked.
INT1 during low-power modes	Controlled by INT1 ENABLE bit (INT1.0) and INT1 NMI bit (SCCR2.1).	Controlled by INT1 ENABLE bit (INT1.0) and INT1 NMI bit (SCCR2.1).	Enabled as an NMI.	Controlled by INT1 ENABLE bit (INT1.0) and INT1 NMI bit (SCCR2.1).
Available Devices	All devices	All devices	Mask-ROM devices	Mask-ROM devices

For the TMS370Cxxx devices (ROMless, mask-ROM, and EPROM), the watchdog timer operates only in the standard watchdog mode, which is functionally the same as the standard watchdog mode described for the TMS370CxxxA devices in subsection 7.7.1. In this mode, you can set the WD OVRFL RST ENA bit (T1CTL2.7) to configure the watchdog timer as a watchdog, or clear the WD OVRFL RST ENA bit to configure the watchdog timer as a nonwatchdog.

A.2 Timer 1 Control Register 2 (T1CTL2) Bits

Two of the T1CTL2 bits operate differently according to which device you're using:

- **WD OVRFL INT FLAG bit (T1CTL2.5)**: For the TMS370CxxxA devices, this bit is set if the last reset is initiated by the watchdog counter. This bit is cleared by writing a zero to it or by any system reset that is not initiated by the watchdog counter; it is not cleared following a watchdog-initiated reset. Watchdog resets can occur, regardless of the status of this flag.

  For the TMS370Cxxx devices, the bit indicates the status of the watchdog overflow interrupt. If the WD OVRFL RST ENA bit = 1, this bit is cleared by writing a zero to it or by a cycle power. If the WD OVRFL RST ENA bit = 0, this bit is cleared by any system reset. Once a watchdog reset has occurred and if the flag is not cleared, a subsequent watchdog reset will not reset the system.

- **WD OVRFL RST ENA bit (T1CTL2.7)**: For the TMS370CxxxA devices, this bit is cleared by any system reset (for example, a power-up reset, an oscillator fault, or a reinitialized incorrect value to the watchdog timer). For the TMS370Cxxx devices, only a power-up reset can clear this bit.

A.3 System Control and Configuration Register 2 (SCCR2) Bits

Two of the SCCR2 bits operate differently according to which device you're using:

- **OSC FLT DISABLE bit (SCCR2.2)**: For the TMS370CxxxA devices, this bit is reserved; writing a zero or a one to this bit has no effect. If you are using a TMS370Cxxx device, you must clear (0) this bit to ensure proper operation.

- **OSC FLT RST ENA bit (SCCR2.5)**: For the TMS370CxxxA devices, this is a reserved bit; writing a zero or a one to this bit has no effect. When a system detects an oscillator fault, the device generates a system reset, regardless of the status of this bit.

  For the TMS370Cxxx devices, this bit determines whether or not a system reset is generated when an oscillator fault is detected.

A.4 V_{CC1} and V_{CC2} Pins

The V_{CC1} and V_{CC2} pins are internally connected on the TMS370CxxxA devices. For the TMS370Cxxx devices, these pins are not internally connected.

Remember, when using a TMS370CxxxA device, you cannot use the internal connections between pins (for example, the connection between V_{SS1} and V_{SS2}) for a jumper from one side of the chip to the other.

A.5 Low-Power and Idle Modes

For the TMS370CxxxA devices (ROMless, mask-ROM, and EPROM), there are two low-power modes (halt and standby) and an idle mode. Bits 6 and 7 of SCCR2 select the halt, standby, or idle modes. For mask-ROM devices, low-power modes can be disabled permanently through a programmable contact at the time when the mask is manufactured. Refer to Section 4.2, page 4-6 for more information about the low-power and idle modes.

The only difference between the low-power and idle modes described for the TMS370CxxxA devices and the modes for the TMS370Cxxx devices is that low-power modes cannot be disabled permanently for TMS370Cxxx devices.

A.6 Electrical Specifications

A.6.1 Differences for TMS370Cx5x Devices

The differences in electrical specifications between TMS370Cx5xA devices and TMS370Cx5x devices are summarized in this section.

Table A–2. Switching Characteristics and Timing Requirements for External Read and Write†

		TMS370Cx5xA Devices			TMS370Cx5x Devices	
	Parameter	**Min**	**Max**	**Unit**	**Min**	**Max**
t_c	CLKOUT (system clock) cycle time	200	2000	ns	200	2000
$t_{w(COL)}$	CLKOUT low pulse duration	0.5 t_c – 25	0.5 t_c	ns	0.5 t_c – 20	0.5 t_c
$t_{w(COH)}$	CLKOUT high pulse duration	0.5 t_c	0.5 t_c + 20	ns	0.5 t_c	0.5 t_c + 20
$t_{d(COL-A)}$	Delay time, CLKOUT low to address R/$\overline{W}$, and $\overline{OCF}$ valid		0.25 t_c + 75	ns		0.25 t_c + 40
$t_{v(A)}$	Address valid to $\overline{EDS}$, $\overline{CSE1}$, $\overline{CSE2}$, $\overline{CSH1}$, $\overline{CSH2}$, $\overline{CSH3}$, and $\overline{CSPF}$ low	0.5 t_c – 90		ns	0.5 t_c – 50	
$t_{su(D)}$	Write data setup time to $\overline{EDS}$ high	0.75 t_c – 80‡		ns	0.75 t_c – 40‡	
$t_{h(EH-A)}$	Address, R/$\overline{W}$, and $\overline{OCF}$ hold time from $\overline{EDS}$, $\overline{CSE1}$, $\overline{CSE2}$, $\overline{CSH1}$, $\overline{CSH2}$, $\overline{SH3}$, and $\overline{CSPF}$ high	0.5 t_c – 60		ns	0.5 t_c – 40	
$t_{h(Eh-D)W}$	Write data hold time from $\overline{EDS}$ high	0.75 t_c + 15		ns	0.75 t_c + 15	
$t_{d(DZ-EL)}$	Delay time, data bus high impedance to $\overline{EDS}$ low (read cycle)	0.25 t_c – 35		ns	0.25 t_c – 30	
$t_{d(EH-D)}$	Delay time, $\overline{EDS}$ high to data bus enable (read cycle)	1.25 t_c – 40		ns	1.25 t_c – 40	
$t_{d(EL-DV)}$	Delay time, $\overline{EDS}$ low to read data valid		t_c – 95‡	ns		t_c – 65‡
$t_{h(EH-D)R}$	Read data hold time from $\overline{EDS}$ high	0		ns	0	
$t_{su(WT-COH)}$	$\overline{WAIT}$ setup time to CLKOUT high	0.25 t_c + 70§		ns	0.25 t_c + 75§	
$t_{h(COH-WT)}$	$\overline{WAIT}$ hold time from CLKOUT high	0		ns	0	
$t_{d(ED-WTV)}$	Delay time, $\overline{EDS}$ low to $\overline{WAIT}$ valid		0.5 t_c – 60	ns		0.5 t_c – 70
t_w	Pulse duration, $\overline{EDS}$, $\overline{CSE1}$, $\overline{CSE2}$, $\overline{CSH1}$, $\overline{CSH2}$, $\overline{CSH3}$, and $\overline{CSPF}$ low	t_c – 80‡	t_c + 40‡	ns	t_c – 40‡	t_c + 40‡
$t_{d(AV-DV)R}$	Delay time, address valid to read data valid		1.5 t_c – 115‡	ns		1.5 t_c – 75‡
$t_{d(AV-WTV)}$	Delay time, address valid to $\overline{WAIT}$ valid		t_c – 115	ns		t_c – 85
$t_{d(AV-EH)}$	Delay time, address valid to $\overline{EDS}$ high (end of write)	1.5 t_c – 85‡		ns	1.5 t_c – 40‡	

† t_c = system clock cycle time = 4/CLKIN.
‡ If wait states, PFWait, or the autowait feature is used, add t_c to this value for each wait state invoked.
§ If the autowait feature is enabled, the $\overline{WAIT}$ input may assume a "don't care" condition until the third cycle of the access. The $\overline{WAIT}$ signal must be synchronized with the high pulse of the CLKOUT signal while still conforming to the minimum setup time.

Table A–3. I_{OL} (Low-Level Output Current)

Parameter		Test Conditions	Min	Typ	Max	Unit
V_{OL}	Low-level output voltage (port A, B, C, D, and $\overline{RESET}$)	I_{OL} = 1.4 mA for TMS370Cx5xA I_{OL} = 2.0 mA for TMS370Cx5x			0.4	V

A.6.2 Differences in SCI and SPI Specifications

A

The differences in SCI and SPI electrical specifications between TMS370CxxxA devices and TMS370Cxxx devices are summarized in this section.

Table A–4. SCI Isosynchronous Mode Timing Characteristics for Internal Clock

Parameter		TMS370CxxxA		Unit	TMS370Cxxx	
		Min	Max		Min	Max
$t_{d(SCCL-TXDV)}$	Delay time, SCITXD valid after SCICLK low	– 50	60	ns	– 50	50

Table A–5. SPI Slave External Timing Characteristics and Requirements

Parameter		TMS370CxxxA		Unit	TMS370Cxxx	
		Min	Max		Min	Max
$t_{w(SPCH)}$	SPICLK high pulse duration	t_c – 55	0.5 $t_{c(SPC)}$ + 45	ns	t_c – 45	0.5 $t_{c(SPC)}$ + 45
$t_{d(SPCL-SIMOV)}$	Delay time, SPISIMO valid after SPICLK low (polarity = 1)	–65	50	ns	–50	50
$t_{d(SPCL-SOMIV)S}$	Delay time, SPISOMI valid after SPICLK low (polarity = 1)		3.25 t_c + 130	ns		3.25 t_c + 125

A.7 Summary of Differences

The differences between the TMS370CxxxA devices and the TMS370Cxxx devices are summarized in Table A–6.

Table A–6. Differences Between TMS370CxxxA devices and TMS370Cxxx Devices

	TMS370Cxxx ROMless, Mask-ROM, EPROM	TMS370CxxxA ROMless, EPROM	TMS370CxxxA Mask-ROM
Standard Watchdog Option (Watchdog and Nonwatchdog Modes)	✓	✓	✓
Hard Watchdog Option			✓
Simple Counter Option			✓
Value of WD OVRFL RST ENA Bit (T1CTL2.7)	Cleared by power-up reset only.	Cleared by any system reset.	Cleared by any system reset.
Value of WD OVRFL INT FLAG Bit (T1CTL2.5)	Once a watchdog reset has occurred and the bit remains set (1), a subsequent watchdog reset will not reset the system.	If a watchdog reset has occurred, it must be cleared by the software in order to determine the source of resets that follow.	If a watchdog reset has occurred, it must be cleared by the software in order to determine the source of resets that follow.
Value of OSC FLT RST ENA Bit (SCCR2.5)	Determines whether or not a system reset is generated when an oscillator fault is detected.	Reserved bit (value has no effect).	Reserved bit (value has no effect).
Value of OSC FLT DISABLE Bit (SCCR2.2)	Must be cleared to ensure proper operation.	Reserved bit (value has no effect).	Reserved bit (value has no effect).
V_{CC1} and V_{CC2} Pins	Not internally connected.	Internally connected.	Internally connected.
Low-Power Mode Enabled by	Software	Software	Programmable contact when mask is manufactured.

Appendix B

Peripheral File Memory Map

This appendix summarizes the peripheral file (PF) and control bit information. Each PF register is presented as a row of boxes containing the control or status bits that belong to the register. The register designation (for example, SCCR0), memory map address, and the PF hex address (for example, P010) are to the left of each register.

The read/write accessibility of each bit is indicated in parentheses below each bit symbol, with the following definitions:

R = Read
W = Write
P = Write in privilege mode only
C = Clear only
S = Set only
–0 = Cleared when the register is reset
–1 = Set when the register is reset
–* = This bit exhibits a special behavior during or after a reset; see the description for this bit in the appropriate section

The shaded boxes in the tables denote privilege mode bits; that is, these bits can be written to only in the privilege mode.

Topic **Page**

B.1 Peripheral File Frame 1: System Configuration Registers

Designation	ADDR	PF	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SCCR0	1010h	P010	COLD START (RW–*)	OSC POWER (RP–0)	PF AUTO WAIT (RP–0)	OSC FLT FLAG (RW–0)	MC PIN WPO (R–0)	MC PIN DATA (R–*)	—	µP/µC MODE (R–*)
SCCR1	1011h	P011	—	—	—	AUTO-WAIT DISABLE (RP–0)	—	MEMORY DISABLE (RP–*)	—	—
SCCR2	1012h	P012	HALT/ STANDBY (RP–0)	PWRDWN/ IDLE (RP–0)	—	BUS STEST (RP–0)	CPU STEST (RP–1)	—	INT1 NMI (RP–0)	PRIVILEGE DISABLE (RS–0)
INT1	1017h	P017	INT1 FLAG (RC–0)	INT1 PIN DATA (R–0)	—	—	—	INT1 POLARITY (RW–0)	INT1 PRIORITY (RW–0)	INT1 ENABLE (RW–0)
INT2	1018h	P018	INT2 FLAG (RC–0)	INT2 PIN DATA (R–0)	—	INT2 DATA DIR (RW–0)	INT2 DATA OUT (RW–0)	INT2 POLARITY (RW–0)	INT2 PRIORITY (RW–0)	INT2 ENABLE (RW–0)
INT3	1019h	P019	INT3 FLAG (RC–0)	INT3 PIN DATA (R–0)	—	INT3 DATA DIR (RW–0)	INT3 DATA OUT (RW–0)	INT3 POLARITY (RW–0)	INT3 PRIORITY (RW–0)	INT3 ENABLE (RW–0)
DEECTL	101Ah	P01A	BUSY (R–*)	—	—	—	—	AP (RW–0)	W1W0 (RW–0)	EXE (RW–0)
EPCTL	101Ch or 101Eh	P01C or P01E	BUSY (R–0)	VPPS (RW–0)	—	—	—	—	W0 (RW–0)	EXE (RW–0)

For more information about these registers and control bits, refer to the following sections:

- SCCR0, SCCR1, and SCCR2: Section 4.3
- INT1, INT2, and INT3: Section 5.2
- DEECTL: subsection 6.2.2
- EPCTL: subsection 6.4.1

B.2 Peripheral File Frame 2: Digital Port Control Registers

Designation	ADDR	PF	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
APORT1	1020h	P020	Reserved							
APORT2	1021h	P021	Port A Control Register 2							
ADATA	1022h	P022	Port A Data							
ADIR	1023h	P023	Port A Direction							
BPORT1	1024h	P024	Reserved							
BPORT2	1025h	P025	Port B Control Register 2							
BDATA	1026h	P026	Port B Data							
BDIR	1027h	P027	Port B Direction							
CPORT1	1028h	P028	Reserved							
CPORT2	1029h	P029	Port C Control Register 2							
CDATA	102Ah	P02A	Port C Data							
CDIR	102Bh	P02B	Port C Direction							
DPORT1	102Ch	P02C	Port D Control Register 1							
DPORT2	102Dh	P02D	Port D Control Register 2							
DDATA	102Eh	P02E	Port D Data							
DDIR	102Fh	P02F	Port D Direction							

For more information about these registers and control bits, refer to Section 4.4.

B.3 Peripheral File Frame 3: SPI Control Registers

Designation	ADDR	PF	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SPICCR	1030h	P030	SPI SW RESET (RW–0)	CLOCK POLARITY (RW–0)	SPI BIT RATE2 (RW–0)	SPI BIT RATE1 (RW–0)	SPI BIT RATE0 (RW–0)	SPI CHAR2 (RW–0)	SPI CHAR1 (RW–0)	SPI CHAR0 (RW–0)
SPICTL	1031h	P031	RECEIVER OVERRUN (R–0)	SPI INT FLAG (R–0)	—	—	—	MASTER/ SLAVE (RW–0)	TALK (RW–0)	SPI INT ENA (RW–0)
	1032h to 1036h	P032 to P036	Reserved							
SPIBUF	1037h	P037	RCVD7 (R–0)	RCVD6 (R–0)	RCVD5 (R–0)	RCVD4 (R–0)	RCVD3 (R–0)	RCVD2 (R–0)	RCVD1 (R–0)	RCVD0 (R–0)
	1038h	P038	Reserved							
SPIDAT	1039h	P039	SDAT7 (RW–0)	SDAT6 (RW–0)	SDAT5 (RW–0)	SDAT4 (RW–0)	SDAT3 (RW–0)	SDAT2 (RW–0)	SDAT1 (RW–0)	SDAT0 (RW–0)
	103Ah to 103Ch	P03A to P03C	Reserved							
SPIPC1	103Dh	P03D	—	—	—	—	SPICLK DATA IN (R–0)	SPICLK DATA OUT (RW–0)	SPICLK FUNCTION (RW–0)	SPICLK DATA DIR (RW–0)
SPIPC2	103Eh	P03E	SPISIMO DATA IN (R–0)	SPISIMO DATA OUT (RW–0)	SPISIMO FUNCTION (RW–0)	SPISIMO DATA DIR (RW–0)	SPISOMI DATA IN (R–0)	SPISOMI DATA OUT (RW–0)	SPISOMI FUNCTION (RW–0)	SPISOMI DATA DIR (RW–0)
SPIPRI	103Fh	P03F	SPI STEST (RP–0)	SPI PRIORITY (RP–0)	SPI ESPEN (RP–0)	—	—	—	—	—

For more information about these registers and control bits, refer to Section 10.9.

B.4 Peripheral File Frame 4: Timer 1 Control Registers

Designation	ADDR	PF	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1CNTR	1040h	P040	Bit 15	T1 Counter MSbyte						Bit 8
T1CNTR	1041h	P041	Bit 7	T1 Counter LSbyte						Bit 0
T1C	1042h	P042	Bit 15	Compare Register MSbyte						Bit 8
T1C	1043h	P043	Bit 7	Compare Register LSbyte						Bit 0
T1CC	1044h	P044	Bit 15	Capture/Compare Register MSbyte						Bit 8
T1CC	1045h	P045	Bit 7	Capture/Compare Register LSbyte						Bit 0
WDCNTR	1046h	P046	Bit 15	Watchdog Counter MSbyte						Bit 8
WDCNTR	1047h	P047	Bit 7	Watchdog Counter LSbyte						Bit 0
WDRST	1048h	P048	Bit 7	Watchdog Reset Key						Bit 0
T1CTL1	1049h	P049	WD OVRFL TAP SEL † (RP–0)	WD INPUT SELECT2† (RP–0)	WD INPUT SELECT1† (RP–0)	WD INPUT SELECT0† (RP–0)	—	T1 INPUT SELECT2 (RW–0)	T1 INPUT SELECT1 (RW–0)	T1 INPUT SELECT0 (RW–0)
T1CTL2	104Ah	P04A	WD OVRFL RST ENA † (RS–0)	WD OVRFL INT ENA (RW–0)	WD OVRFL INT FLAG (RC–*)	T1 OVRFL INT ENA (RW–0)	T1 OVRFL INT FLAG (RC–0)	—	—	T1 SW RESET (S–0)
T1CTL3	104Bh	P04B	Dual Compare Mode							
			T1EDGE INT FLAG (RC–0)	T1C2 INT FLAG (RC–0)	T1C1 INT FLAG (RC–0)	—	—	T1EDGE INT ENA (RW–0)	T1C2 INT ENA (RW–0)	T1C1 INT ENA (RW–0)
			Capture / Compare Mode							
			T1EDGE INT FLAG (RC–0)	—	T1C1 INT FLAG (RC–0)	—	—	T1EDGE INT ENA (RW–0)	—	T1C1 INT ENA (RW–0)
T1CTL4	104Ch	P04C	Dual Compare Mode							
			T1 MODE = 0 (RW–0)	T1C1 OUT ENA (RW–0)	T1C2 OUT ENA (RW–0)	T1C1 RST ENA (RW–0)	T1CR OUT ENA (RW–0)	T1EDGE POLARITY (RW–0)	T1CR RST ENA (RW–0)	T1EDGE DET ENA (RW–0)
			Capture / Compare Mode							
			T1 MODE = 1 (RW–0)	T1C1 OUT ENA (RW–0)	—	T1C1 RST ENA (RW–0)	—	T1EDGE POLARITY (RW–0)	—	T1EDGE DET ENA (RW–0)
T1PC1	104Dh	P04D	—	—	—	—	T1EVT DATA IN (R–0)	T1EVT DATA OUT (RW–0)	T1EVT FUNCTION (RW–0)	T1EVT DATA DIR (RW–0)
T1PC2	104Eh	P04E	T1PWM DATA IN (R–0)	T1PWM DATA OUT (RW–0)	T1PWM FUNCTION (RW–0)	T1PWM DATA DIR (RW–0)	T1IC/CR DATA IN (R–0)	T1IC/CR DATA OUT (RW–0)	T1IC/CR FUNCTION (RW–0)	T1IC/CR DATA DIR (RW–0)
T1PRI	104Fh	P04F	T1 STEST (RP–0)	T1 PRIORITY (RP–0)	—	—	—	—	—	—

† Once the WD OVRFL RST ENA bit is set, these bits cannot be changed until a reset; this applies only to the standard watchdog and to the simple counter. In the hard watchdog, these bits can be modified at any time; the WD INPUT SELECT2 bit is ignored.

For more information about these registers and control bits, refer to Section 7.9.

B.5 Peripheral File Frame 4: PACT Control Registers

Desig-nation	ADDR	PF	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PACTSCR	1040h	P040	DEFTIM OVRFL INT ENA (RW–0)	DEFTIM OVRFL INT FLAG (RC–0)	CMD/DEF AREA ENA (RW–0)	**FAST MODE SELECT (RP–0)**	**PACT PRESCALE SELECT 3 (RP–0)**	**PACT PRESCALE SELECT 2 (RP–0)**	**PACT PRESCALE SELECT 1 (RP–0)**	**PACT PRESCALE SELECT 0 (RP–0)**
CDSTART	1041h	P041	CMD/DEF AREA INT ENA (RW–0)	—	CMD/DEF AREA START BIT 5 (RW–0)	CMD/DEF AREA START BIT 4 (RW–0)	CMD/DEF AREA START BIT 3 (RW–0)	CMD/DEF AREA START BIT 2 (RW–0)	—	—
CDEND	1042h	P042	—	CMD/DEF AREA END BIT 6 (RW–0)	CMD/DEF AREA END BIT 5 (RW–0)	CMD/DEF AREA END BIT 4 (RW–0)	CMD/DEF AREA END BIT 3 (RW–0)	CMD/DEF AREA END BIT 2 (RW–0)	—	—
BUFPTR	1043h	P043	1 (R–1)	1 (R–1)	BUFFER POINTER BIT 5 (R–1)	BUFFER POINTER BIT 4 (R–1)	BUFFER POINTER BIT 3 (R–0)	BUFFER POINTER BIT 2 (R–0)	BUFFER POINTER BIT 1 (R–0)	— (R–0)
	1044h	P044	Reserved							
SCICTLP	1045h	P045	PACT RXRDY (RC–0)	PACT TXRDY (R–1)	PACT PARITY (R–0)	PACT FE (RC–0)	PACT SCI RX INT ENA (RW–0)	PACT SCI TX INT ENA (RW–0)	—	PACT SCI SW RESET (RW–0)
RXBUFP	1046h	P046	PACT RXDT7 (R–0)	PACT RXDT6 (R–0)	PACT RXDT5 (R–0)	PACT RXDT4 (R–0)	PACT RXDT3 (R–0)	PACT RXDT2 (R–0)	PACT RXDT1 (R–0)	PACT RXDT0 (R–0)
TXBUFP	1047h	P047	PACT TXDT7 (RW–0)	PACT TXDT6 (RW–0)	PACT TXDT5 (RW–0)	PACT TXDT4 (RW–0)	PACT TXDT3 (RW–0)	PACT TXDT2 (RW–0)	PACT TXDT1 (RW–0)	PACT TXDT0 (RW–0)
OPSTATE	1048h	P048	PACT OP8 STATE (RW–0)	PACT OP7 STATE (RW–0)	PACT OP6 STATE (RW–0)	PACT OP5 STATE (RW–0)	PACT OP4 STATE (RW–0)	PACT OP3 STATE (RW–0)	PACT OP2 STATE (RW–0)	PACT OP1 STATE (RW–0)
CDFLAGS	1049h	P049	CMD/DEF INT 7 FLAG (RC–0)	CMD/DEF INT 6 FLAG (RC–0)	CMD/DEF INT 5 FLAG (RC–0)	CMD/DEF INT 4 FLAG (RC–0)	CMD/DEF INT 3 FLAG (RC–0)	CMD/DEF INT 2 FLAG (RC–0)	CMD/DEF INT 1 FLAG (RC–0)	CMD/DEF INT 0 FLAG (RC–0)
CPCTL1	104Ah	P04A	CP2 INT ENA (RW–0)	CP2 INT FLAG (RC–0)	CP2 CAPT RISING EDGE (RW–0)	CP2 CAPT FALLING EDGE (RW–0)	CP1 INT ENA (RW–0)	CP1 INT FLAG (RC–0)	CP1 CAPT RISING EDGE (RW–0)	CP1 CAPT FALLING EDGE (RW–0)
CPCTL2	104Bh	P04B	CP4 INT ENA (RW–0)	CP4 INT FLAG (RC–0)	CP4 CAPT RISING EDGE (RW–0)	CP4 CAPT FALLING EDGE (RW–0)	CP3 INT ENA (RW–0)	CP3 INT FLAG (RC–0)	CP3 CAPT RISING EDGE (RW–0)	CP3 CAPT FALLING EDGE (RW–0)
CPCTL3	104Ch	P04C	CP6 INT ENA (RW–0)	CP6 INT FLAG (RC–0)	CP6 CAPT RISING EDGE (RW–0)	CP6 CAPT FALLING EDGE (RW–0)	CP5 INT ENA (RW–0)	CP5 INT FLAG (RC–0)	CP5 CAPT RISING EDGE (RW–0)	CP5 CAPT FALLING EDGE (RW–0)
CPPRE	104Dh	P04D	BUFFER HALF/FULL INT ENA (RW–0)	BUFFER HALF/FULL INT FLAG (RC–0)	INPUT CAPT PRESCALE SELECT 3 (RW–0)	INPUT CAPT PRESCALE SELECT 2 (RW–0)	INPUT CAPT PRESCALE SELECT 1 (RW–0)	CP6 EVENT ONLY (RW–0)	EVENT COUNTER SW RESET (RW–0)	OP SET/CLR SELECT (RW–0)
WDRST	104Eh	P04E	Watchdog Reset Key							
PACTPRI	104Fh	P04F	**PACT STEST (RP–0)**	—	**PACT GROUP 1 PRIORITY (RP–0)**	**PACT GROUP 2 PRIORITY (RP–0)**	**PACT GROUP 3 PRIORITY (RP–0)**	**PACT MODE SELECT (RP–0)**	**PACT WD PRESCALE SELECT 1 (RP–0)**	**PACT WD PRESCALE SELECT 0 (RP–0)**

For more information about these registers and control bits, refer to Section 12.11.

B.6 Peripheral File Frame 5: SCI Control Registers

Designation	ADDR	PF	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SCICCR	1050h	P050	STOP BITS (RW–0)	EVEN/ODD PARITY (RW–0)	PARITY ENABLE (RW–0)	ASYNC/ ISOSYNC (RW–0)	ADDRESS/ IDLE WUP (RW–0)	SCI CHAR2 (RW–0)	SCI CHAR1 (RW–0)	SCI CHAR0 (RW–0)
SCICTL	1051h	P051	—	—	SCI SW RESET (RW–0)	CLOCK (RW–0)	TXWAKE (RS–0)	SLEEP (RW–0)	TXENA (RW–0)	RXENA (RW–0)
BAUD MSB	1052h	P052	BAUDF (MSB) (RW–0)	BAUDE (RW–0)	BAUDD (RW–0)	BAUDC (RW–0)	BAUDB (RW–0)	BAUDA (RW–0)	BAUD9 (RW–0)	BAUD8 (RW–0)
BAUD LSB	1053h	P053	BAUD7 (RW–0)	BAUD6 (RW–0)	BAUD5 (RW–0)	BAUD4 (RW–0)	BAUD3 (RW–0)	BAUD2 (RW–0)	BAUD1 (RW–0)	BAUD0 (LSB) (RW–0)
TXCTL	1054h	P054	TXRDY (R–1)	TX EMPTY (R–1)	—	—	—	—	—	SCI TX INT ENA (RW–0)
RXCTL	1055h	P055	RX ERROR (R–0)	RXRDY (R–0)	BRKDT (R–0)	FE (R–0)	OE (R–0)	PE (R–0)	RXWAKE (R–0)	SCI RX INT ENA (RW–0)
	1056h	P056	Reserved							
RXBUF	1057h	P057	RXDT7 (R–0)	RXDT6 (R–0)	RXDT5 (R–0)	RXDT4 (R–0)	RXDT3 (R–0)	RXDT2 (R–0)	RXDT1 (R–0)	RXDT0 (R–0)
	1058h	P058	Reserved							
TXBUF	1059h	P059	TXDT7 (RW–0)	TXDT6 (RW–0)	TXDT5 (RW–0)	TXDT4 (RW–0)	TXDT3 (RW–0)	TXDT2 (RW–0)	TXDT1 (RW–0)	TXDT0 (RW–0)
	105Ah	P05A	Reserved							
	105Bh	P05B								
	105Ch	P05C								
SCIPC1	105Dh	P05D	—	—	—	—	SCICLK DATA IN (R–0)	SCICLK DATA OUT (RW–0)	SCICLK FUNCTION (RW–0)	SCICLK DATA DIR (RW–0)
SCIPC2	105Eh	P05E	SCITXD DATA IN (R–0)	SCITXD DATA OUT (RW–0)	SCITXD FUNCTION (RW–0)	SCITXD DATA DIR (RW–0)	SCIRXD DATA IN (R–0)	SCIRXD DATA OUT (RW–0)	SCIRXD FUNCTION (RW–0)	SCIRXD DATA DIR (RW–0)
SCIPRI	105Fh	P05F	SCI STEST (RP–0)	SCITX PRIORITY (RP–0)	SCIRX PRIORITY (RP–0)	SCI ESPEN (RP–0)	—	—	—	—

For more information about these registers and control bits, refer to Section 9.8.

B.7 Peripheral File Frame 6: Timer 2 Control Registers

Designation	ADDR	PF	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T2CNTR	1060h	P060	Bit 15	T2 Counter MSbyte						Bit 8
T2CNTR	1061h	P061	Bit 7	T2 Counter LSbyte						Bit 0
T2C	1062h	P062	Bit 15	Compare Register MSbyte						Bit 8
T2C	1063h	P063	Bit 7	Compare Register LSbyte						Bit 0
T2CC	1064h	P064	Bit 15	Capture/Compare Register MSbyte						Bit 8
T2CC	1065h	P065	Bit 7	Capture/Compare Register LSbyte						Bit 0
T2IC	1066h	P066	Bit 15	Capture Register 2 MSbyte						Bit 8
T2IC	1067h	P067	Bit 7	Capture Register 2 LSbyte						Bit 0
T2CTL1	106Ah	P06A	—	—	—	T2 OVRFL INT ENA (RW–0)	T2 OVRFL INT FLAG (RC–0)	T2 INPUT SELECT1 (RW–0)	T2 INPUT SELECT0 (RW–0)	T2 SW RESET (S–0)
T2CTL2	106Bh	P06B	Dual Compare Mode							
			T2EDGE1 INT FLAG (RC–0)	T2C2 INT FLAG (RC–0)	T2C1 INT FLAG (RC–0)	—	—	T2EDGE1 INT ENA (RW–0)	T2C2 INT ENA (RW–0)	T2C1 INT ENA (RW–0)
			Dual Capture Mode							
			T2EDGE1 INT FLAG (RC–0)	T2EDGE2 INT FLAG (RC–0)	T2C1 INT FLAG (RC–0)	—	—	T2EDGE1 INT ENA (RW–0)	T2EDGE2 INT ENA (RW–0)	T2C1 INT ENA (RW–0)
T2CTL3	106Ch	P06C	Dual Compare Mode							
			T2 MODE= 0 (RW–0)	T2C1 OUT ENA (RW–0)	T2C2 OUT ENA (RW–0)	T2C1 RST ENA (RW–0)	T2EDGE1 OUT ENA (RW–0)	T2EDGE1 POLARITY (RW–0)	T2EDGE1 RST ENA (RW–0)	T2EDGE1 DET ENA (RW–0)
			Dual Capture Mode							
			T2 MODE= 1 (RW–0)	—	—	T2C1 RST ENA (RW–0)	T2EDGE2 POLARITY (RW–0)	T2EDGE1 POLARITY (RW–0)	T2EDGE2 DET ENA (RW–0)	T2EDGE1 DET ENA (RW–0)
T2PC1	106Dh	P06D	—	—	—	—	T2EVT DATA IN (RW–0)	T2EVT DATA OUT (RW–0)	T2EVT FUNCTION (RW–0)	T2EVT DATA DIR (RW–0)
T2PC2	106Eh	P06E	T2IC2/PWM DATA IN (R–0)	T2IC2/PWM DATA OUT (RW–0)	T2IC2/PWM FUNCTION (RW–0)	T2IC2/PWM DATA DIR (RW–0)	T2IC1/CR DATA IN (R–0)	T2IC1/CR DATA OUT (RW–0)	T2IC1/CR FUNCTION (RW–0)	T2IC1/CR DATA DIR (RW–0)
T2PRI	106Fh	P06F	T2 STEST (RF–0)	T2 PRIORITY (RF–0)	—	—	—	—	—	—

For more information about these registers and control bits, refer to Section 8.8.

B.8 Peripheral File Frame 7: A/D Converter Control Registers

Designation	ADDR	PF	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCTL	1070h	P070	CONVERT START (RW–0)	SAMPLE START (RW–0)	REF VOLT SELECT2 (RW–0)	REF VOLT SELECT1 (RW–0)	REF VOLT SELECT0 (RW–0)	AD INPUT SELECT2 (RW–0)	AD INPUT SELECT1 (RW–0)	AD INPUT SELECT0 (RW–0)
ADSTAT	1071h	P071	—	—	—	—	—	AD READY (R–0)	AD INT FLAG (RC–0)	AD INT ENA (RW–0)
ADDATA	1072h	P072	A-to-D Conversion Data Register (R–0)							
	1073h to 107Ch	P073 to P07C	Reserved							
ADIN	107Dh	P07D	Port E Data Input Register (R–0)							
ADENA	107Eh	P07E	Port E Input Enable Register (RW–0)							
ADPRI	107Fh	P07F	AD STEST (RP–0)	AD PRIORITY (RP–0)	AD ESPEN (RP–0)	—	—	—	—	—

For more information about these registers and control bits, refer to Section 11.4.

B

Appendix C

Block Diagrams

C

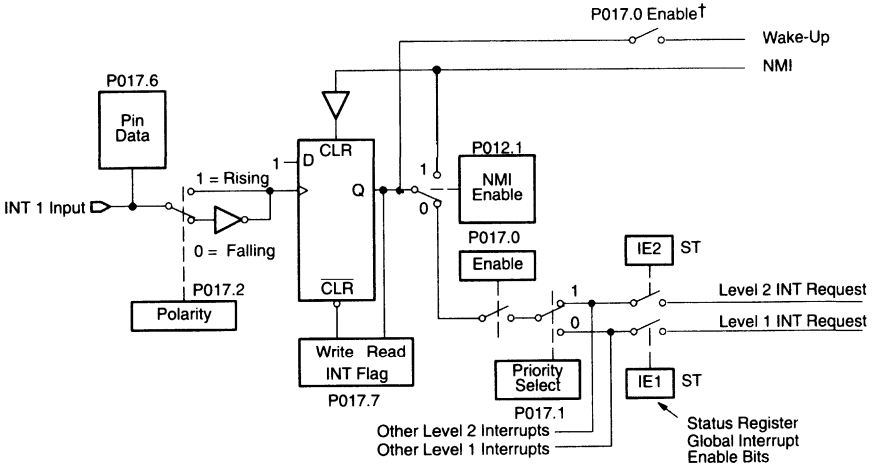
This appendix summarizes the block diagrams of the major circuits.

Topic **Page**

C.1 Interrupts

The block diagram for interrupt 1 is shown in Figure C–1. This interrupt is controlled by the SCCR2 (P012) and INT1 (P017) registers. The control bits for these registers are shown in Section B.1, page B-2.

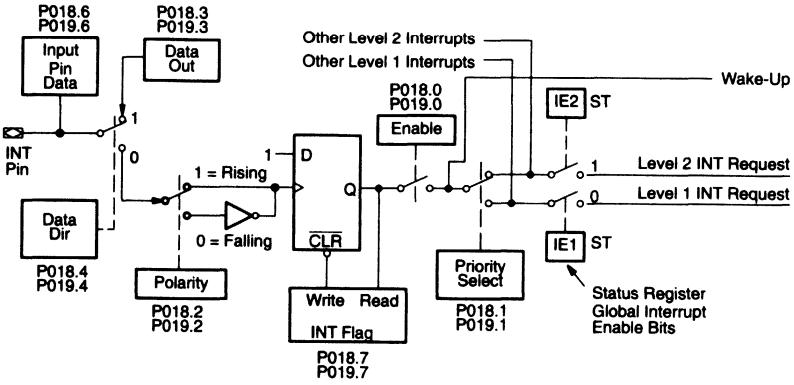
Figure C–1. Interrupt 1 Block Diagram



† This bit is ignored if you are using the hard watchdog option.

The block diagram for interrupts 2 and 3 is shown in Figure C–2. These interrupts are controlled by the INT2 (P018) and INT3 (P019) registers. The control bits for these registers are shown in Section B.1, page B-2.

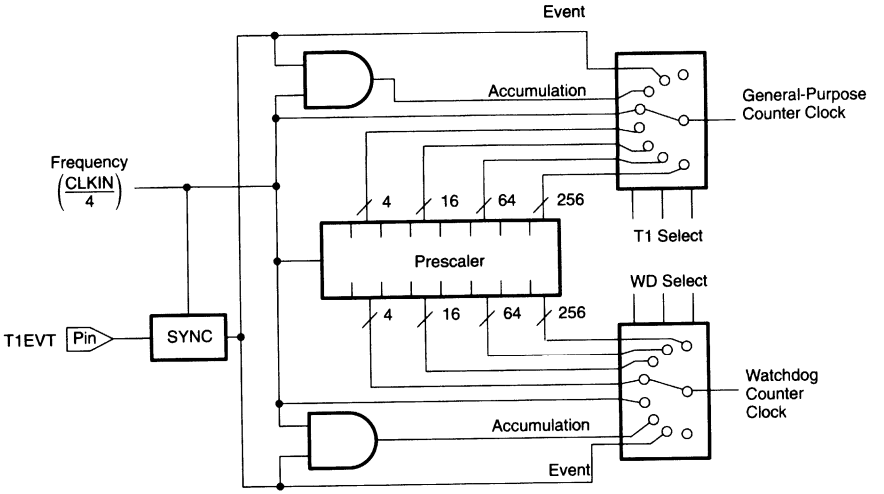
Figure C–2. Interrupts 2 and 3 Block Diagram



C.2 Timer 1 Module

The prescaler for the timer 1 module is shown in Figure C–3. The prescaler is controlled by the T1CTL1 register, located at P049 in peripheral file frame 4. The control bits for this register are shown in Section B.4, page B-5.

Figure C–3. Timer 1 System Clock Prescaler Block Diagram



Note: For the hard watchdog option, the 8-bit prescaler provides four possible clock sources by dividing the system clock by 4, 16, 64, or 256.

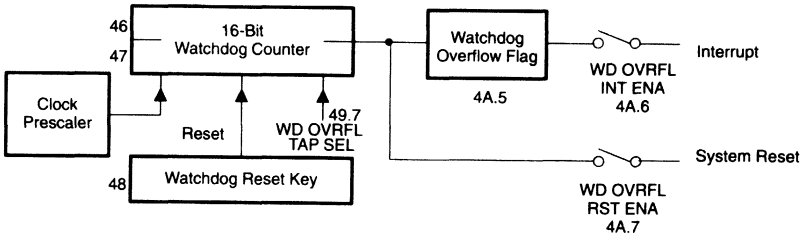
The timer 1 watchdog timer is shown in Figure C–4 *(a)* through *(c)*. The watchdog timer is controlled by the following registers:

- WDCNTR (P046 and P047)
- WDRST (P048)
- T1CTL1 (P049)
- T1CTL2 (P04A)

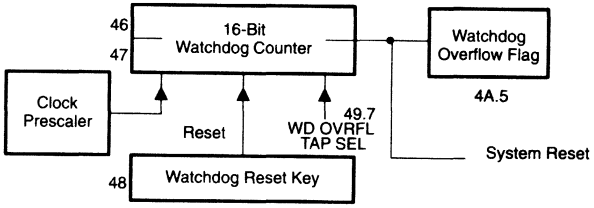
The control bits for these registers are shown in Section B.4, page B-5.

Figure C–4. Watchdog Timer Block Diagrams

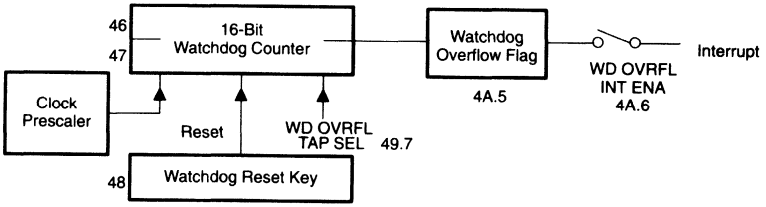
(a) Standard Watchdog



(b) Hard Watchdog



(c) Simple Counter



The timer 1 module has two operating modes:

- **Dual compare mode** is shown in Figure C–5
- **Capture/compare mode** is shown in Figure C–6

For a summary of the timer 1 module control registers and bits, refer to Section B.4, page B-5.

Figure C–5. Timer 1: Dual Compare Mode Block Diagram

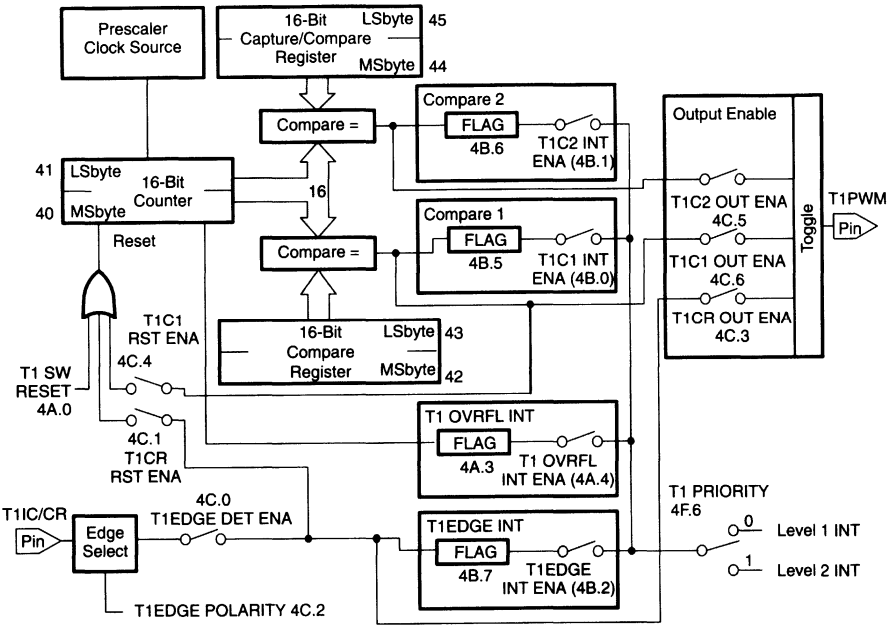
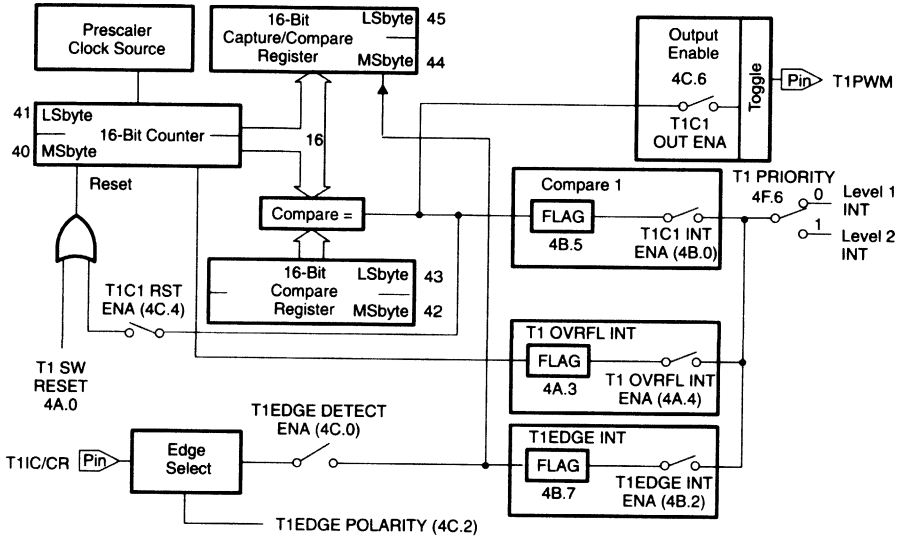


Figure C–6. Timer 1: Capture/Compare Mode Block Diagram



C.3 Timer 2 Module

The timer 2 module has two operating modes:

- **Dual compare mode** is shown in Figure C–7
- **Dual capture mode** is shown in Figure C–8

For a summary of the timer 2 module control registers and bits, refer to Section B.7, page B-8.

C

Figure C–7.Timer 2: Dual Compare Mode Block Diagram

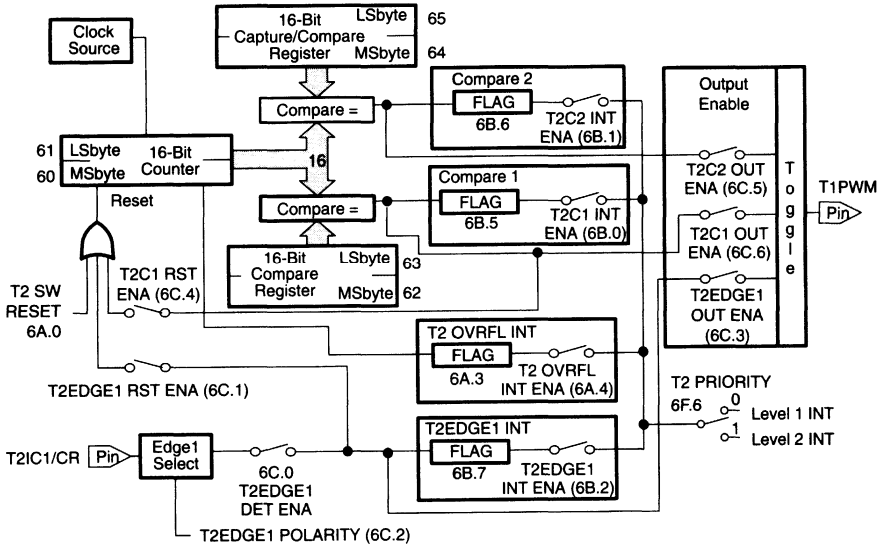
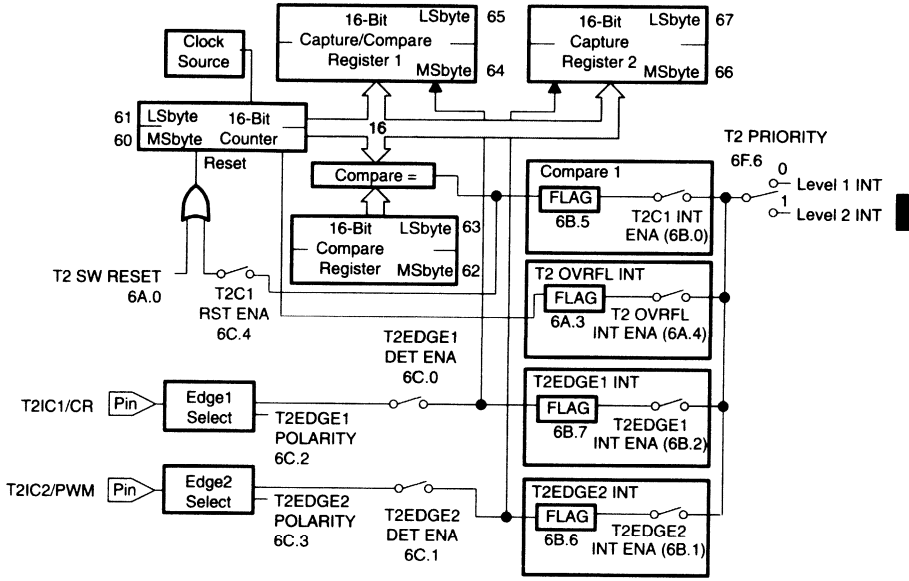


Figure C–8. Timer 2: Dual Capture Mode Block Diagram

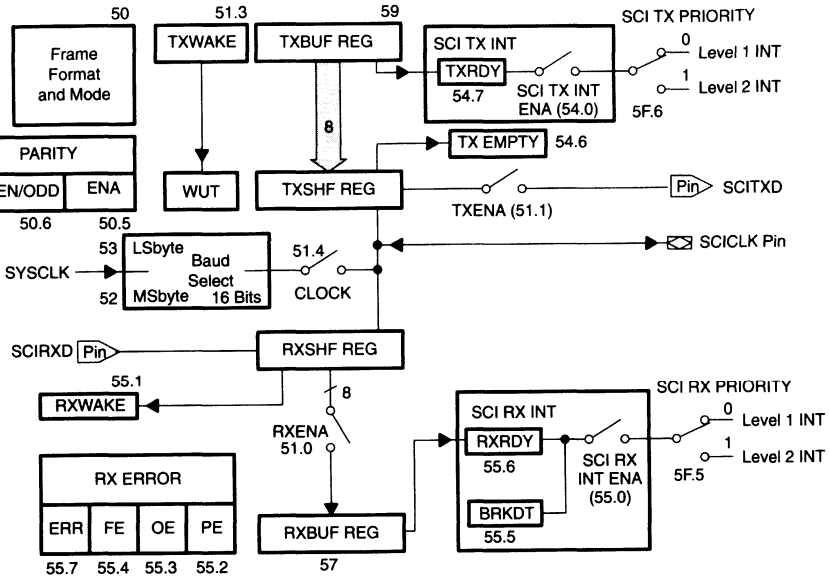


C

C.4 Serial Communications Interface

The block diagram for the SCI module is shown in Figure C–9. For a summary of the SCI control registers and bits, refer to Section B.6, page B-7.

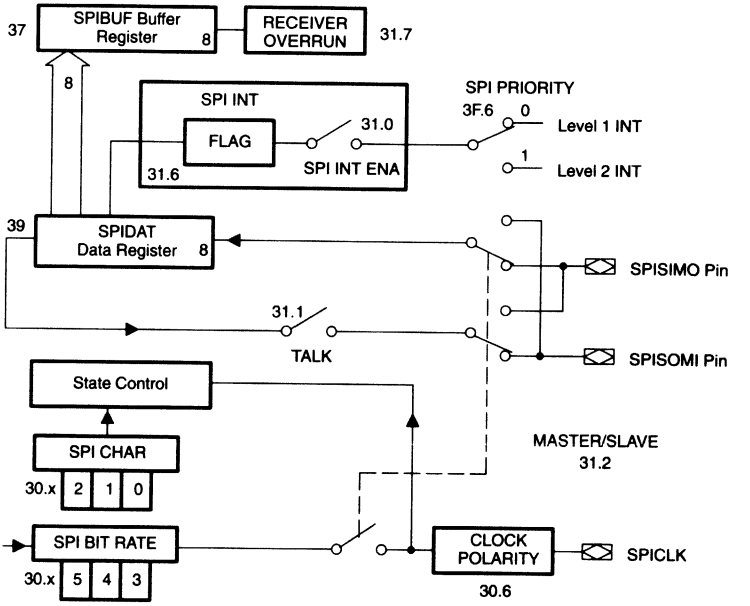
Figure C–9. SCI Block Diagram



C.5 Serial Peripheral Interface

The block diagram for the SPI module is shown in Figure C–10. For a summary of the SPI control registers and bits, refer to Section B.3, page B-4.

Figure C–10. SPI Block Diagram†

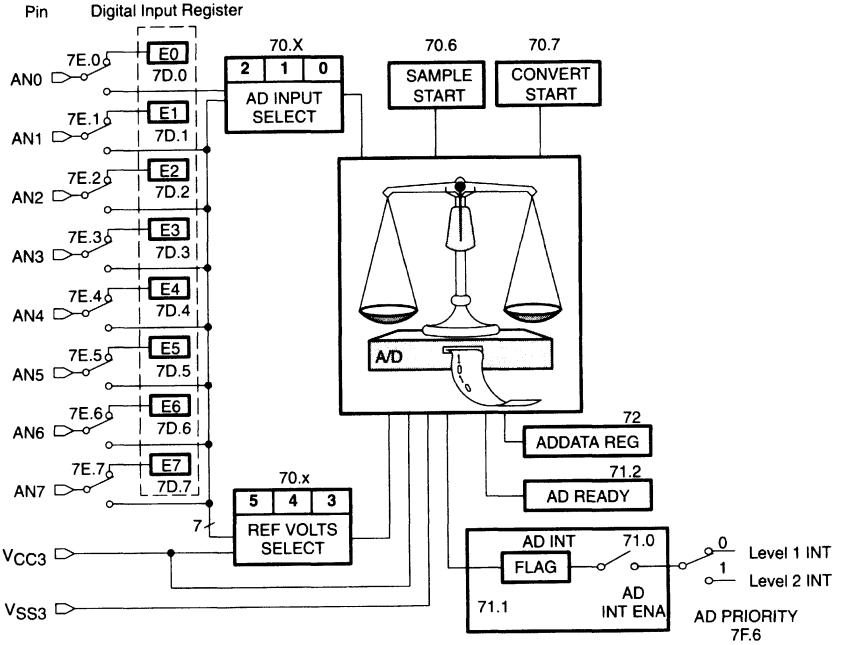


† The diagram is shown in slave mode.

C.6 Analog-to-Digital Converter

The block diagram for the A/D converter module is shown in Figure C–11. For a summary of the A/D control registers and bits, refer to Section B.8, page B-9.

Figure C–11. A/D Converter Block Diagram



Appendix D

ASCII Character Set

The TMS370 assembler recognizes the following ASCII characters.

ASCII Character Set (7-Bit Code)

LSN	0 (0)	1 (16)	2 (32)	3 (48)	4 (64)	5 (80)	6 (96)	7 (112)
0	NUL	DLE	SP	0	@	P	'	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	–	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

Note: To obtain the decimal value, add the decimal MSN to the decimal LSN.

Appendix E

Opcode/Instruction Cross-Reference

Table E–1 provides an opcode-to-instruction cross-reference of all 73 mnemonics and 245 opcodes of the TMS370 instruction set. To check the instruction of a known opcode, locate the left (high) digit across the top of the table, then find the right (low) digit along the side of the table. The intersection contains the instruction mnemonic, operands, and byte/cycle peculiar to that opcode. Some opcodes, such as B0, are shared by two instructions, in which case both mnemonics are shown along with the byte/cycles count.

Table E–1. TMS370 Family Opcode/Instruction Map

LSN \ MSN	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	JMP ra 2/7							INCW #n,Rp 3/11	MOV Ps,A 2/8			CLRC / TST A 1/9	MOV A,B 1/9	MOV A,Rd 2/7	TRAP 15 1/14	LDST n 2/6
1	JN ra 2/5		MOV A,Pd 2/8			MOV B,Pd 2/8		MOV Rs,Pd 3/10		MOV Ps,B 2/7				MOV B,Rd 2/7	TRAP 14 1/14	MOV n(SP),A 2/7
2	JZ ra 2/5	MOV Rs,A 2/7	MOV #n,A 2/6	MOV Rs,B 2/7	MOV Rs,Rd 3/9	MOV #n,B 2/6	MOV B,A 1/8	MOV #n,Rd 3/8			MOV Ps,Rd 3/10	DEC A 1/8	DEC B 1/8	DEC Rn 2/6	TRAP 13 1/14	MOV A,n(SP) 2/7
3	JC ra 2/5	AND Rs,A 2/7	AND #n,A 2/6	AND Rs,B 2/7	AND Rs,Rd 3/9	AND #n,B 2/6	AND B,A 1/8	AND #n,Rd 3/8	AND A,Pd 2/9	AND B,Pd 2/9	AND #n,Pd 3/10	INC A 1/8	INC B 1/8	INC Rn 2/6	TRAP 12 1/14	CMP n(SP),A 2/8
4	JP ra 2/5	OR Rs,A 2/7	OR #n,A 2/6	OR Rs,B 2/7	OR Rs,Rd 3/9	OR #n,B 2/6	OR B,A 1/8	OR #n,Rd 3/8	OR A,Pd 2/9	OR B,Pd 2/9	OR #n,Pd 3/10	INV A 1/8	INV B 1/8	INV Rn 2/6	TRAP 11 1/14	extend inst,2 opcodes
5	JPZ ra 2/5	XOR Rs,A 2/7	XOR #n,A 2/6	XOR Rs,B 2/7	XOR Rs,Rd 3/9	XOR #n,B 2/6	XOR B,A 1/8	XOR #n,Rd 3/8	XOR A,Pd 2/9	XOR B,Pd 2/9	XOR #n,Pd 3/10	CLR A 1/8	CLR B 1/8	CLR Rd 2/6	TRAP 10 1/14	
6	JNZ ra 2/5	BTJO Rs,A,ra 3/9	BTJO #n,A,ra 3/8	BTJO Rs,B,ra 3/9	BTJO Rs,Rd,ra 4/11	BTJO #n,B,ra 3/8	BTJO B,A,ra 2/10	BTJO #n,Rd,ra 4/10	BTJO A,Pd,ra 3/11	BTJO B,Pd,ra 3/10	BTJO #n,Pd,ra 4/11	XCHB A 1/10	XCHB A / TST B 1/10	XCHB Rd 2/8	TRAP 9 1/14	IDLE 1/6
7	JNC ra 2/5	BTJZ Rs,A,ra 3/9	BTJZ #n,A,ra 3/8	BTJZ Rs,B,ra 3/9	BTJZ Rs,Rd,ra 4/11	BTJZ #n,B,ra 3/8	BTJZ B,A,ra 2/10	BTJZ #n,Rd,ra 4/10	BTJZ A,Pd,ra 3/10	BTJZ B,Pd,ra 3/10	BTJZ #n,Pd,ra 4/11	SWAP A 1/11	SWAP B 1/11	SWAP Rn 2/9	TRAP 8 1/14	MOV #n,Pd 3/10
8	JV ra 2/5	ADD Rs,A 2/7	ADD #n,A 2/6	ADD Rs,B 2/7	ADD Rs,Rd 3/9	ADD #n,B 2/6	ADD B,A 1/8	ADD #n,Rd 3/8	MOVW #16,Rpd 4/13	MOVW Rps,Rpd 3/12	MOVW #16(B),Rpd 4/15	PUSH A 1/9	PUSH B 1/9	PUSH Rs 2/7	TRAP 7 1/14	SETC 1/7
9	JL ra 2/5	ADC Rs,A 2/7	ADC #n,A 2/6	ADC Rs,B 2/7	ADC Rs,Rd 3/9	ADC #n,B 2/6	ADC B,A 1/8	ADC #n,Rd 3/8	JMPL lab 3/9	JMPL @Rp 2/8	JMPL lab(B) 3/11	POP A 1/9	POP B 1/9	POP Rd 2/7	TRAP 6 1/14	RTS 1/9
A	JLE ra 2/5	SUB Rs,A 2/7	SUB #n,A 2/6	SUB Rs,B 2/7	SUB Rs,Rd 3/9	SUB #n,B 2/6	SUB B,A 1/8	SUB #n,Rd 3/8	MOV lab,A 3/10	MOV @Rp,A 2/9	MOV lab(B),A 3/12	DJNZ A,ra 2/10	DJNZ B,ra 2/10	DJNZ Rn,ra 3/8	TRAP 5 1/14	RTI 1/12
B	JHS ra 2/5	SBB Rs,A 2/7	SBB #n,A 2/6	SBB Rs,B 2/7	SBB Rs,Rd 3/9	SBB #n,B 2/6	SBB B,A 1/8	SBB #n,Rd 3/8	MOV A,lab 3/10	MOV A,@Rp 2/9	MOV A,lab(B) 3/12	COMPL A 1/8	COMPL B 1/8	COMPL Rn 2/6	TRAP 4 1/14	PUSH ST 1/8

Table E–1. TMS370 Family Opcode/Instruction Map (Concluded)

LSN \ MSN	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C	JNV ra 2/5	MPY Rs,A 2/46	MPY #n,A 2/45	MPY Rs,B 2/46	MPY Rs,Rd 3/48	MPY #n,B 2/45	MPY B,A 1/47	MPY #n,Rd 3/47	BR lab 3/9	BR @Rp 2/8	BR lab(B) 3/11	RR A 1/8	RR B 1/8	RR Rn 2/6	TRAP 3 1/14	POP ST 1/8
D	JGE ra 2/5	CMP Rs,A 2/7	CMP #n,A 2/6	CMP Rs,B 2/7	CMP Rs,Rd 3/9	CMP #n,B 2/6	CMP B,A 1/8	CMP #n,Rd 3/8	CMP lab,A 3/11	CMP @Rp,A 2/10	CMP lab(B),A 3/13	RRC A 1/8	RRC B 1/8	RRC Rn 2/6	TRAP 2 1/14	LDSP 1/7
E	JG ra 2/5	DAC Rs,A 2/9	DAC #n,A 2/8	DAC Rs,B 2/9	DAC Rs,Rd 3/11	DAC #n,B 2/8	DAC B,A 1/10	DAC #n,Rd 3/10	CALL lab 3/13	CALL @Rp 2/12	CALL lab(B) 3/15	RL A 1/8	RL B 1/8	RL Rn 2/6	TRAP 1 1/14	STSP 1/8
F	JLO ra 2/5	DSB Rs,A 2/9	DSB #n,A 2/8	DSB Rs,B 2/9	DSB Rs,Rd 3/11	DSB #n,B 2/8	DSB B,A 1/10	DSB #n,Rd 3/10	CALLR lab 3/15	CALLR @Rp 2/14	CALLR lab(B) 3/17	RLC A 1/8	RLC B 1/8	RLC Rn 2/6	TRAP 0 1/14	NOP 1/7

Second byte of two-byte instructions (F4xx):

F4	8	MOVW n(Rp) 4/15	DIV Rs,A 3/14-63
F4	9	JMPL n(Rp) 4/16	
F4	A	MOV n(Rp),A 4/17	
F4	B	MOV A,n(Rp) 4/16	
F4	C	BR n(Rp) 4/17	
F4	D	CMP n(Rp),A 4/18	
F4	E	CALL n(Rp) 4/20	
F4	F	CALLR n(Rp) 4/22	

= Immediate operand
#16 = Immediate 16-bit number
lab = 16-bit label
n = Immediate 8-bit number
Pd = Peripheral register containing destination byte
Pn = Peripheral register
Ps = Peripheral register containing source byte
ra = Relative address
Rd = Register containing destination byte
Rn = Register file
Rp = Register pair
Rpd = Destination register pair
Rps = Source register pair
Rs = Register containing source byte

Note: All conditional jumps (opcodes 01–0F), BTJO, BTJZ, and DJNZ instructions use two additional cycles if the branch is taken. The BTJO, BTJZ, and DJNZ instructions have a relative address as the last operand.

Appendix F

Instruction/Opcode Cross-Reference and Bus Activity Table

This appendix contains both an instruction-to-opcode cross-reference and an instruction bus activity table. The bus activity table specifies the cycle-by-cycle actions of a given instruction.

Topic **Page**

F.1 Instruction/Opcode Cross-Reference

Table F–1 provides an instruction-to-opcode cross-reference of all 73 mnemonics and 246 opcodes of the TMS370 instruction set. The columns are grouped according to addressing modes.

Table F–1. TMS370 Family Instruction/Opcode Set

	General																	Extended				Other
	A	B	Rn	A, B	B, A	Rn, A	#n, A	Rn, B	#n, B	Rn, Rn	#n, Rn	A, Rn	B, Rn	A, Pn	Pn, A	B, Pn	Pn, B	#n, Pn	†	‡	§	¶	#
ADC					69	19	29	39	59	49	79												
ADD					68	18	28	38	58	48	78												
AND					63	13	23	33	53	43	73			83		93		A3					
BR																			8C	AC	9C	EC	
BTJO					66	16	26	36	56	46	76			86		A6		96					
BTJZ					67	17	27	37	57	47	77			87		A7		97					
CALL																			8E	9E	AE	EE	
CALLR																			8F	9F	AF	EF	
CLR	B5	C5	D5																				
CLRC																							BO
CMP					6D	1D	2D	3D	5D	4D	7D								8D	AD	9D	ED	F3
CMPBIT																							75,A5
COMPL	BB	CB	DB																				
DAC					6E	1E	2E	3E	5E	4E	7E												
DEC	B2	C2	D2																				
DINT																							F0 00
DIV																							F4 F8
DJNZ	BA	CA	DA																				
DSB					6F	1F	2F	3F	5F	4F	7F												
EINT																							F0 0C
EINTH																							F0 04
EINTL																							F0 08
IDLE																							F6
INC	B3	C3	D3																				
INV	B4	C4	D4																				
JBIT0																							77,A7
JBIT1																							76,A6
JMP																							00
JMPL																			89	A9	99	E9	
JC																							03
JEQ/JZ																							02
JG																							0E
JGE																							0D
JHS																							0B

† Direct {(label) → (A)}
‡ Indexed {(label + (B)) → (A)}
§ Indirect {(Rn – 1:Rn) → (A)}
¶ Offset indirect (dual opcode instruction, the first of which is F4) {{b + (Rn – 1:Rn)) → (A)}
# Unless otherwise indicated, includes both single opcode instructions that do not qualify as a general or extended addressing mode and also dual opcode instructions that do not qualify as an offset indirect addressing mode.

Table F–1. TMS370 Family Instruction/Opcode Set (Concluded)

	General																	Extended				Other
	A	B	Rn	A, B	B, A	Rn, A	#n, A	Rn, B	#n, B	Rn, Rn	#n, Rn	A, Rn	B, Rn	A, Pn	Pn, A	B, Pn	Pn, B	#n, Pn	†	‡	§	¶	#
JL																						09
JLE																						0A
JLO																						0F
JN																						01
JNC																						07
JNE/JNZ																						06
JNV																						0C
JP																						04
JPZ																						05
JV																						08
LDSP																						FD
LDST																						F0
MOV				C0	62	12	22	32	52	42	72	D0	D1	21	80	51	91	F7	8B	AA	9A	EA	‖
MOVW																			88	A8	98	E8	
MPY					6C	1C	2C	3C	5C	4C	7C												
NOP																							FF
OR					64	14	24	34	54	44	74			84		94		A4					
POP	B9	C9	D9																				FC
PUSH	B8	C8	D8																				FB
RL	BE	CE	DE																				
RLC	BF	CF	DF																				
RR	BC	CC	DC																				
RRC	BD	CD	DD																				
RTI																							FA
RTS																							F9
SBB					6B	1B	2B	3B	5B	4B	7B												
SBIT0																							*
SBIT1																							□
SETC																							F8
STSP																							FE
SUB					6A	1A	2A	3A	5A	4A	7A												
SWAP	B7	C7	D7																				
TRAP																							◊
TST	B0	C6																					
XCHB	B6	C6	D6																				
XOR					65	15	25	35	55	45	75			85		95		A5					

† Direct {(label) → (A)}
‡ Indexed {(label + (B)) → (A)}
§ Indirect {(Rn – 1:Rn) → (A)}
¶ Offset indirect (dual opcode instruction, the first of which is F4) {(b + (Rn – 1:Rn)) → (A)}
# Unless otherwise indicated, includes both single opcode instructions that do not qualify as a general or extended addressing mode and also dual opcode instructions that do not qualify as an offset indirect addressing mode.
‖ The MOV instruction also includes the following options and their opcodes: Rn,Pn {71}; Pn,Rn {A2}; A,label(B) {AB}; A,n(SP) {F2}; A,n(Rn) {F4 EB}; label,A {8A}; n(SP),A {F1}
* The SBIT0 instruction consists of the following options and their opcodes: Rname {73}; Pname {A3}
□ The SBIT1 instruction consists of the following options and their opcodes: Rname {74}; Pname {A4}
◊ The TRAP instruction consists of 16 options using operands 0 through 15 with opcodes EFh through E0h, respectively

F.2 Bus Activity Table

The TMS370 family employs a microcoded instruction set. Each instruction is broken down into microcode states, and a miniprogram is executed for the instruction using these states. Each microcode state lasts for one internal clock cycle and includes provisions for conditional jumps, branching, and control of internal CPU operations. In order to increase efficiency and variety, each instruction can use sections of common microcode states (similar to subroutines).

Table F–4 provides a cycle-by-cycle accounting for each of the 246 instruction and operand combinations for the TMS370 family microcontrollers. Each line consists of the opcode value, the mnemonic and operands, and a summary of all the cycles.

The table gives the minimum time for each instruction by showing the number of internal states. Each state lasts for four CLKIN or crystal periods, so each state represents 200 ns for a crystal running at 20 MHz. This time may increase if the application uses the autowait mode, peripheral autowait mode, or wait pin. **The wait modes affect only the "<<" cycles that access external memory.**

The instructions are typically expressed in this format:
OPCODE INSTR O1[,O2][,O3]

Operands are always read in increasing address order. This distinction is especially important for the MOV Rn,Pn and the MOV Pn,Rn instructions, in which the operands are in reversed address order. A and B are implied operands and do not require additional bytes.

The operand symbols used in Table F–4 are as follows:

= Immediate operand
#16 = Immediate 16-bit number
lab = 16-bit label
n = Immediate 8-bit number
Pd = Peripheral register containing destination byte
Pn = Peripheral register
Ps = Peripheral register containing source byte
ra = Relative address
Rd = Register containing destination byte
Rn = Register file
Rp = Register pair
Rpd = Destination register pair
Rps = Source register pair
Rs = Register containing source byte

The two-letter abbreviations used in each line are divided into two groups.

- The first group (shown in Table F–2) generates valid external bus cycles when accessing external memory.
- The second group (shown in Table F–3) operates only internally and does not generate valid bus cycles, because internal memory accesses do not give valid external bus cycles.

Table F–2. Possible Bus Cycles

	Name	Type	Description
OP	Opcode read	Long	Opcode fetch, $\overline{OCF}$, active during this cycle.
PO	Pseudo-operand read	Long	Operand is read then discarded, and PC does not advance. Operand usually becomes an opcode on next instruction cycle.
O*n*	Operand *n* read	Long	Read instruction's operands (*n*=operand order in instruction).
PF	Prefix byte		First byte of a two-byte opcode. $\overline{OCF}$ is active during this cycle and not during next OP cycle. Always 0F4h.
Pr	Peripheral read	Long	Read of 1000h–10FFh. The external reads are affected by PF auto-wait.
Pw	Peripheral write	Long	Write to 1000h–10FFh. The external writes are affected by PF auto-wait.
Lr	Long read	Long	General long read, range of 0–0FFFFh.
Lw	Long write	Long	General long write, range of 0–0FFFFh.
Vr	Vector read	Long	Reads vectors in trap table (7FC0–7FDF).
<<	Continuation	Long	Finish memory access, nominal one cycle. If external memory is accessed using autowait, PF autowait, or external wait signals, it could require two or more cycles. Only cycle type affected by wait modes.

Table F–3. Internal Cycles

	Name	Type	Description
IC	Internal cycle	Other	Indeterminate internal operation
RJ	Relative jump	Other	Add these if jump is taken; takes two cycles, so they always come in pairs.
SH	Stack push	Other	(SP)+1 –> (SP); n –> ((SP))
SP	Stack pop	Other	((SP)) –> n ; (SP)–1 –> (SP)
Sr	Stack read	Other	((SP)) usually dummy cycle
Ar	Register A read	Register	1-cycle register accesses
Br	Register B read	Register	1-cycle register accesses
Rr	Register read	Register	1-cycle register accesses
Aw	Register A write	Register	1-cycle register accesses
Bw	Register B write	Register	1-cycle register accesses
Rw	Register write	Register	1-cycle register accesses

F

Table F–4. Bus Activity Table

code	Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
29	ADC	#n,A	OC	<<	O1	<<	Aw	Rr	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
59	ADC	#n,B	OC	<<	O1	<<	Bw	Rr	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
79	ADC	#n,Rd	OC	<<	O2	<<	Br	Rr	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
69	ADC	B,A	OC	<<	PO	<<	Ar	Rr	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
19	ADC	Rs,A	OC	<<	O1	<<	Aw	Rr	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
39	ADC	Rs,B	OC	<<	O1	<<	Bw	Rr	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
49	ADC	Rs,Rd	OC	<<	O1	<<	Br	Rr	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
28	ADD	#n,A	OC	<<	O1	<<	Ar	Rr	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
58	ADD	#n,B	OC	<<	O1	<<	Br	Rr	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
78	ADD	#n,Rd	OC	<<	O2	<<	Br	Rr	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
68	ADD	B,A	OC	<<	PO	<<	Ar	Rr	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
18	ADD	Rs,A	OC	<<	O1	<<	Ar	Rr	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
38	ADD	Rs,B	OC	<<	O1	<<	Br	Rr	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
48	ADD	Rs,Rd	OC	<<	O1	<<	Br	Rr	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
23	AND	#n,A	OC	<<	O1	<<	Ar	Rr	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
53	AND	#n,B	OC	<<	O1	<<	Br	Rr	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
A3	AND	#n,Pd	OC	<<	O1	<<	O2	Pr	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
73	AND	#n,Rd	OC	<<	O1	<<	O2	Rr	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
83	AND	A,Pd	OC	<<	O1	<<	Ar	Pr	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
63	AND	B,A	OC	<<	PO	<<	Br	Ar	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
93	AND	B,Pd	OC	<<	O1	<<	Br	Pr	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
13	AND	Rs,A	OC	<<	O1	<<	Rr	Ar	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
33	AND	Rs,B	OC	<<	O1	<<	Rr	Br	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
43	AND	Rs,Rd	OC	<<	O1	<<	Rr	Rr	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
9C	BR	@Rp	OC	<<	O1	<<	Rr	O2	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
8C	BR	lab	OC	<<	O1	<<	O2	IC	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
AC	BR	lab(B)	OC	<<	O1	<<	O2	Br	Aw	Pw	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
F4	EC BR	n(Rp)	PF	<<	OC	<<	IC	IC	<<	O2	<<	IC	Rr	IC	IC	IC	IC	IC	SH	IC	IC	21	22
26	BTJO	#n,A,ra	OC	<<	O1	<<	Ar	IC	<<	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ
56	BTJO	#n,B,ra	OC	<<	O1	<<	Br	IC	<<	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ
A6	BTJO	#n,Pd,ra	OC	<<	O1	<<	O2	Pr	<<	IC	O3	<<	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ
76	BTJO	#n,Rd,ra	OC	<<	O1	<<	O2	Rr	<<	IC	O3	<<	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ
86	BTJO	A,Pd,ra	OC	<<	O1	<<	Ar	Pr	<<	IC	O2	<<	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ
66	BTJO	B,A,ra	OC	<<	PO	<<	Br	IC	<<	RJ	O1	<<	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ
96	BTJO	B,Pd,ra	OC	<<	O1	<<	Br	Pr	<<	IC	O2	<<	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ
16	BTJO	Rs,A,ra	OC	<<	O1	<<	Rr	Ar	<<	IC	O2	<<	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ
36	BTJO	Rs,B,ra	OC	<<	O1	<<	Rr	Br	<<	IC	O2	<<	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ
46	BTJO	Rs,Rd,ra	OC	<<	O1	<<	Rr	Rr	<<	IC	O2	<<	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ
27	BTJZ	#n,A,ra	OC	<<	O1	<<	Ar	IC	<<	RJ	O3	<<	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ
57	BTJZ	#n,B,ra	OC	<<	O1	<<	Br	IC	<<	RJ	O3	<<	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ
A7	BTJZ	#n,Pd,ra	OC	<<	O1	<<	O2	Pr	<<	IC	O3	<<	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ
77	BTJZ	#n,Rd,ra	OC	<<	O1	<<	O2	Rr	<<	IC	O3	<<	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ
87	BTJZ	A,Pd,ra	OC	<<	O1	<<	Ar	Pr	<<	IC	O2	<<	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ
67	BTJZ	B,A,ra	OC	<<	PO	<<	Br	IC	<<	RJ	O1	<<	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ
97	BTJZ	B,Pd,ra	OC	<<	O1	<<	Br	Pr	<<	IC	O2	<<	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ
17	BTJZ	Rs,A,ra	OC	<<	O1	<<	Rr	Ar	<<	IC	O2	<<	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ
37	BTJZ	Rs,B,ra	OC	<<	O1	<<	Rr	Br	<<	IC	O2	<<	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ
47	BTJZ	Rs,Rd,ra	OC	<<	O1	<<	Rr	O2	<<	Rr	O3	<<	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ	RJ
9E	CALL	@Rp	OC	<<	O1	<<	Rr	<<	<<	IC	SH	<<	IC	IC	IC	IC	IC	IC	IC	IC	IC	IC	IC
8E	CALL	lab	OC	<<	O1	<<	O2	IC	<<	SH	IC	<<	SH	SH	IC	IC	IC	IC	IC	IC	IC	IC	IC
AE	CALL	lab(B)	OC	<<	O1	<<	O2	IC	<<	IC	O2	<<	SH	IC	IC	IC	IC	IC	IC	IC	IC	IC	IC
F4	EE CALL	n(Rp)	PF	<<	OC	<<	IC	IC	<<	O1	<<	<<	Rr	IC	IC	IC	IC	IC	IC	IC	IC	IC	IC
code	Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

Table F–4. Bus Activity Table (Continued)

code	Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	
9F	CALLR	@Rp	OC	<<	O1	<<	Rr	Rr	IC	IC	IC	SH	IC	SH	IC	IC	.					.		
8F	CALLR	lab	OC	<<	O1	<<	O2	<<	IC	IC	IC	IC	SH	IC	SH	IC	IC					.		
AF	CALLR	lab(B)	OC	<<	O1	<<	O2	<<	Br	IC	IC	IC	IC	IC	SH	IC	SH	IC	IC			.		
F4 EF	CALLR	n(Rp)	PF	<<	OC	<<	IC	IC	O1	<<	O2	<<	Rr	Rr	IC	IC	IC	IC	IC	SH	IC	SH	IC	IC
B5	CLR	A	OC	<<	PO	<<	Ar	IC	IC	Aw							.					.		
C5	CLR	B	OC	<<	PO	<<	Br	IC	IC	Bw							.					.		
D5	CLR	Rd	OC	<<	O1	<<	Rr	Rw									.					.		
B0	CLRC		OC	<<	PO	<<	Ar	IC	IC	Ar	Aw	.					(same as TST A instruction)							
F3	CMP	n(SP),A	OC	<<	O1	<<	IC	RJ	Rr	IC		.					.					.		
2D	CMP	#n,A	OC	<<	O1	<<	Ar	IC				.					.					.		
5D	CMP	#n,B	OC	<<	O1	<<	Br	IC				.					.					.		
7D	CMP	#n,Rp	OC	<<	O1	<<	O2	<<	Rr	IC		.					.					.		
9D	CMP	@Rp,A	OC	<<	O1	<<	Rr	Rr	Lr	<<	Ar	IC					.					.		
6D	CMP	B,A	OC	<<	PO	<<	Br	IC	Ar	IC		.					.					.		
AD	CMP	lab(B),A	OC	<<	O1	<<	O2	<<	Br	IC	IC	Lr	<<	Ar	IC		.					.		
8D	CMP	lab,A	OC	<<	O1	<<	O2	<<	IC	Lr	<<	Ar	IC				.					.		
F4 ED	CMP	n(Rp),A	PF	<<	OC	<<	IC	IC	O1	<<	O2	<<	Rr	Rr	IC	IC	Lr	<<	Ar	IC		.		
1D	CMP	Rs,A	OC	<<	O1	<<	Rr	Ar	IC			.					.					.		
3D	CMP	Rs,B	OC	<<	O1	<<	Rr	Br	IC			.					.					.		
4D	CMP	Rs,Rd	OC	<<	O1	<<	Rr	O2	<<	Rr	IC	.					.					.		
A5	CMPBIT	Pname	OC	<<	O1	<<	O2	<<	Pr	<<	Pw	<<					(same as XOR #n,Pd instruction)							
75	CMPBIT	Rname	OC	<<	O1	<<	O2	<<	Rr	Rw		.					(same as XOR #n,Rn instruction)							
BB	COMPL	A	OC	<<	PO	<<	Ar	IC	IC	Aw		.					.					.		
CB	COMPL	B	OC	<<	PO	<<	Br	IC	IC	Bw		.					.					.		
DB	COMPL	Rn	OC	<<	O1	<<	Rr	Rw				.					.					.		
2E	DAC	#n,A	OC	<<	O1	<<	Ar	IC	IC	Aw		.					.					.		
5E	DAC	#n,B	OC	<<	O1	<<	Br	IC	IC	Bw		.					.					.		
7E	DAC	#n,Rd	OC	<<	O1	<<	O2	<<	Rr	IC	IC	Rw					.					.		
6E	DAC	B,A	OC	<<	PO	<<	Br	IC	Ar	IC	IC	Aw					.					.		
1E	DAC	Rs,A	OC	<<	O1	<<	Rr	Ar	IC	IC	Aw	.					.					.		
3E	DAC	Rs,B	OC	<<	O1	<<	Rr	Br	IC	IC	Bw	.					.					.		
4E	DAC	Rs,Rd	OC	<<	O1	<<	Rr	O2	<<	Rr	IC	IC	Rw				.					.		
B2	DEC	A	OC	<<	PO	<<	Ar	IC	IC	Aw		.					.					.		
C2	DEC	B	OC	<<	PO	<<	Br	IC	IC	Bw		.					.					.		
D2	DEC	Rn	OC	<<	O1	<<	Rr	Rw				.					.					.		
F4 F0	DINT		OC	<<	PO	<<	Sr	Br	IC			.					(same as LDST 00h instruction)							
F8	DIV	Rs,A	PF	<<	OC	<<	IC	IC	O1	<<	Ar	Rr	Br	IC	IC	IC	overflow condition					.		
F4 F8	DIV	Rs,A	PF	<<	OC	<<	IC	IC	O1	<<	Ar	Rr	Br	IC	IC	Aw	[Bw	IC	IC	IC	Aw]		IC	
																	. Repeat Block 55–63 cycles							
BA	DJNZ	A,ra	OC	<<	PO	<<	Ar	IC	IC	Aw	O1	<<	RJ	RJ			.					.		
CA	DJNZ	B,ra	OC	<<	PO	<<	Br	IC	IC	Bw	O1	<<	RJ	RJ			.					.		
DA	DJNZ	Rn,ra	OC	<<	O1	<<	Rr	Rw	O2	<<	RJ	RJ					.					.		
2F	DSB	#n,A	OC	<<	O1	<<	Ar	IC	IC	Aw		.					.					.		
5F	DSB	#n,B	OC	<<	O1	<<	Br	IC	IC	Bw		.					.					.		
7F	DSB	#n,Rd	OC	<<	O1	<<	O2	<<	Rr	IC	IC	Rw					.					.		
6F	DSB	B,A	OC	<<	PO	<<	Br	IC	Ar	IC	IC	Aw					.					.		
1F	DSB	Rs,A	OC	<<	O1	<<	Rr	Ar	IC	IC	Aw	.					.					.		
3F	DSB	Rs,B	OC	<<	O1	<<	Rr	Br	IC	IC	Bw	.					.					.		
4F	DSB	Rs,Rd	OC	<<	O1	<<	Rr	O2	<<	Rr	IC	IC	Rw				.					.		
F00C	EINT		OC	<<	PO	<<	Sr	Br	IC			.					(same as LDST 0Ch instruction)							
F004	EINTH		OC	<<	PO	<<	Sr	Br	IC			.					(same as LDST 04h instruction)							
F008	EINTL		OC	<<	PO	<<	Sr	Br	IC			.					(same as LDST 08h instruction)							
F6	IDLE		OC	<<	PO	<<	IC	IC				.					(usually exited by interrupt)							
code	Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	

Table F–4. Bus Activity Table (Continued)

code	Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
B3	INC	A	OC	<<	PO	<<	Ar	IC	IC	Aw													
C3	INC	B	OC	<<	PO	<<	Br	IC	IC	Bw													
D3	INC	Rn	OC	<<	O1	<<	Rr	Rw															
70	INCW	#n,Rp	OC	<<	O1	<<	O2	<<	Rr	Rw	IC	Rr	Rw										
B4	INV	A	OC	<<	PO	<<	Ar	IC	IC	Aw													
C4	INV	B	OC	<<	PO	<<	Br	IC	IC	Bw													
D4	INV	Rn	OC	<<	O1	<<	Rr	Rw															
A7	JBIT0	Pname,ra	OC	<<	O1	<<	O2	<<	Pr	<<	IC	O3	<<	RJ	RJ		(same as BTJZ #n,Pd,ra instruction)						
77	JBIT0	Rname,ra	OC	<<	O1	<<	O2	<<	Rr	IC	O3	<<	RJ	RJ			(same as BTJZ #n,Rn,ra instruction)						
A6	JBIT1	Pname,ra	OC	<<	O1	<<	O2	<<	Pr	<<	IC	O3	<<	RJ	RJ		(same as BTJO #n,Pd,ra instruction)						
76	JBIT1	Rname,ra	OC	<<	O1	<<	O2	<<	Rr	IC	O3	<<	RJ	RJ			(same as BTJO #n,Rn,ra instruction)						
01–0F	Jcnd	ra	OC	<<	O1	<<	IC	RJ	RJ														
00	JMP	ra •	OC	<<	O1	<<	IC	RJ	RJ														
99	JMPL	@Rp	OC	<<	O1	<<	Rr	Rr	IC	IC													
89	JMPL	lab	OC	<<	O1	<<	O2	<<	IC	IC	IC												
A9	JMPL	lab(B)	OC	<<	O1	<<	O2	<<	Br	IC	IC	IC	IC										
E9	JMPL	n(Rp)	PF	<<	OC	<<	IC	IC	O1	<<	O2	<<	Rr	Rr	IC	IC	IC	IC					
FD	LDSP		OC	<<	PO	<<	Sr	Br	IC														
F0	LDST	#n	OC	<<	O1	<<	IC	IC															
F2	MOV	A,n(SP)	OC	<<	O1	<<	IC	Ar	Rw														
F1	MOV	n(SP),A	OC	<<	O1	<<	IC	Rr	Aw														
22	MOV	#n,A	OC	<<	O1	<<	Ar	Aw															
52	MOV	#n,B	OC	<<	O1	<<	Br	Bw															
F7	MOV	#n,Pd	OC	<<	O1	<<	IC	O2	<<	IC	Pw	<<											
72	MOV	#n,Rs	OC	<<	O1	<<	O2	<<	Rr	Rw													
9A	MOV	@Rp,A	OC	<<	O1	<<	Rr	Rr	Lr	<<	Aw												
9B	MOV	A,@Rp	OC	<<	O1	<<	Rr	Rr	Ar	Lw	<<												
C0	MOV	A,B	OC	<<	PO	<<	Br	IC	IC	Ar	Bw												
8B	MOV	A,lab	OC	<<	O1	<<	O2	<<	IC	Ar	Lw	<<											
AB	MOV	A,lab(B)	OC	<<	O1	<<	O2	<<	Br	IC	IC	Ar	Lw	<<									
EB	MOV	A,n(Rpd)	PF	<<	OC	<<	IC	IC	O1	<<	O2	<<	Rr	Rr	IC	IC	Ar	LW	<<				
21	MOV	A,Pd	OC	<<	O1	<<	Ar	IC	Pw	<<													
D0	MOV	A,Rd	OC	<<	O1	<<	Rr	Ar	Rw														
62	MOV	B,A	OC	<<	PO	<<	Br	IC	Ar	Aw													
51	MOV	B,Pd	OC	<<	O1	<<	Br	IC	Pw	<<													
D1	MOV	B,Rd	OC	<<	O1	<<	Rr	Br	Rw														
AA	MOV	lab(B),A	OC	<<	O1	<<	O2	<<	Br	IC	IC	Lr	<<	Aw									
8A	MOV	lab,A	OC	<<	O1	<<	O2	<<	IC	Lr	<<	Aw											
EA	MOV	n(Rp),A	PF	<<	OC	<<	IC	IC	O1	<<	O2	<<	Rr	Rr	IC	IC	LR	<<	Aw				
80	MOV	Ps,A	OC	<<	O1	<<	Ar	Pr	<<	Aw													
91	MOV	Ps,B	OC	<<	O1	<<	Br	Pr	<<	Bw													
A2	MOV	Ps,Rd	OC	<<	O2r	<<	O1p	<<	Pr	<<	IC	Rw					operand order is reversed during assembly						
12	MOV	Rs,A	OC	<<	O1	<<	Rr	Ar	Aw														
32	MOV	Rs,B	OC	<<	O1	<<	Rr	Br	Bw														
71	MOV	Rs,Pd	OC	<<	O2p	<<	O1r	<<	Rr	IC	Pw	<<					operand order is reversed during assembly						
42	MOV	Rs,Rd	OC	<<	O1	<<	Rr	O2	<<	Rr	Rw												
A8	MOVW	#16n(B),Rpd	OC	<<	O1	<<	O2	<<	Br	IC	IC	O3	<<	IC	Rw	IC	Rw						
88	MOVW	#16n,Rpd	OC	<<	O1	<<	O2	<<	IC	O3	<<	IC	Rw	IC	Rw								
E8	MOVW	#n(Rp),Rpd	PF	<<	OC	<<	IC	IC	O1	<<	O2	<<	Rr	Rr	IC	IC	O3	<<	IC	RW	IC	RW	
98	MOVW	Rps,Rpd	OC	<<	O1	<<	Rr	Rr	O2	<<	IC	Rw	IC	Rw									
code	Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

Table F–4. Bus Activity Table (Continued)

code	Instruction		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2C	MPY	#n,A	OC	<<	O1	<<	Ar	. . . . .									45	cycles total						
5C	MPY	#n,B	OC	<<	O1	<<	Br	. . . . .									45	cycles total						
7C	MPY	#n,Rd	OC	<<	O1	<<	O2	<<	Rr	. . . . .							47	cycles total						
6C	MPY	B,A	OC	<<	PO	<<	Br	IC	Ar	. . . . .							47	cycles total						
1C	MPY	Rs,A	OC	<<	O1	<<	Rr	Ar	Br	IC	IC	Br	Bw	IC	Aw	. . . . .	46	cycles total						
3C	MPY	Rs,B	OC	<<	O1	<<	Rr	Br	. . . . .								46	cycles total						
4C	MPY	Rs,Rd	OC	<<	O1	<<	Rr	O2	<<	Rr	. . . . .						48	cycles total						
FF	NOP		OC	<<	PO	<<	Sr	Br	IC			.					.				.			
24	OR	#n,A	OC	<<	O1	<<	Ar	Aw				.					.				.			
54	OR	#n,B	OC	<<	O1	<<	Br	Bw				.					.				.			
A4	OR	#n,Pd	OC	<<	O1	<<	O2	<<	Pr	<<	Pw	<<					.				.			
74	OR	#n,Rd	OC	<<	O1	<<	O2	<<	Rr	Rw		.					.				.			
84	OR	A,Pd	OC	<<	O1	<<	Ar	Pr	<<	Pw	<<	.					.				.			
64	OR	B,A	OC	<<	PO	<<	Br	IC	Ar	Aw		.					.				.			
94	OR	B,Pd	OC	<<	O1	<<	Br	Pr	<<	Pw	<<	.					.				.			
14	OR	Rs,A	OC	<<	O1	<<	Rr	Ar	Aw			.					.				.			
34	OR	Rs,B	OC	<<	O1	<<	Rr	Br	Bw			.					.				.			
44	OR	Rs,Rd	OC	<<	O1	<<	Rr	O2	<<	Rr	Rw	.					.				.			
B9	POP	A	OC	<<	PO	<<	Ar	IC	IC	SP	Aw	.					.				.			
C9	POP	B	OC	<<	PO	<<	Br	IC	IC	SP	Bw	.					.				.			
D9	POP	Rd	OC	<<	O1	<<	Rr	SP	Rw			.					.				.			
FC	POP	ST	OC	<<	PO	<<	SP	Br	IC	IC		.					.				.			
B8	PUSH	A	OC	<<	PO	<<	Ar	IC	IC	IC	SH	.					.				.			
C8	PUSH	B	OC	<<	PO	<<	Br	IC	IC	IC	SH	.					.				.			
D8	PUSH	Rs	OC	<<	O1	<<	Rr	IC	SH			.					.				.			
FB	PUSH	ST	OC	<<	PO	<<	Sr	Br	IC	SH		.					.				.			
BE	RL	A	OC	<<	PO	<<	Ar	IC	IC	Aw		.					.				.			
CE	RL	B	OC	<<	PO	<<	Br	IC	IC	Bw		.					.				.			
DE	RL	Rn	OC	<<	O1	<<	Rr	Rw				.					.				.			
BF	RLC	A	OC	<<	PO	<<	Ar	IC	IC	Aw		.					.				.			
CF	RLC	B	OC	<<	PO	<<	Br	IC	IC	Bw		.					.				.			
DF	RLC	Rn	OC	<<	O1	<<	Rr	Rw				.					.				.			
BC	RR	A	OC	<<	PO	<<	Ar	IC	IC	Aw		.					.				.			
CC	RR	B	OC	<<	PO	<<	Br	IC	IC	Bw		.					.				.			
DC	RR	Rn	OC	<<	O1	<<	Rr	Rw				.					.				.			
BD	RRC	A	OC	<<	PO	<<	Ar	IC	IC	Aw		.					.				.			
CD	RRC	B	OC	<<	PO	<<	Br	IC	IC	Bw		.					.				.			
DD	RRC	Rn	OC	<<	O1	<<	Rr	Rw				.					.				.			
FA	RTI		OC	<<	PO	<<	SP	Br	IC	SP	SP	IC	IC	IC			(PCl,PCm, ST)							
F9	RTS		OC	<<	PO	<<	SP	Br	IC	SP	IC	.					(PCl, PCm)							
2B	SBB	#n,A	OC	<<	O1	<<	Ar	Aw				.					.				.			
5B	SBB	#n,B	OC	<<	O1	<<	Br	Bw				.					.				.			
7B	SBB	#n,Rd	OC	<<	O1	<<	O2	<<	Rr	Rw		.					.				.			
6B	SBB	B,A	OC	<<	PO	<<	Br	IC	Ar	Aw		.					.				.			
1B	SBB	Rs,A	OC	<<	O1	<<	Rr	Ar	Aw			.					.				.			
3B	SBB	Rs,B	OC	<<	O1	<<	Rr	Br	Bw			.					.				.			
4B	SBB	Rs,Rd	OC	<<	O1	<<	Rr	O2	<<	Rr	Rw	.					.				.			
A3	SBIT0	Pn	OC	<<	O1	<<	O2	<<	Pr	<<	Pw	<<					(same as AND #n,Pd)							
73	SBIT0	Rn	OC	<<	O1	<<	O2	<<	Rr	Rw		.					(same as AND #n,Rn)							
A4	SBIT1	Pn	OC	<<	O1	<<	O2	<<	Pr	<<	Pw	<<					(same as OR #n,Pd)							
74	SBIT1	Rn	OC	<<	O1	<<	O2	<<	Rr	Rw		.					(same as OR #n,Rn)							
F8	SETC		OC	<<	PO	<<	Sr	Br	IC			.					.				.			
FE	STSP		OC	<<	PO	<<	Sr	Br	IC	Bw		.					.				.			
code	Instruction		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

Table F–4. Bus Activity Table (Continued)

code	Instruction		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2A	SUB	#n,A	OC	<<	O1	<<	Ar	Aw																
5A	SUB	#n,B	OC	<<	O1	<<	Br	Bw																
7A	SUB	#n,Rd	OC	<<	O1	<<	O2	<<	Rr	Rw														
6A	SUB	B,A	OC	<<	PO	<<	Br	IC	Ar	Aw														
1A	SUB	Rs,A	OC	<<	O1	<<	Rr	Ar	Aw															
3A	SUB	Rs,B	OC	<<	O1	<<	Rr	Br	Bw															
4A	SUB	Rs,Rd	OC	<<	O1	<<	Rr	O2	<<	Rr	Rw													
B7	SWAP	A	OC	<<	PO	<<	Ar	IC	IC	IC	IC	IC	Aw											
C7	SWAP	B	OC	<<	PO	<<	Br	IC	IC	IC	IC	IC	Bw											
D7	SWAP	Rn	OC	<<	O1	<<	Rr	IC	IC	IC	Rw													
EF–E0	TRAP	n	OC	<<	PO	<<	IC	IC	IC	IC	SH	Vr	<<	SH	Vr	<<								
B0	TST	A	OC	<<	PO	<<	Ar	IC	IC	Ar	Aw													
C6	TST	B	OC	<<	PO	<<	Br	IC	IC	Br	Bw	Bw												
B6	XCHB	A	OC	<<	PO	<<	Ar	IC	IC	Br	Bw	Aw												
C6	XCHB	B	OC	<<	PO	<<	Br	IC	IC	Br	Bw	Bw												
D6	XCHB	Rd	OC	<<	O1	<<	Rr	Br	Bw	Rw														
25	XOR	#n,A	OC	<<	O1	<<	Ar	Aw																
55	XOR	#n,B	OC	<<	O1	<<	Br	Bw																
A5	XOR	#n,Pd	OC	<<	O1	<<	O2	<<	Pr	<<	Pw	<<												
75	XOR	#n,Rd	OC	<<	O1	<<	O2	<<	Rr	Rw														
85	XOR	A,Pd	OC	<<	O1	<<	Ar	Pr	<<	Pw	<<													
65	XOR	B,A	OC	<<	PO	<<	Br	IC	Ar	Aw														
95	XOR	B,Pd	OC	<<	O1	<<	Br	Pr	<<	Pw	<<													
15	XOR	Rs,A	OC	<<	O1	<<	Rr	Ar	Aw															
35	XOR	Rs,B	OC	<<	O1	<<	Rr	Br	Bw															
45	XOR	Rs,Rd	OC	<<	O1	<<	Rr	O2	<<	Rr	Rw													
code	Instruction		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

Notes:

1) Opcodes and operands are executed in the same order as in the written instruction. (except MOV Rs,Pd and MOV Ps,Rd).
2) All register pairs are accessed least significant byte then most significant byte. (n then, n–1).
3) Calls push PCH then PCL.
4) All external writes occur on the last cycle of the instruction.
5) All instructions make at least one operand fetch, even if not needed. The PC does not advance; it treats the pseudo-operand as an opcode on the next opcode fetch. Identified by PO.
6) MPY performs register A,B accesses and internal cycles for the number of cycles shown.
7) DIV execution time depends on the values divided but ranges from 55–63 cycles—fourteen cycles if overflow is detected. Overflow is detected if divisor <= dividend MSB.

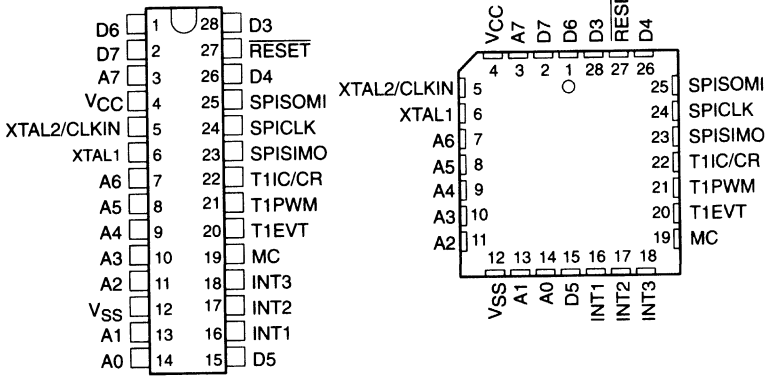
Appendix G

Device Pinouts

This appendix provides pinouts for the following individual device categories. For pin descriptions, refer to Chapter 2.

- TMS370Cx1x devices
- TMS370Cx2x devices
- TMS370Cx3x devices
- TMS370Cx4x devices
- TMS370Cx5x devices

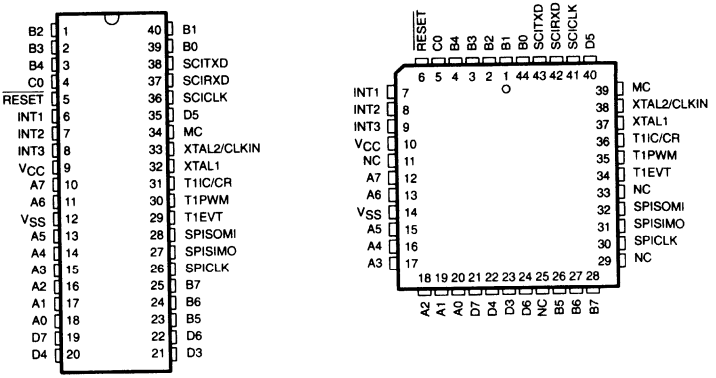
Figure G–1. Pinouts for TMS370Cx1x Devices (Top View)



A. 28-Pin PDIP (N)
B. 28-Pin CDIP (JD)

C. 28-Pin PLCC (FN)
D. 28-Pin CLCC (FZ)

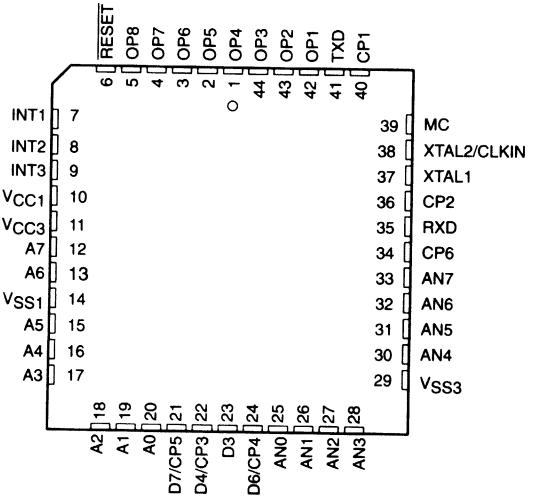
Figure G–2. Pinouts for TMS370Cx2x Devices (Top View)



A. 40-Pin PDIP (N)
B. 40-Pin CDIP (JD)
C. 40-Pin SDIP (N2)
D. 40-Pin CSDIP (JC)

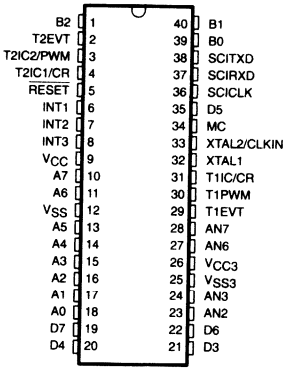
E. 44-Pin PLCC (FN)
F. 44-Pin CLCC (FZ)

Figure G–3. Pinout for TMS370Cx3x Devices (Top View)

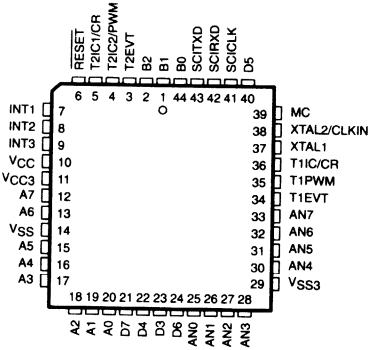


A. 44-Pin PLCC (FN)
B. 44-Pin CLCC (FZ)

Figure G–4. Pinouts for TMS370Cx4x Devices (Top View)

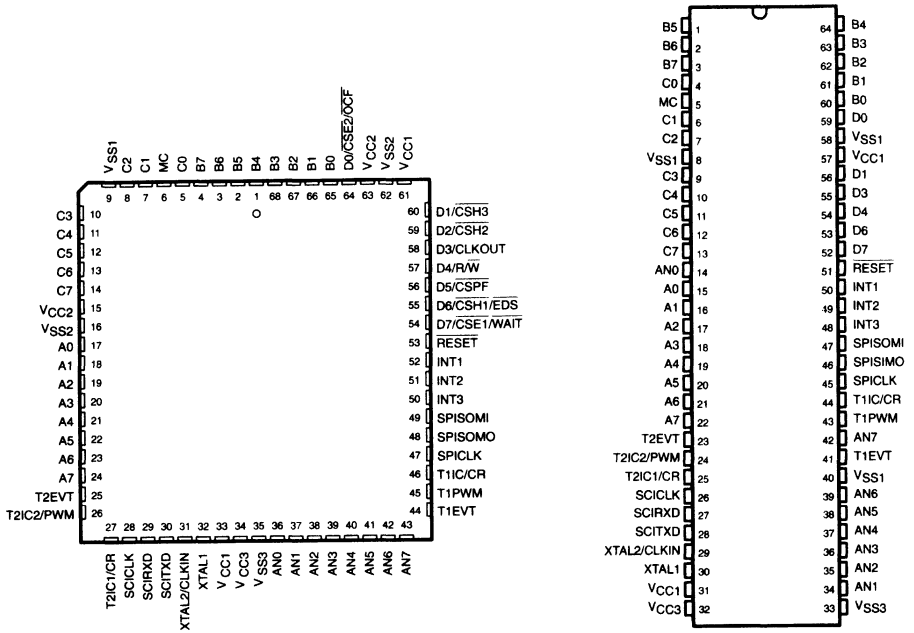


A. 40-Pin PDIP (N)
B. 40-Pin CDIP (JD)
C. 40-Pin SDIP (N2)
D. 40-Pin CSDIP (JC)



E. 44-Pin PLCC (FN)
F. 44-Pin CLCC (FZ)

Figure G–5. Pinouts for TMS370Cx5x Devices (Top View)



A. 68-Pin PLCC (FN)
B. 68-Pin CLCC (FZ)

C. 64-Pin SDIP (NM)
D. 64-Pin CSDIP (JN)

Appendix H

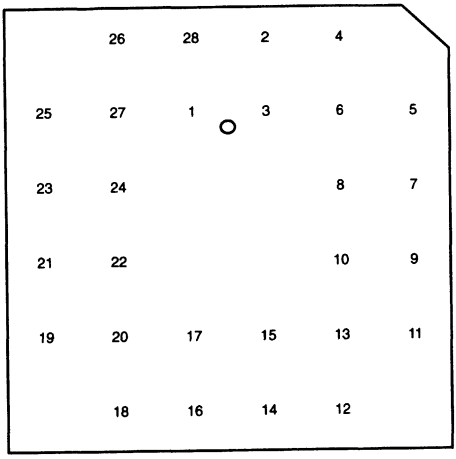
PLCC-to-PGA Socket Pinouts

This appendix shows the pinouts for the standard PLCC-to-PGA sockets that are commonly used in prototype and production applications. You can use these pinouts when you wirewrap your breadboard with a socket. These diagrams will make constructing, debugging, and troubleshooting with the TMS370 family quicker and easier.

The figures shown in this appendix are for sockets that correspond to the following package types:

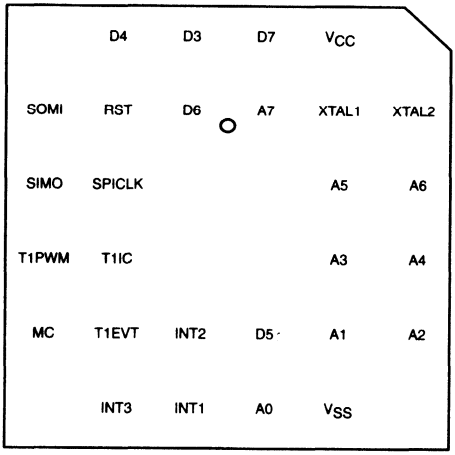
- ❑ 28-pin PLCC/CLCC
- ❑ 44-pin PLCC/CLCC
- ❑ 68-pin PLCC/CLCC

Figure H–1. 28-Pin PGA Pinout



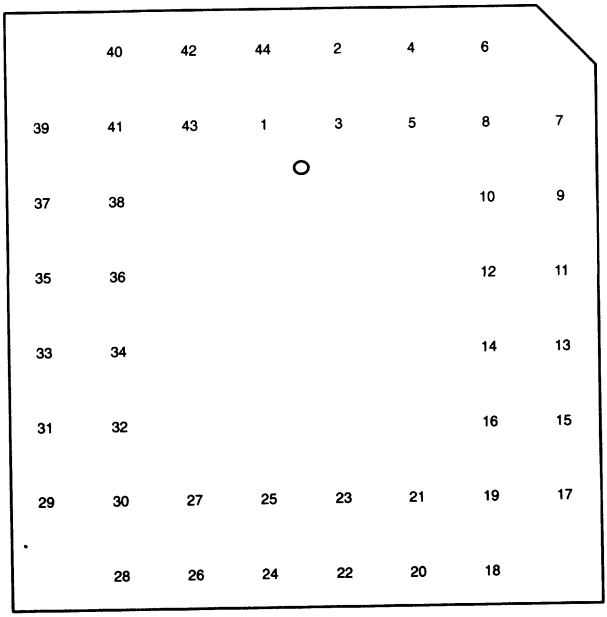
BOTTOM VIEW

Figure H–2. TMS370Cx1x Device PGA Pinout



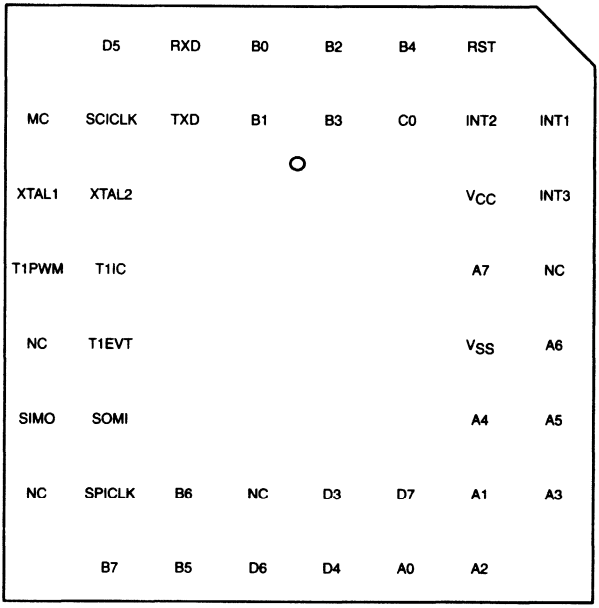
BOTTOM VIEW

Figure H–3.44-Pin PGA Pinout



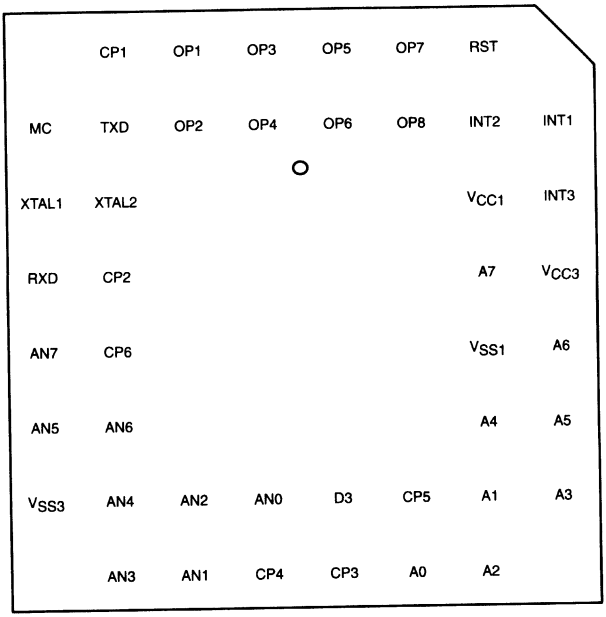
BOTTOM VIEW

Figure H–4. TMS370Cx2x Device PGA Pinout



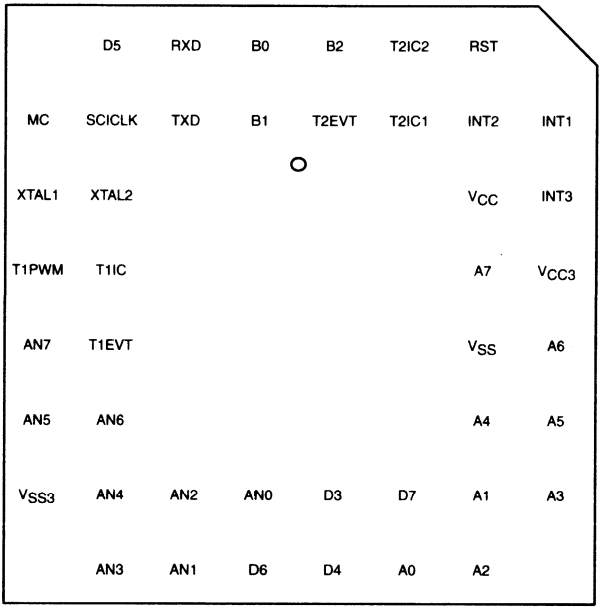
BOTTOM VIEW

Figure H–5. TMS370Cx3x Device PGA Pinout



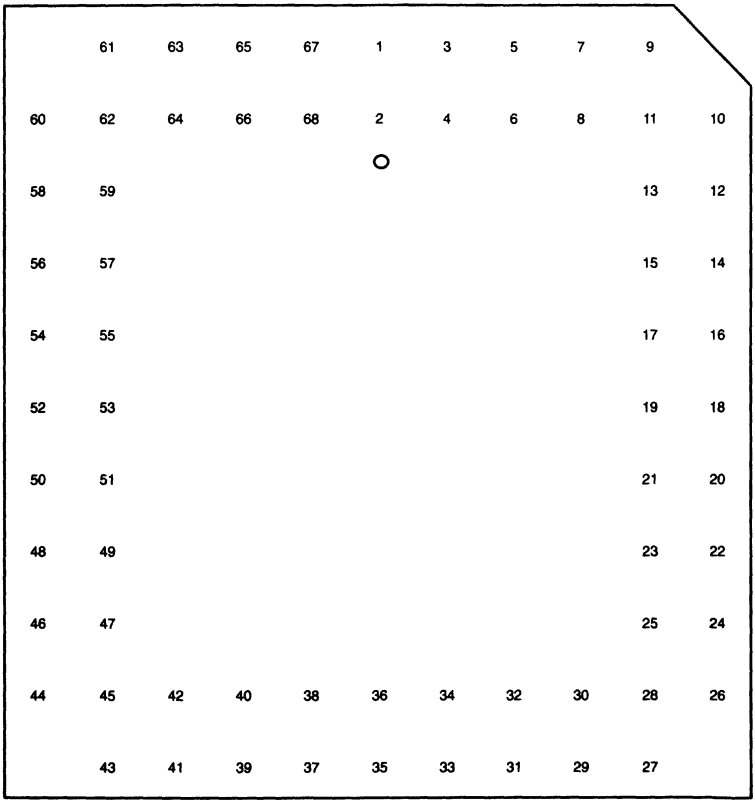
BOTTOM VIEW

Figure H–6. TMS370Cx4x Device PGA Pinout



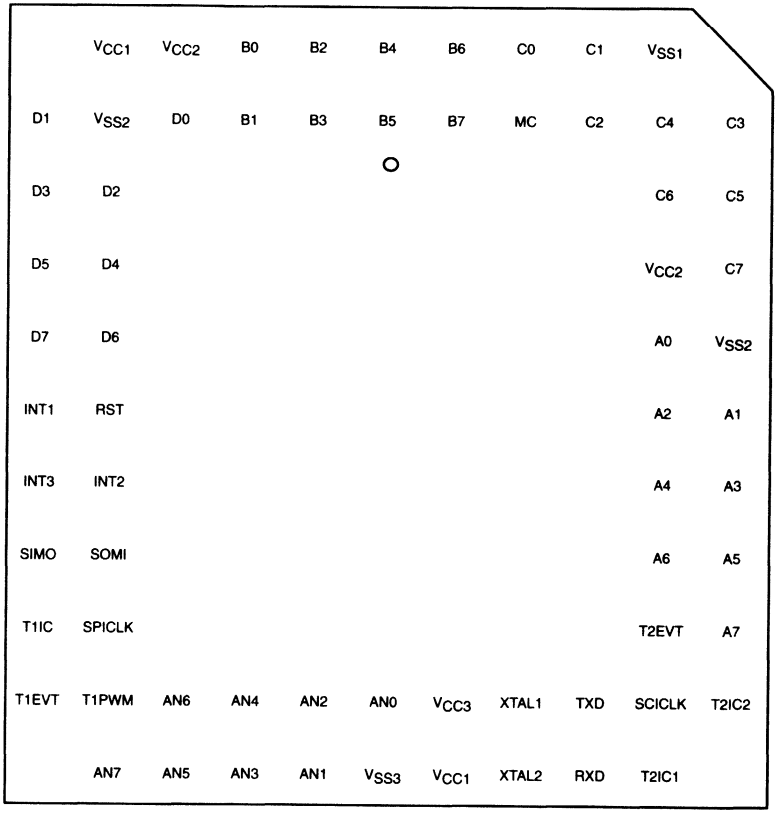
BOTTOM VIEW

Figure H–7.68-Pin PGA Pinout



BOTTOM VIEW

Figure H–8. TMS370Cx5x Device PGA Pinout



BOTTOM VIEW

Appendix I

PACT.H

The macros defined in the PACT.H file make it easier to define the initial values for the PACT command/definition area. You can obtain the latest version of this file from the microcontroller bulletin board. This appendix describes some of the peculiarities of using these macros.

Topic **Page**

I.1 General Comments

I.1.1 Addressing Commands and Definitions in Dual-Port RAM

The initial value of the command/definition area is usually defined in program memory and then copied to dual-port RAM during the initialization routine. Section 14.5 contains an example program of how to do this. If the values in a command or definition are to be read or written while the PACT module is running, the runtime location of that command or definition must be known. Each of the six macros allows for an optional parameter (a register label) that, if passed as a symbol, will equate the symbol to the register containing the least significant byte of the command or definition. For example, you can use the MOVW instruction with the register label parameter to change the compare value of a Standard Compare command from its initial value of 80h to 100h.

```
stdcmp   80h,op2,enable|opp_act,pwm1len ;if the command
                                        ;looks like this
movw     #100h,pwm1len                  ;this modifies
                                        ;the compare value
```

Often, the code that reads from or writes to the command/definition area is in a separate file from the code that initializes this area. You can make references to specific commands or definitions by declaring the register label parameter as .globreg. Bytes other than the least significant byte in a command or definition can be referenced as offsets from the least significant byte. Likewise, individual bits can be referenced according to the definition of the least significant byte. For example, the byte containing the output pin value of the Standard Compare command and the bit that enables that command can be referenced in this way:

```
         .globreg pwm1len           ;PWM 1 compare value
pwm1pin  .equ pwm1len-2             ;PWM 1 output pin byte
pwm1en   .dbit 3,pwm1len-3          ;PWM 1 enable bit
```

The assembler requires that global register symbols used in equates be declared .globreg before they are used. You can do this by creating a file that has all of the .globreg symbols at the top, followed by the equates and the bit definitions. This file then can be included at the beginning of the file that defines the command/definition area and again at the beginning of each file that references that area. Using this technique, you have only to reassemble the module that defines the command/definition area if this area changes; modules that reference specific commands or definitions will be corrected at link time.

For the macro to be able to calculate the final destination of the command or definition, you must define two symbols (cmd_st and table) before the macro is invoked. See the example in subsection 14.5.1.

I.1.2 Defining Output Pins

The symbols OP1 through OP8 are defined in the PACT.H file to make it easier to read the output pin selected for a specific command. The numbers one through eight or any previously defined symbol that equates to a value in this range could be used. The macros automatically subtract one and then shift the bits into their proper place for the requested command.

I.1.3 Defining Actions

Twenty-three possible actions are equated to numeric values at the beginning of the PACT.H file. The commands and definitions for which each action is valid are shown in the comment section beside the equate statement. The | operator concatenates multiple actions. The numeric value assigned each action is valid only when used in the macro expansion. These symbols should not be used to set or clear the action bits while the PACT module is running.

I.2 Comments for Specific Macros

I.2.1 Standard Compare Command

It is not necessary to specify the compare value or the output pin if the enable action is not specified. For example, the Standard Compare command can be used as a dummy command so that a definition may follow as shown below.

```
stdcmp  ,,nxt_def ;dummy command, next line is a definition
```


I.2.2 Conditional Compare Command

The time compare value that is passed to this macro will be reduced by two before it is encoded into the command. This allows the value passed to the macro to more accurately reflect the time delay until the specified actions occur. The time compare value must be greater than or equal to two.

I.2.3 Virtual Timer Definition

The virtual timer period value passed to the macro will be reduced by two so that the desired period is achieved. This value must be able to be represented in the maximum value format described in subsection 12.5.2. If you want to have the macro truncate the value to fit into the maximum value format without generating an error, comment out the appropriate error lines in the PACT.H file.

The initial timer value is optional; if it is used, it must be an even number.

I.2.4 Baud Timer Definition

The maximum count value for this definition is derived by the equation given in Section 12.8. The value obtained must then meet the maximum value format, or an error will be generated.

The initial timer value is optional; if it is used, it must be an even number.

I.3 PACT.H Macros

```
;This file contains macro definitions for all PACT commands and definitions.
;All the actions desired in each of the commands/definitions must be passed
;in the macro as they are defined in the following equate table. All the
;actions are passed as one parameter in the macro. These actions are
;concatenated by '|' to form one parameter. These actions can be defined in
;any order.
;NOTE: If an action, which is not a valid action for a particular command
;      or definition, is used in that command, incorrect assembly may occur
;      without flagging an error.
;If the user wants to use different action names, the equate table must be
;modified.
;                                              Version 1.00
;OUTPUT PINS
op1         .EQU  1
op2         .EQU  2
op3         .EQU  3
op4         .EQU  4
op5         .EQU  5
op6         .EQU  6
op7         .EQU  7
op8         .EQU  8

; ACTIONS              VTD BRD OTD SCC CCC DEC
clr_pin     .EQU 0   ;                 x   x       Default condition
clr_evt1    .EQU 0   ;                             x  Default condition
nxt_def     .EQU 1   ;                 x   x       x  Next entry is a def
int_cmp     .EQU 2   ;                 x   x          Interrupt on compare =
int_evt1    .EQU 2   ;                             x  Interrupt on event 1
int_trst    .EQU 4   ;  x              x              Interrupt on timer = 0
enable      .EQU 8   ;  x         x    x           x  Enable timer or pin
rst_def_tmr  .EQU 10h ;              x              Reset def tmr on evt max
rst_def_ev2  .EQU 10h ;                             x  Reset def tmr on evt 2
set_pin     .EQU 20h ;               x   x          Set output pin on =
set_evt1    .EQU 20h ;                             x  Set output pin on evt1
step        .EQU 40h ;           x   x           x  Go to half resolution
int_evt     .EQU 80h ;           x                  Interrupt on each event
int_max_evt  .EQU 100h;              x              Interrupt on max event
opp_act     .EQU 200h;                   x         xOpp action on timer rst
int_evt2    .EQU 400h;                             x  Int on event 2
tx          .EQU 800h;      x                      Use as TX bit rate
rx          .EQU 1000h;     x                      Use as RX bit rate
vir_cap     .EQU 2000h;          x                 Cap virt timer each evt
cap_def_ev1 .EQU 2000h;                            x  Cap def timer on event 1
def_cap     .EQU 4000h;          x                 Cap def timer on evt max
cap_def_ev2 .EQU 4000h;                            x  Cap def timer on event 2
evt_plus1   .EQU 8000h;                      x        Action on event plus 1


;STANDARD COMPARE COMMAND
; stdcmp <compare value>,<pin>,<actions>,<register label>
```

```
;
; compare value: 16-bit timer compare value
; pin: Output pin selection. (D18-D20)
; Possible actions:enable,set_pin,clr_pin,int_cmp,step,
;                  nxt_def,int_trst,opp_act
; register label:  a symbol to be equated to the register containing the
;                  least significant byte of this command


STDCMP     .MACRO cmpval,pin,actions,lab
           .var   b1,b2,b3,b4
           .if    ((pin.v<1)|(pin.v>8))&((actions.v&enable)=enable)
** ERROR, pin selection is illegal **
           .endif
           .if    (actions.v&0FD90h)!=0
** ERROR, illegal action specified **
           .endif
           .asg   cmpval.v&0FFh,b1.v
           .asg   (cmpval.v>>8)&0FFh,b2.v
           .if    (pin.v<1)|(pin.v>8)
           .asg   1,pin.v
           .endif
           .asg   pin.v-1,pin.v
           .asg   actions.v&63h|pin.v<<2,b3.v
           .asg   actions.v&0Ch|actions.v>>8&2h,b4.v
           .byte  b1.v,b2.v,b3.v,b4.v
           .if    lab.l!=0
           .asg   cmd_st-$+table+4,b1.v
:lab:      .equ   r:b1.v:
           .endif
           .ENDM


;CONDITIONAL COMPARE COMMAND
; CONCMP <event compare value>,<time compare value>,<pin>,<actions>,
;          <register label>
;
; event compare value: 8-bit value compared to the event counter
; time compare value: 16-bit value compared to the referred timer
; pin: Output pin  (only pin 1-7 are valid)
; Possible actions: nxt_def,int_cmp,set_pin,clr_pin,evt_plus1
; register label: a symbol to be equated to the register containing the
;                 least significant byte of this command



CONCMP     .MACRO evcmpval,cmpval,pin,actions,lab
           .var   b1,b2,b3,b4
           .if    (cmpval.v=0)|(cmpval.v=1)
** ERROR, compare value must be greater than 1 **
           .endif
```

```
            .asg   cmpval.v-2,cmpval.v
            .if    (pin.v>7)|(pin.v<0)
** ERROR, pin selection is illegal **
            .endif
            .if    (actions.v&07FDCh)!=0
** ERROR, illegal action specified **
            .endif
            .if    (evcmpval.v>255)|(evcmpval.v<0)
** ERROR, Event counter compare value out of range **
            .endif
            .asg   cmpval.v&0FFh,b1.v
            .asg   (cmpval.v>>8)&0FFh,b2.v
            .if    pin.v=0
            .asg   7,pin.v
            .else
            .asg   pin.v-1,pin.v
            .endif
            .asg   80h|actions.v&23h|pin.v<<2|actions.v>>9&40h,b3.v
            .asg   evcmpval.v,b4.v
            .byte  b1.v,b2.v,b3.v,b4.v
            .if    lab.l!=0
            .asg   cmd_st-$+table+4,b1.v
:lab:       .equ   r:b1.v:
            .endif
            .ENDM

;DOUBLE EVENT COMMAND
; DEVCMP <event value 1>,<event value 2>,<output pin>,<actions>,
;          <register label>
;
; event value 1: 8-bit value compared to the event counter
; event value 2: 8-bit value compared to the event counter
; pin: Output pin
; Possible actions: nxt_def,int_evt1,set_pin,clr_pin,step,opp_act,int_evt2
;                   rst_def_ev2,cap_def_ev1,cap_def_ev2,enable,
; register label: a symbol to be equated to the register containing the
;                   least significant byte of this command


DEVCMP      .MACRO e1cmpval,e2cmpval,pin,actions,lab
            .var   b1,b2,b3,b4
            .if    (e1cmpval.v>255)|(e1cmpval.v<0)
** ERROR, Event compare 1 value out of range **
            .endif
            .if    (e2cmpval.v>255)|(e2cmpval.v<0)
** ERROR, Event compare 2 value out of range **
            .endif
            .asg   e1cmpval.v,b1.v
            .asg   e2cmpval.v,b2.v
            .if    (pin.v<1)|(pin.v>8)
            .asg   1,pin.v
** ERROR, pin selection is illegal **
            .endif
```

```
          .asg   pin.v-1,pin.v
          .if    (actions.v&09984h)!=0
** ERROR, illegal action specified **
          .endif
          .asg   actions.v&063h|pin.v<<2,b3.v
          .asg   actions.v&18h|actions.v>>8&66h|1,b4.v
          .byte  b1.v,b2.v,b3.v,b4.v
          .if    lab.l!=0
          .asg   cmd_st-$+table+4,b1.v
:lab:     .equ   r:b1.v:
          .endif
          .ENDM


;VIRTUAL TIMER DEFINITION
; virtmr <period>,<actions>,<initial timer value>,<register label>
;
; period: The period of the virtual timer, the maximum count plus 1
; Possible actions: enable,int_trst
; initial timer value: 16-bit virtual timer initial value.
; register label: a symbol to be equated to the register containing the
;                 least significant byte of this definition


VIRTMR    .MACRO period,actions,tmrval,lab
          .var   b1,b2,b3,b4
          .if    (period.v=0)|(period.v=1)
** Error, Max Timer value must be greater than 2 **
          .endif
          .if    (actions.v&0FFF3h)!=0
** ERROR, illegal action specified **
          .endif
          .asg   period.v-2,period.v
          .asg   tmrval.v&0FEh,b1.v
          .asg   (tmrval.v>>8)&0FFh,b2.v
          .if    ((period.v>>8)&0FFh) > 1Fh
          .asg   (period.v>>9)&70h|(period.v<<3)&80h|08h,b3.v
          .if    (period.v&0Fh)!=0
** ERROR, Max. Timer value truncated in last 4 bits **
          .endif
          .else
          .asg   (period.v<<3)&0F0h|(actions.v&0Ch)>>1,b3.v
          .if    period.v&01h!=0
** ERROR, Max. Timer value truncated in last bit **
          .endif
          .endif
          .if    tmrval.v&01h!=0
** ERROR, Timer value truncated in last bit **
          .endif
          .asg   b3.v|actions.v&0Ch>>1,b3.v
          .asg   (period.v>>5)&0FFh,b4.v
          .byte  b1.v,b2.v,b3.v,b4.v
```

```
            .if    lab.l!=0
            .asg   cmd_st-$+table+4,b1.v
:lab:       .equ   r:b1.v:
            .endif
            .ENDM


;BAUD TIMER DEFINITION
; BRTMR <maximum count>,<actions>,<initial timer value>,<register label>
;
; maximum count: number that determines the baud rate
; initial timer value: 16-bit virtual timer initial value
; Possible actions: RX,TX
; register label: a symbol to be equated to the register containing the
;                 least significant byte of this definition


BRTMR       .MACRO maxcount,actions,tmrval,lab
            .var   b1,b2,b3,b4
            .if    ((actions.v&0E7FFh)!=0)
** ERROR, illegal action specified **
            .endif
            .asg   tmrval.v&0FEh,b1.v
            .asg   (tmrval.v>>8)&0FFh,b2.v
            .if    ((maxcount.v>>8)&0FFh) > 1Fh
            .asg   (maxcount.v>>9)&70h|(maxcount.v<<3)&80h|08h,b3.v
            .if    maxcount.v&0Fh!=0
** ERROR, Max. Timer value truncated in last 4 bits **
            .endif
            .else
            .asg   (maxcount.v<<3)&0F0h,b3.v
            .if    maxcount.v&01h!=0
** ERROR, Max. Timer value truncated in last bit **
            .endif
            .endif
            .if    tmrval.v&01h!=0
** ERROR, Timer value truncated in last bit **
            .endif
            .asg   (maxcount.v>>5)&0FFh,b4.v
            .asg   b3.v|((actions.v&1800h)>>10)|1,b3.v
            .byte  b1.v,b2.v,b3.v,b4.v
            .if    lab.l!=0
            .asg   cmd_st-$+table+4,b1.v
:lab:       .equ   r:b1.v:
            .endif
            .ENDM


;OFFSET TIMER DEFINITION
; OFSTMR <max event count>,<actions>,<initial value>,<register label>
;
; max event count: The maximum value the event counter may reach before
```

```
;                 being reset.
; Possible actions: step,int_max_evt,enable,rst_def_tmr,
;                 vir_cap,def_cap,int_evt
; initial value: 16-bit initial timer value
; register label: a symbol to be equated to the register containing the
;                 least significant byte of this definition


OFSTMR      .MACRO maxcount,actions,tmrval,lab
            .var   b1,b2,b3,b4
            .if    (maxcount.v>255)|(maxcount.v<0)
** ERROR, Maximum event value out of range **
            .endif
            .if    ((actions.v&09E27h)!=0)
** ERROR, illegal action specified **
            .endif
            .asg   (tmrval.v&0FFh|1),b1.v
            .asg   (tmrval.v>>8)&0FFh,b2.v
            .asg   (actions.v&090h)|((actions.v&8)>>1)|(actions.v&40h)>>6,b3.v
            .asg   b3.v|((actions.v&100h)>>7)|((actions.v>>8)&60h),b3.v
            .asg   maxcount.v&0FFh,b4.v
            .byte  b1.v,b2.v,b3.v,b4.v
            .if    lab.l!=0
            .asg   cmd_st-$+table+4,b1.v
:lab:       .equ   r:b1.v:
            .endif
            .ENDM
```

Appendix J

Glossary

A

absolute addressing mode: An addressing mode in which code or operands produce the actual address.

addressing mode: The method by which an instruction calculates the location of its required data.

analog-to-digital (A/D) converter: An 8-bit successive-approximation converter with internal sample-and-hold circuitry.

ANSI C: A version of the C programming language that conforms to the C standards defined by the *American National Standards Institute.*

archiver: A software program that allows you to collect several individual files into a single file called an archive library. The archiver also allows you to add, delete, extract, or replace members of the archive library.

assembler: A software program that creates a machine-language program from a source file that contains assembly language instructions, directives, and macro directives. The assembler substitutes absolute operation codes for symbolic operation codes, and absolute or relocatable addresses for symbolic addresses.

assembly language: A symbolic language that describes the binary machine code in a more readable form. Each of the 73 unique instructions of the TMS370 family converts to one machine operation.

asynchronous communications mode: An SCI mode that needs no synchronizing clock. This format consists of a start bit followed by data bits, an optional parity bit, and a stop bit. This format is commonly used with RS-232-C communications and PC serial ports.

B

baud: The communication speed for serial ports; equivalent to bits per second.

BCD: *Binary coded decimal.* Each 4-bit nibble expresses a digit from 0–9 and usually packs two digits to a byte, giving a range of 0–99.

breakpoint, trace, and timing (BTT) features: A set of features supported by the BTT board (included with the XDS/22 emulation system). These features allow you to set hardware breakpoints, collect trace samples, and perform timing analysis.

buffer pointer: A 5-bit register in the PACT module peripheral frame that points to the next available location in the circular capture buffer.

byte: A sequence of 8 adjacent bits operated upon as a unit.

C

C: A high-level, general-purpose programming language useful for writing compilers and operating systems and for programming microprocessors.

C compiler: A program that translates C source statements into assembly language source statements.

capture register: A timer 2 register that is loaded with the 16-bit counter value on the occurrence of an external input transition. Either edge of the external input can be configured to trigger the capture.

chip select: For some blocks of the TMS370 memory map, the most significant bits of the address are pre-decoded to activate chip-select signals. These chip-select signals allow the TMS370 to access external addresses with a minimum of external logic and to perform memory bank selection under software control.

circular buffer: A variable length area in the PACT module dual-port RAM that stores the value of a PACT timer when a capture request is made. As new values are captured, they are put into successive locations in the buffer. When the buffer is full, the oldest captures are replaced with newer captures.

code conversion utility: A software program that translates a COFF object file into one of several standard ASCII hexadecimal formats suitable for loading into an EPROM programmer.

COFF: *Common Object File Format.* An implementation of the object file format of the same name developed by AT&T. The TMS370 compiler, assembler, and linker use and generate COFF files.

command/definition area: A variable length area in the PACT module dual-port RAM that is used to define the actions taken by the PACT module.

comment: A source statement (or portion of a source statement) that documents or improves the readability of a source file. Comments are not compiled, assembled, or linked; they have no effect on the object file.

CDT370: A low-cost code-development tool that is external to the target system and provides direct control over the TMS370 processor that is on the target system.

compare register: A timer 1 or timer 2 register that contains a value that is compared to the counter value. The compare function triggers when the counter matches the contents of the compare register.

constant: A value that does not change during execution.

CPU: An 8-bit register-oriented processor with a status register, program counter, and stack pointer. The TMS370 CPU uses the register file, accessed in one bus cycle, as working registers.

D

debugger: A window-oriented software interface that helps you to debug '370 programs running on an emulator, CDT370, or design kit.

dedicated capture registers: An area in the PACT module dual-port RAM that stores the value of the default timer at the time of a specified edge on one of the PACT input capture pins. Unlike the circular buffer, the location of the dedicated capture register does not change.

default timer (also hardware timer): A 20-bit hardware counter in the PACT module that is incremented by the PACT prescaled clock.

design kit: A low-cost tool that allows you to analyze the hardware and software capabilities of the TMS370 family.

dual-port RAM: An area in RAM that can be read from and written to by both the TMS370 CPU and the PACT module.

E

edge detection: A type of circuitry that senses an active pulse transition on a given timer input and provides appropriate output transitions to the rest of the module. The active transition can be configured to be low-to-high or high-to-low.

EEPROM: *Electrically erasable programmable read only memory.* Memory that has the capability to be programmed and erased under direct program control.

EPROM: *Erasable programmable read only memory.* Memory that has the capability to be programmed under direct program control.

extended addressing mode: An addressing mode with a 16-bit range.

G

gang programmer: An interactive, menu-driven system that provides programming support for on-chip EEPROM or EPROM of the TMS370 microcontrollers in a production environment.

general addressing mode: An addressing mode with an 8-bit range.

H

halt mode: An operating mode that reduces operating power by stopping the internal clock, which stops processing in all the modules. This is the lowest-power mode in which all register contents are preserved.

I

idle mode: An operating mode in which the CPU stops processing and waits for the next interrupt. It is not a low-power mode.

immediate operand: An operand whose actual constant value is specified in the instruction and placed after the opcode in the machine code.

index: An 8-bit unsigned number added to a base address to give a final address.

instruction: The basic unit of programming that causes the execution of one operation; consists of an opcode and operands, along with optional labels and comments.

interrupt: A signal to the CPU that stops the flow of a program and forces the CPU to execute instructions at an address corresponding to the source of the interrupt. When the interrupt is finished, the CPU resumes execution at the point where it was interrupted.

isosynchronous communications mode: An SCI mode in which data transmission is synchronized by a clock signal (SCICLK) common to both the sender and receiver. The format is identical to the asynchronous mode and consists of a start bit, data bits, an optional parity bit, and a stop bit.

L

label: A symbol that begins in column 1 of a source statement and corresponds to the address of that statement.

linker: A software tool that combines object files to form an object module that can be allocated into system memory and executed by the devices.

low-power mode: An operating mode that reduces operating power by reducing or stopping the activity of various modules. There are two low-power modes: halt and standby.

LSB: *Least significant bit.*

LSbyte: *Least significant byte.*

M

machine code: The actual bytes read by the CPU during an instruction execution; usually read by a programmer as hexadecimal bytes.

MC pin: *Mode control pin.* The pin that determines the operating mode of the TMS370 device, depending on the voltage applied to the pin. Twelve volts on the MC pin after reset places the processor in the write protection override (WPO) mode.

memory map: A description of the addresses of the various sections and features of the TMS370 processor. The map depends on the operating mode.

microcomputer mode with external expansion: An operating mode in which the address, control, and data memory extend off-chip to access external memory or peripherals.

microcomputer single-chip mode: An operating mode in which the device uses only on-chip memory.

microcontroller programmer: An interactive, menu-driven system that provides a method of programming TMS370 family devices and EPROMs directly or through an XDS.

microprocessor mode with internal program memory: An operating mode in which the on-chip program memory is available to the processor.

microprocessor mode without internal program memory: An operating mode in which the on-chip program memory is not available to the processor. The processor must have external memory.

J

mini-SCI: The mini-UART function available in the PACT module.

mnemonic: A symbol that represents the opcode part of an assembly language instruction.

MSB: *Most significant bit.*

MSbyte: *Most significant byte.*

multiprocessor communications: An SCI format option that enables one processor to efficiently send blocks of data to other processors on the same serial link.

N

nested interrupts: The ability of an interrupt to suspend the service routine of a prior interrupt. Nested interrupts are implemented in TMS370 devices by executing an interrupt service routine that uses the EINT, EINTL, or EINTH instructions to set the global interrupt enable bits in the status register.

nonmaskable interrupt (NMI): An interrupt that causes the processor to execute the NMI routine. On TMS370 devices, INT1 can be configured as an NMI.

NRZ (nonreturn to zero) format: A communication format in which the inactive state is a logic state.

O

offset: A signed value that is added to the base operand to give the final address.

opcode: *Operation code.* The first byte of the machine code that describes to the CPU the type of operation and combination of operands. Some TMS370 instructions use 16-bit opcodes.

operand: The part of an instruction that tells the programmer where the CPU will fetch or store data.

P

PACT: *Programmable acquisition and control timer module.* A timer coprocessor module for the TMS370 microcontroller family.

peripheral file (PF): The 128 or 256 bytes of memory, starting at 1000h, that contain the registers that control the on-board peripherals and system configuration.

peripheral file frames: A set of sixteen contiguous peripheral file registers, usually related by function.

PPM: *Pulse-position modulation.* A serial signal in which the information is contained in the frequency of a signal with a constant pulse width. By using the timer compare features, a TMS370 device can output a PPM signal with a constant duty cycle without any program intervention.

prescaler: A circuit that slows the rate of a clocking source to the counter. The timer 1 prescaler can slow the clocking source by a factor of 4, 16, 64, or 256.

privilege mode: A mode immediately following reset in which the program can alter the privileged registers and bits. Once the privilege mode is disabled, these registers cannot be changed before another reset. This mode does not affect the EEPROM or the watchdog registers.

program counter (PC): A CPU register that identifies the current statement in the program.

prototyping device: A device used before mask-ROM devices are available that has identical functions, pinout, size, and timings to those of the actual device. Programmable memory such as EEPROM or EPROM is used in place of the masked ROM.

pulse accumulation: A timer 1 or timer 2 mode that keeps a cumulative count of SYSCLK pulses gated by the T1EVT or T2EVT signal.

PWM: *Pulse-width modulation.* A serial signal in which the information is contained in the width of a pulse of a constant frequency signal. By using the timer compare features, a TMS370 device can output a PWM signal with a constant duty cycle without any program intervention.

R

RAM: *Random access memory.*

ratiometric conversion: An analog-to-digital conversion in which the conversion value is a ratio of the V_{REF} source to the analog input. As V_{REF} is increased, the input voltage needed to give a certain conversion value changes; however, all conversion values keep the same relationship to V_{REF}.

J

referred timer: The timer that a PACT command uses for time comparisons. This is the last timer defined in the PACT command/definition area before the command was encountered, or if no timer has been defined, it is the least significant 16 bits of the hardware timer.

register file (RF): The first 128 or 256 bytes of memory that can be accessed by the majority of the instructions.

relative addressing mode: An operating mode in which operands and code produce an absolute address at some distance from the current location.

$\overline{\text{RESET}}$ pin: A pin that, when held low, starts hardware initialization and insures an orderly software startup. If the MC pin is low when the $\overline{\text{RESET}}$ signal returns high, then the processor enters the microcomputer mode. If the MC pin is high when the $\overline{\text{RESET}}$ signal returns high, then it enters the microprocessor mode.

S

serial communications interface (SCI): A built-in serial interface that can be programmed to be asynchronous or isosynchronous. Many timing, data format, and protocol factors are programmable and controlled by the SCI module in operation.

serial peripheral interface (SPI): A built-in serial interface that facilitates communication between networked master and slave CPUs. As in the SCI, the SPI is set up by software; from then on, the CPU takes no part in timing, data format, or protocol.

signed integer: A number system used to express positive and negative integers.

stack: The part of the register file used as last-in, first-out memory for temporary variable storage. The stack is used during interrupts and calls to store the current program status. The area occupied by the stack is determined by the stack pointer and by the application program.

stack pointer (SP): An 8-bit CPU register that points to the last entry or top of the stack. The SP is automatically incremented before data is pushed onto the stack and decremented after data is popped from the stack.

standby mode: A power reduction mode in which the CPU stops processing, but the on-chip oscillator remains active. Timers remain active and can cause the CPU to exit the standby mode.

status register (ST): A CPU register that monitors the operation of the instructions and contains the global interrupt enable bits.

symbolic debugging: The ability of a software tool to retain symbolic information so that it can be used by a debugging tool such as an XDS/22, a design kit, or a CDT370.

symbol table: A portion of a COFF object file that contains information about the symbols that are defined and used by the file.

T

time slots: The internal cycles in which the PACT module can make a 32-bit access to the dual-port RAM. Each command or definition requires one, two, or three time slots. The number of time slots available is a function of the PACT prescaled clock and the frequency of access to the dual-port RAM by the CPU.

U

unsigned integer: A number system used to express positive integers.

V

virtual timer: An entry in the PACT command/definition area that creates an independent time base that is incremented by the PACT prescaled clock and cleared upon reaching a maximum value that is set by this definition.

W

WAIT pin: The pin that allows an external device to cause the processor to wait an indefinite number of clock cycles. When the wait line is released, the processor resynchronizes with the rising edge of the clockout signal and continues with the program.

wait states, automatic: Extra clock cycles inserted automatically on every external memory access to accommodate peripherals or expansion memory with slower access time than the TMS370 processor. These wait states are governed by two control bits: PF AUTOWAIT (SCCR0.5) and AUTOWAIT DISABLE (SCCR1.4).

watchdog timer: A timer option that can be programmed to generate an interrupt when it times out. This function serves as a hardware monitor over the software to prevent a "lost" program and is available in both the timer 1 and PACT modules. If timer 1 does not need a watchdog, this timer can be used as a general-purpose timer.

write protect override (WPO): The only mode in which a TMS370 device can modify the on-board EEPROM. The WPO mode is entered when external circuitry applies 12 volts to the MC pin after the device has been reset into one of its normal operating modes.

X

XDS/22: A code-development tool that is external to the target system and provides direct control over the TMS370 processor that is on the target system.

Index

B

C

D

E

J

L

M

N

O

P

R

S

T

U

V

W

X

Z

SPNS014B