



 **TEXAS  
INSTRUMENTS**

---

# **TMS370 Family**

**Data  
Manual**

*Data Manual*

**TMS370 Family**

**1993**

**1993**

---

***8-Bit Microcontroller Family***

## 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.

- 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.

- ❑ 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.



**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. . .

<b>If you want to . . .</b>	<b>Do this. . .</b>
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: <b>(800) 477-8924</b>
Ask questions about product operation or report suspected problems	Call the Microcontroller Hotline: <b>(713) 274-2370</b> <b>FAX: (713) 274-4203</b>
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

<b>3</b>	<b>CPU and Memory Organization</b>	<b>3-1</b>
	<i>The TMS370 has a register-to-register architecture. This chapter describes the CPU registers and memory organization.</i>	
3.1	CPU/Register File Interaction	3-2
3.2	CPU Registers	3-4
3.2.1	Stack Pointer (SP)	3-4
3.2.2	Status Register (ST)	3-5
3.2.3	Program Counter (PC)	3-7
3.3	Memory Map	3-8
3.3.1	Register File	3-9
3.3.2	Peripheral File	3-11
3.3.3	Data EEPROM Modules	3-12
3.3.4	Program Memory	3-13
3.4	Memory Operating Modes	3-15
3.4.1	Microcomputer Single-Chip Mode	3-16
3.4.2	Microcomputer Mode With External Expansion (All Devices With Memory Expansion and Internal Program Memory)	3-18
3.4.3	Microprocessor Mode Without Internal Memory (Memory Expansion Devices Only)	3-22
3.4.4	Microprocessor Mode With Internal Program Memory (Memory Expansion Devices Only)	3-23
3.4.5	Memory Mode Summary	3-25
<b>4</b>	<b>System and Digital I/O Configuration</b>	<b>4-1</b>
	<i>Discusses the system and I/O configuration. Features and options are described, as well as the registers that control the configuration.</i>	
4.1	System Configuration	4-2
4.1.1	Privilege Mode	4-2
4.1.2	Oscillator Fault	4-4
4.1.3	Automatic Wait States	4-4
4.2	Low-Power and Idle Modes	4-6
4.2.1	Standby Mode	4-7
4.2.2	Halt Mode	4-8
4.2.3	Using Interrupts to Exit From the Halt Mode	4-8
4.2.4	Oscillator Power Bit	4-10
4.3	System Control Registers	4-11
4.3.1	System Control and Configuration Register 0 (SCCR0)	4-11
4.3.2	System Control and Configuration Register 1 (SCCR1)	4-13
4.3.3	System Control and Configuration Register 2 (SCCR2)	4-14
4.4	Digital I/O Configuration	4-16
4.4.1	Function A and Function B Signal Definitions	4-19
4.4.2	Microprocessor Mode	4-22
4.4.3	Microcomputer Mode	4-22

7.6	Interrupts	7-16
7.7	Watchdog Timer	7-17
7.7.1	Standard Watchdog Configuration (OTP/Reprogrammable EPROM Devices)	7-18
7.7.2	Hard Watchdog Configuration (Mask-ROM Devices Only)	7-20
7.7.3	Simple Counter Configuration (Mask-ROM Devices Only)	7-22
7.7.4	Summary of Watchdog Options	7-23
7.8	Low-Power Modes	7-24
7.8.1	Halt Mode	7-24
7.8.2	Standby Mode	7-24
7.9	Timer 1 Control Registers	7-25
7.9.1	Timer 1 Control Register 1 (T1CTL1)	7-27
7.9.2	Timer 1 Control Register 2 (T1CTL2)	7-29
7.9.3	Timer 1 Control Register 3 (T1CTL3)	7-31
7.9.4	Timer 1 Control Register 4 (T1CTL4)	7-33
7.9.5	Timer 1 Port Control Registers (T1PC1 and T1PC2)	7-35
7.9.6	Timer 1 Interrupt Priority Control Register (T1PRI)	7-38
<b>8</b>	<b>Timer 2 Module</b>	<b>8-1</b>
	<i>Discusses the architecture and programming of the timer 2 module.</i>	
8.1	Timer 2 Overview	8-2
8.1.1	Physical Description	8-2
8.1.2	Operating Modes	8-3
8.1.3	Control Registers	8-4
8.2	Timer 2 Components	8-5
8.2.1	16-Bit Resettable Counter	8-5
8.2.2	Compare Register	8-5
8.2.3	Capture Register (Dual Capture Mode Only)	8-6
8.2.4	Capture/Compare Register	8-7
8.3	Operating Modes	8-8
8.3.1	Dual Compare Mode	8-8
8.3.2	Dual Capture Mode	8-9
8.4	Edge-Detection Circuitry	8-11
8.5	Clock Sources	8-12
8.5.1	Event Counter Mode	8-12
8.5.2	Pulse Accumulator Mode	8-13
8.6	Interrupts	8-13
8.7	Low-Power Modes	8-14
8.8	Timer 2 Control Registers	8-15
8.8.1	Timer 2 Control Register 1 (T2CTL1)	8-17
8.8.2	Timer 2 Control Register 2 (T2CTL2)	8-18
8.8.3	Timer 2 Control Register 3 (T2CTL3)	8-20
8.8.4	Timer 2 Port Control Registers (T2PC1 and T2PC2)	8-22
8.8.5	Timer 2 Interrupt Priority Control Register (T2PRI)	8-24

10.5	Interrupts .....	10-9
10.6	Clock Sources .....	10-10
10.7	Initialization Upon Reset .....	10-11
10.8	SPI Example .....	10-12
10.9	SPI Control Registers .....	10-13
10.9.1	SPI Configuration Control Register (SPICCR) .....	10-14
10.9.2	SPI Operation Control Register (SPICTL) .....	10-16
10.9.3	Serial Input Buffer (SPIBUF) .....	10-17
10.9.4	Serial Data Register (SPIDAT) .....	10-17
10.9.5	SPI Port Control Registers (SPIPC1 and SPIPC2) .....	10-18
10.9.6	SPI Interrupt Priority Control Register (SPIPRI) .....	10-20
<b>11</b>	<b>Analog-to-Digital Converter Module .....</b>	<b>11-1</b>
	<i>Discusses the architecture and programming of the analog-to-digital converter module.</i>	
11.1	Analog-to-Digital Converter (A/D) Overview .....	11-2
11.1.1	Physical Description .....	11-2
11.1.2	Control Registers .....	11-3
11.2	A/D Operation .....	11-4
11.2.1	Input/Output Pins .....	11-4
11.2.2	Sampling Time .....	11-4
11.2.3	A/D Conversion .....	11-5
11.2.4	Interrupts .....	11-5
11.2.5	Programming Considerations .....	11-6
11.3	A/D Example Program .....	11-7
11.4	A/D Control Registers .....	11-9
11.4.1	Analog Control Register (ADCTL) .....	11-10
11.4.2	Analog Status and Interrupt Register (ADSTAT) .....	11-12
11.4.3	Analog Conversion Data Register (ADDATA) .....	11-12
11.4.4	Analog Port E Data Input Register (ADIN) .....	11-13
11.4.5	Analog Port E Input Enable Register (ADENA) .....	11-13
11.4.6	Analog Interrupt Priority Register (ADPRI) .....	11-14
<b>12</b>	<b>Programmable Acquisition and Control Timer (PACT) .....</b>	<b>12-1</b>
	<i>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.</i>	
12.1	PACT Overview .....	12-2
12.1.1	Physical Description .....	12-2
12.1.2	Control Registers .....	12-4
12.2	PACT Operation .....	12-5
12.2.1	Hardware Pins .....	12-5
12.2.2	Memory Organization .....	12-5
12.2.3	Time Base .....	12-6
12.2.4	Command/Definition File Format .....	12-7
12.2.5	Available Time Slots .....	12-7

<b>13</b>	<b>Assembly Language Instruction Set</b> .....	<b>13-1</b>
	<i>Summarizes the TMS370 family assembly language instruction set and provides individual instruction descriptions.</i>	
13.1	Instruction Operation .....	13-2
13.2	Symbol Definitions .....	13-3
13.3	Addressing Modes .....	13-4
13.3.1	General Addressing Modes .....	13-5
13.3.2	Extended Addressing Modes .....	13-11
13.3.3	Additional Addressing Modes .....	13-16
13.3.4	Status Register .....	13-16
13.4	Instruction Set Overview .....	13-17
13.5	Instruction Set Descriptions .....	13-26
<b>14</b>	<b>Design Aids</b> .....	<b>14-1</b>
	<i>To assist you in system development, contains sample TMS370 applications.</i>	
14.1	Microcomputer Interface Example .....	14-2
14.1.1	Read Cycle Timing .....	14-5
14.1.2	Write Cycle Timing .....	14-9
14.1.3	Design Options .....	14-10
14.1.4	Bank Switching Examples .....	14-12
14.2	Programming With the TMS370 Family .....	14-14
14.3	Serial Communications .....	14-17
14.3.1	SPI Port Interfacing .....	14-17
14.3.2	SCI Port Interfacing .....	14-18
14.4	Analog-to-Digital Converter .....	14-21
14.5	PACT Module .....	14-22
14.5.1	Time After Event Example .....	14-22
14.5.2	Double Event Compare Command Example .....	14-26
14.5.3	PACT SCI Example .....	14-28
14.6	Sample Routines .....	14-30
14.6.1	T1PWM Pin Set-Up .....	14-30
14.6.2	Clear RAM .....	14-31
14.6.3	RAM Self Test .....	14-32
14.6.4	ROM Checksum .....	14-33
14.6.5	Binary-to-BCD Conversion .....	14-34
14.6.6	BCD-to-Binary Conversion .....	14-35
14.6.7	BCD String Addition .....	14-36
14.6.8	Fast Parity .....	14-37
14.6.9	Bubble Sort .....	14-38
14.6.10	Table Search .....	14-39
14.6.11	16-by-16 (32-Bit) Multiplication .....	14-40
14.6.12	Keyboard Scan .....	14-41
14.6.13	Divide 1 .....	14-43
14.6.14	Divide 2 .....	14-44

<b>17</b>	<b>Customer Information</b>	<b>17-1</b>
	<i>Describes mask-ROM prototyping, TMS370 physical characteristics, and parts ordering.</i>	
17.1	Mask-ROM Prototype and Production Flow	17-2
17.2	Mechanical Package Information	17-6
17.3	TMS370 Family Numbering and Symbol Conventions	17-15
17.3.1	Production Device Prefix Designators	17-15
17.3.2	Support Device Prefix Designators	17-16
17.3.3	Device Numbering Conventions	17-16
17.3.4	Device Symbols	17-17
17.4	Ordering Information for Development Support Tools	17-19
17.4.1	TMS370 Macro Assembler, Linker, C Compiler, and Utilities	17-19
17.4.2	TMS370 Design Kit	17-19
17.4.3	TMS370 Microcontroller Programmer	17-19
17.4.4	TMS370 XDS Systems	17-20
17.4.5	TMS370 Compact Development Tool	17-20
17.4.6	XDS Upgrade (Available in Europe Only)	17-20
17.4.7	XDS Target Connectors	17-20
<b>A</b>	<b>Differences Between a TMS370CxxxA Device and a TMS370Cxxx Device</b>	<b>A-1</b>
	<i>Points out the differences between the TMS370CxxxA devices that are described in this manual and the TMS370Cxxx devices.</i>	
A.1	Watchdog Options	A-2
A.2	Timer 1 Control Register 2 (T1CTL2) Bits	A-3
A.3	System Control and Configuration Register 2 (SCCR2) Bits	A-4
A.4	V <sub>CC1</sub> and V <sub>CC2</sub> Pins	A-4
A.5	Low-Power and Idle Modes	A-4
A.6	Electrical Specifications	A-5
A.6.1	Differences for TMS370Cx5x Devices	A-5
A.6.2	Differences in SCI and SPI Specifications	A-6
A.7	Summary of Differences	A-7
<b>B</b>	<b>Peripheral File Memory Map</b>	<b>B-1</b>
	<i>Summarizes the peripheral file and control bit information.</i>	
B.1	Peripheral File Frame 1: System Configuration Registers	B-2
B.2	Peripheral File Frame 2: Digital Port Control Registers	B-3
B.3	Peripheral File Frame 3: SPI Control Registers	B-4
B.4	Peripheral File Frame 4: Timer 1 Control Registers	B-5
B.5	Peripheral File Frame 4: PACT Control Registers	B-6
B.6	Peripheral File Frame 5: SCI Control Registers	B-7
B.7	Peripheral File Frame 6: Timer 2 Control Registers	B-8
B.8	Peripheral File Frame 7: A/D Converter Control Registers	B-9

# Figures

---

---

1-1	TMS370Cx1x Block Diagram .....	1-11
1-2	TMS370Cx2x Block Diagram .....	1-12
1-3	TMS370Cx3x Block Diagram .....	1-13
1-4	TMS370Cx4x Block Diagram .....	1-14
1-5	TMS370Cx5x Block Diagram .....	1-15
2-1	Pinouts for TMS370Cx1x .....	2-2
2-2	Pinouts for TMS370Cx2x .....	2-4
2-3	Pinout for TMS370Cx3x .....	2-6
2-4	Pinouts for TMS370Cx4x .....	2-8
2-5	Pinouts for TMS370Cx5x .....	2-10
3-1	Programmer's Model .....	3-3
3-2	Stack Example .....	3-4
3-3	Program Counter After Reset .....	3-7
3-4	TMS370 Memory Map .....	3-8
3-5	Register File Addresses .....	3-9
3-6	Microcomputer Single-Chip Mode .....	3-17
3-7	Microcomputer Mode With Function A Expansion .....	3-20
3-8	Microcomputer Mode With Function B Expansion .....	3-21
3-9	Microprocessor Mode Without Internal Memory .....	3-22
3-10	Microprocessor Mode With Internal Program Memory .....	3-24
3-11	Memory Operating Modes .....	3-26
4-1	Correct Method to Enter Halt Mode .....	4-9
4-2	Improper Method to Enter Halt Mode .....	4-10
4-3	Peripheral File Frame 2: Digital Port Control Registers .....	4-17
4-4	Port Control Register Operation .....	4-18
4-5	System Interface Example .....	4-23
5-1	Interrupt Control .....	5-3
5-2	Peripheral File Frame 1: External Interrupt Control Registers .....	5-7
5-3	Interrupt 1 Block Diagram .....	5-9
5-4	Interrupts 2 and 3 Block Diagram .....	5-9
5-5	Typical Reset Circuit .....	5-17
5-6	Typical Reset Circuit Using a Supply Voltage Supervisor .....	5-18
6-1	Write Protection Bits in an EEPROM Array .....	6-3
6-2	EEPROM Programming Example .....	6-7
6-3	EEPROM Programming Operation .....	6-13

13–10	Relative Indexed Addressing Mode .....	13-14
13–11	Absolute Indirect Addressing Mode .....	13-14
13–12	Relative Indirect Addressing Mode .....	13-15
13–13	Absolute Offset Indirect Addressing Mode .....	13-15
13–14	Relative Offset Indirect Addressing Mode .....	13-16
13–15	Status Register (ST) .....	13-16
14–1	Microcomputer Interface Example .....	14-2
14–2	Valid Address-to-Data Read Timing .....	14-5
14–3	Chip-Select Low-to-Data Read Timing .....	14-6
14–4	Chip-Select High-to-Next Data Bus Drive Timing .....	14-7
14–5	Read Data Hold After Chip-Select High Timing .....	14-8
14–6	Write Data Set-Up Timing .....	14-9
14–7	Write Data Hold After Chip-Select High .....	14-10
14–8	Master/Slave SPI Interface Example .....	14-17
14–9	SCI/RS–232 Interface Example .....	14-18
14–10	Autobaud Waveform .....	14-19
14–11	A/D Converter Sample Applications .....	14-21
14–12	Time After Event—Example Waveforms .....	14-22
14–13	Double Event Compare—Example Waveforms .....	14-26
14–14	Keyboard Scan Values .....	14-41
15–1	Software Development Flow .....	15-3
15–2	Linker Output Generation .....	15-6
15–3	The Basic Debugger Display .....	15-9
15–4	The BTT Set-Up Dialog Box .....	15-12
15–5	The Dialog Box for Defining Conditions .....	15-13
15–6	An of the INSPECT Window .....	15-13
15–7	Typical XDS System Configuration .....	15-16
15–8	CDT370 Configuration .....	15-19
15–9	Typical TMS370 Microcontroller Programmer Configuration .....	15-22
15–10	Typical TMS370 Gang Programmer Board .....	15-24
16–1	Measurement Points for Timings .....	16-2
16–2	Recommended Crystal/Clock Connections .....	16-4
16–3	Typical Output Load Circuit .....	16-4
16–4	Switching Time Measurement Points .....	16-5
16–5	External Clock Timing .....	16-8
16–6	CLKOUT Timing .....	16-8
16–7	External Clock Timing .....	16-11
16–8	CLKOUT Timing .....	16-11
16–9	External Clock Timing .....	16-14
16–10	CLKOUT Timing .....	16-14
16–11	External Clock Timing .....	16-17
16–12	CLKOUT Timing .....	16-17
16–13	External Clock Timing .....	16-21
16–14	External Read Timing .....	16-23



# Tables

---

---

1-1	TMS370 Family Categories and Their Corresponding Devices	1-3
1-2	TMS370 Family Architecture Summary	1-9
2-1	TMS370Cx1x Pin Descriptions	2-3
2-2	TMS370Cx2x Pin Descriptions	2-5
2-3	TMS370Cx3x Pin Descriptions	2-7
2-4	TMS370Cx4x Pin Descriptions	2-9
2-5	TMS370Cx5x Pin Descriptions	2-11
3-1	Peripheral File Address Map	3-11
3-2	Vector Address Map	3-13
3-3	Memory Modes Available	3-15
3-4	Operating Mode Summary	3-25
4-1	Peripheral File Frame 1: System Configuration and Control Registers	4-2
4-2	Privilege-Mode Configuration Bits	4-3
4-3	Wait-State Control Bits	4-5
4-4	Powerdown/Idle Control Bits	4-6
4-5	Digital I/O Pins by Device	4-16
4-6	Port Configuration Registers Set-Up	4-19
5-1	Interrupts and Reset Vectors	5-2
5-2	Module Interrupt Priority	5-3
5-3	Interrupt Vector Sources	5-4
5-4	Reset Sources	5-15
5-5	Control-Bit States Following Reset	5-16
7-1	Timer 1 I/O Pin Definitions	7-3
7-2	Timer 1 and Watchdog Timer Memory Map	7-4
7-3	Timer 1 Compare Values: (CLKIN = 20 MHz)	7-7
7-4	Counter Overflow Rates	7-14
7-5	Watchdog Option Summary	7-23
7-6	Peripheral File Frame 4: Timer 1 Control Registers	7-26
8-1	Timer 2 I/O Pin Definitions	8-3
8-2	Timer 2 Memory Map	8-4
8-3	Timer 2 Compare Values: (CLKIN = 20 MHz)	8-6
8-4	Peripheral File Frame 6: Timer 2 Control Registers	8-16
9-1	SCI Memory Map	9-5
9-2	Programming the Data Format Using SCICCR	9-6
9-3	Asynchronous Baud Register Values for Common SCI Bit Rates	9-15

---

16-24	External Clocking Requirements .....	16-21
16-25	Switching Characteristics and Timing Requirements for External Read and Write ....	16-22
16-26	SCI Isosynchronous Mode Timing Characteristics and Requirements for Internal Clock .....	16-24
16-27	SCI Isosynchronous Mode Timing Characteristics and Requirements for External Clock .....	16-25
16-28	SPI Master External Timing Characteristics and Requirements .....	16-26
16-29	SPI Slave External Timing Characteristics and Requirements .....	16-27
16-30	Recommended Operating Conditions .....	16-28
16-31	A/D Converter Operating Characteristics Over Full Range of Operating Conditions ..	16-28
16-32	Analog Timing Requirements .....	16-29
17-1	Package Types .....	17-6
17-2	Symbolization Designators .....	17-17
A-1	Watchdog Option Summary for TMS370CxxxA Devices .....	A-2
A-2	Switching Characteristics and Timing Requirements for External Read and Write .....	A-5
A-3	I <sub>OL</sub> (Low-Level Output Current) .....	A-6
A-4	SCI Isosynchronous Mode Timing Characteristics for Internal Clock .....	A-6
A-5	SPI Slave External Timing Characteristics and Requirements .....	A-6
A-6	Differences Between TMS370CxxxA devices and TMS370Cxxx Devices .....	A-7
E-1	TMS370 Family Opcode/Instruction Map .....	E-2
F-1	TMS370 Family Instruction/Opcod Set .....	F-2
F-2	Possible Bus Cycles .....	F-5
F-3	Internal Cycles .....	F-5
F-4	Bus Activity Table .....	F-6





## 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.

## 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

## 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 re-programmed 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

## **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 (Continued)

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

<sup>¶</sup> In ROM-less (microprocessor) mode, all address, data, and control lines are fixed as their function.

<sup>#</sup> For OTP (PLCC) availability information, contact your local TI sales office or distributor.

<sup>\*</sup> System evaluator (reprogrammable EPROM)—for use in prototype environment only.

<sup>□</sup> Refer to Table 17–1, page 17-6 for package type acronyms.

<sup>◇</sup> 64-pin devices do not support off-chip memory expansion.

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

Figure 1–2. TMS370Cx2x Block Diagram

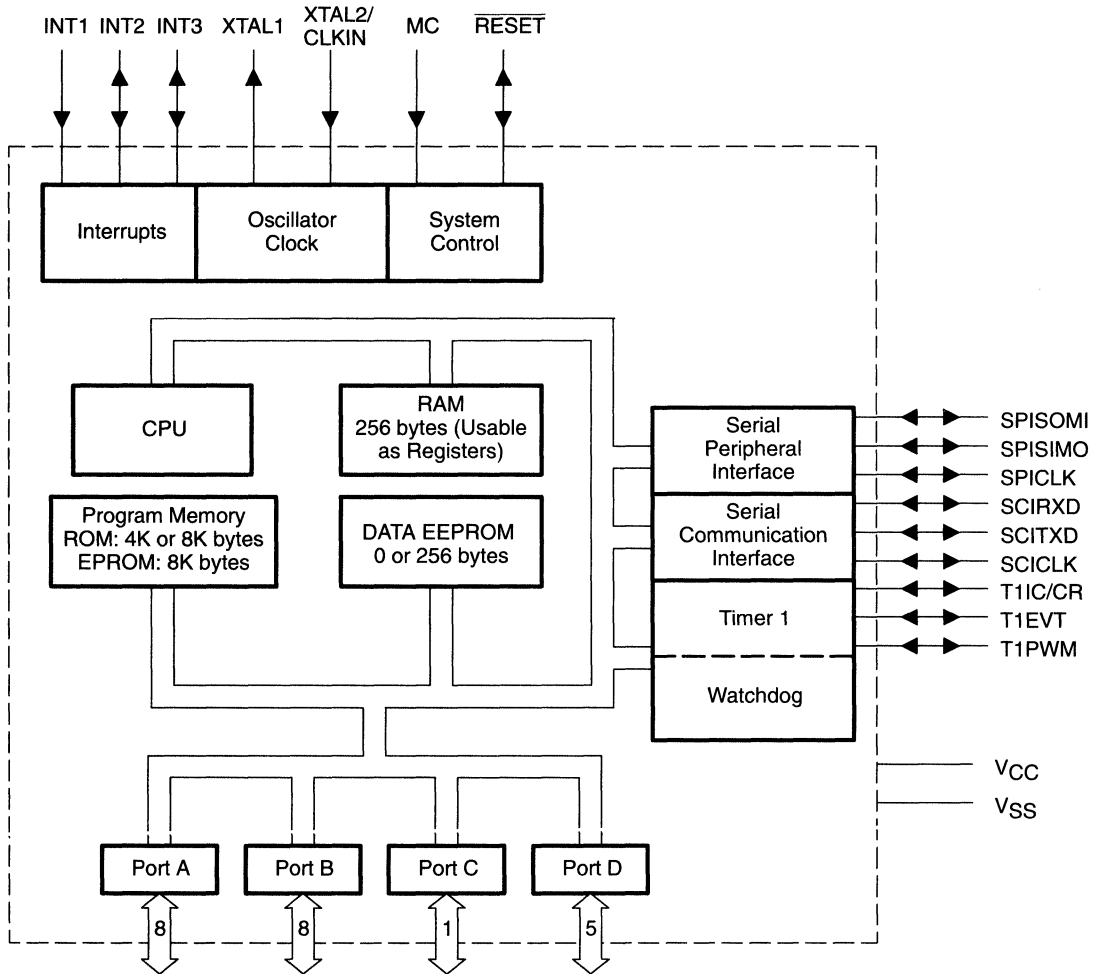
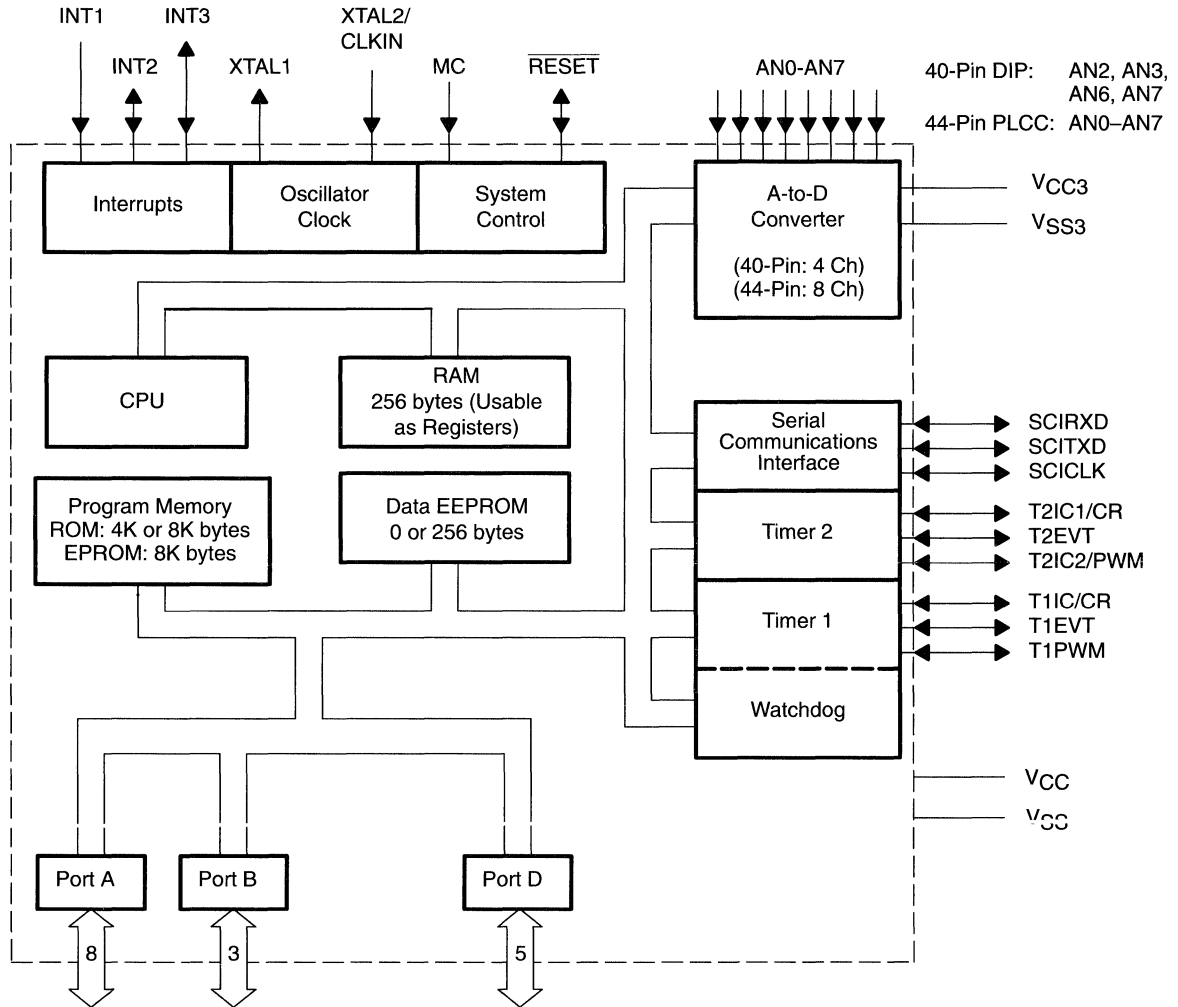


Figure 1–4 is a block diagram of the TMS370Cx4x devices, showing the major functional blocks.

Figure 1–4. TMS370Cx4x Block Diagram





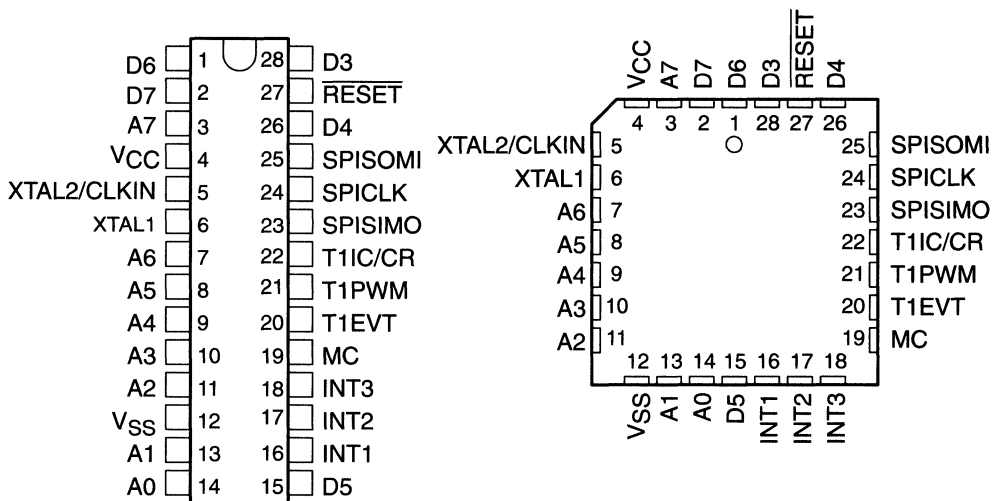


## 2.1 TMS370Cx1x Pinouts and Pin Descriptions

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



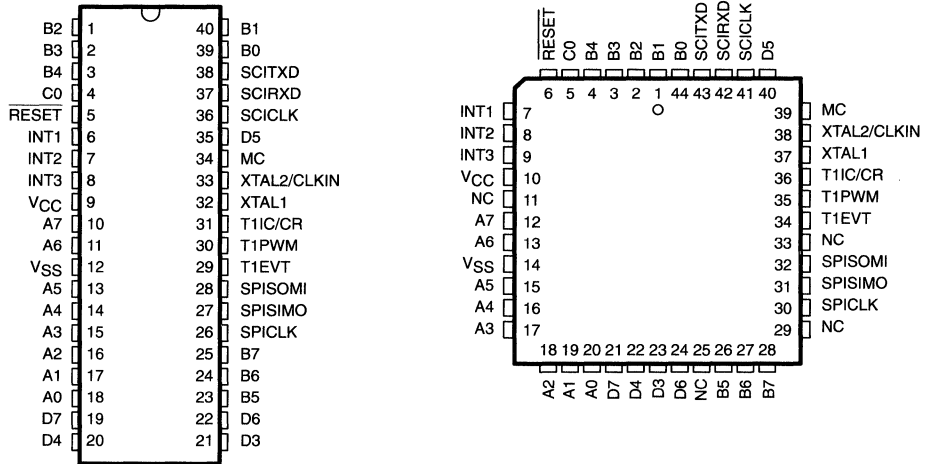
- A. 28-Pin PDIP (N)
- B. 28-Pin CDIP (JD)

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

## 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



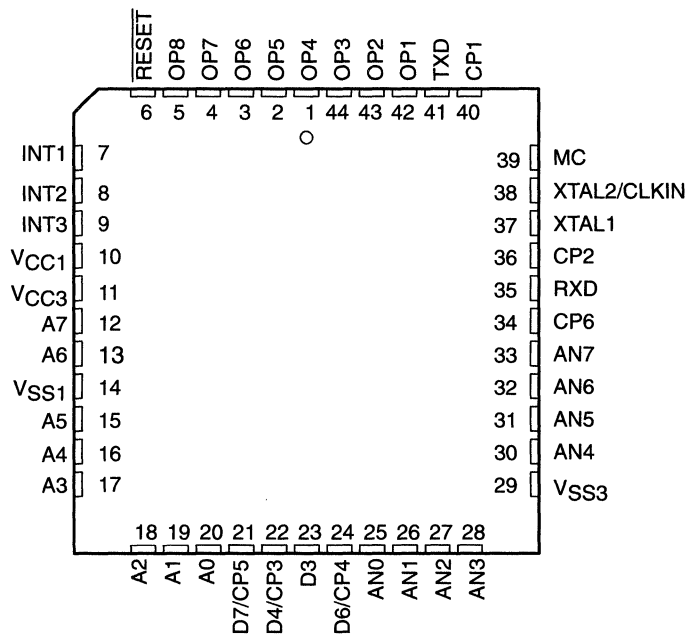
- A. 40-Pin PDIP (N)
- B. 40-Pin CDIP (JD)
- C. 40-Pin SDIP (N2)
- D. 40-Pin CSDIP (JC)

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

## 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)

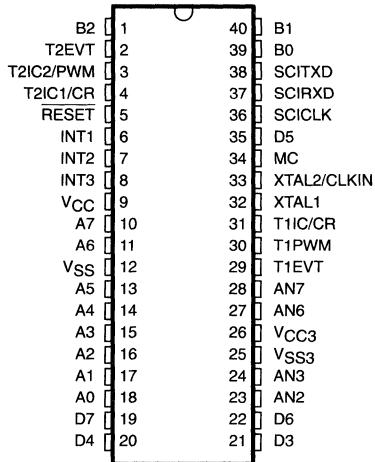


## 2.4 TMS370Cx4x Pinouts and Pin Descriptions

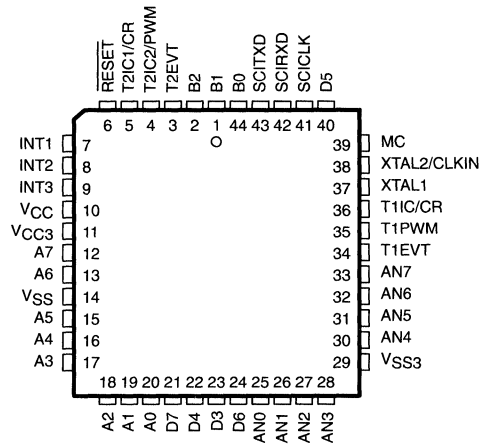
2

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)

## 2.5 TMS370Cx5x Pinouts and Pin Descriptions

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

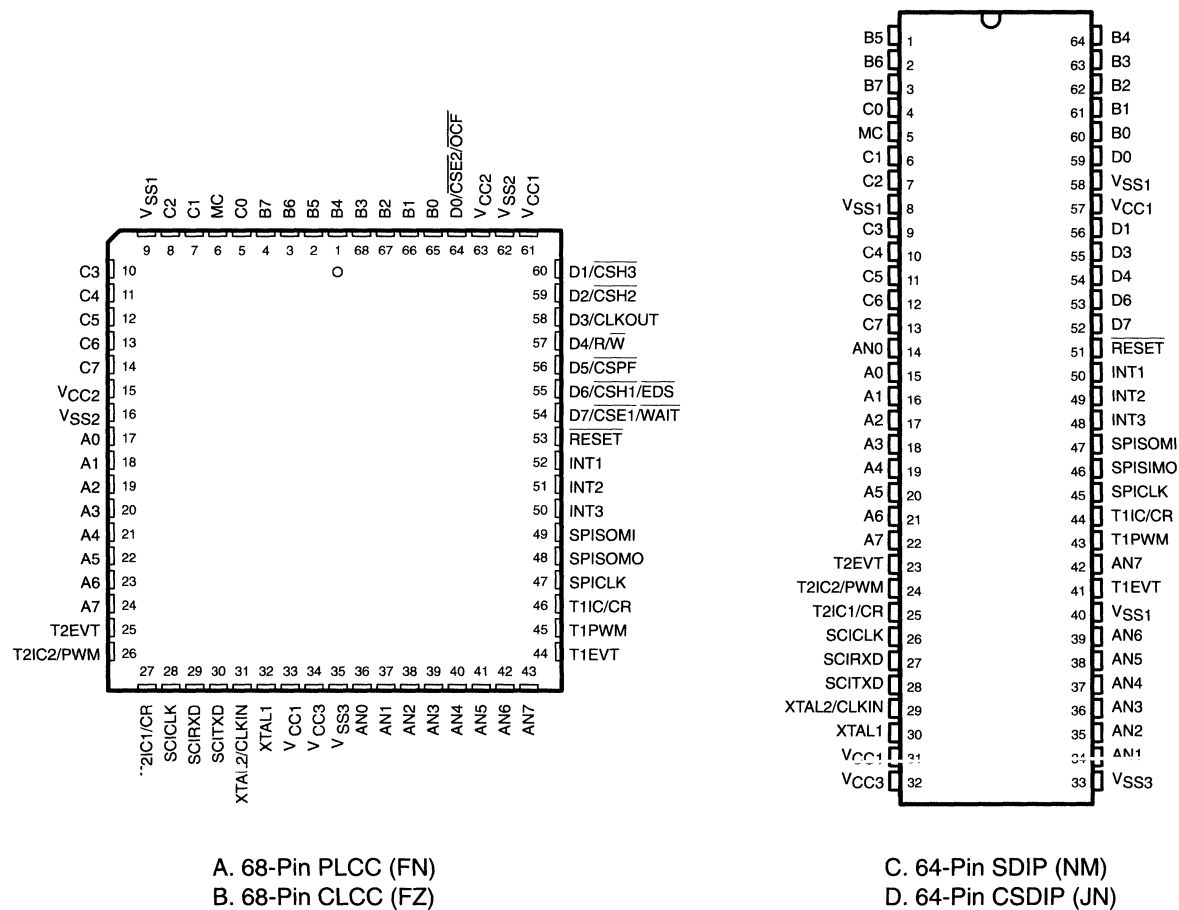


Table 2–5. TMS370Cx5x Pin Descriptions (Continued)

Name	Pin		DIP (64)	PLCC (68)	I/O	Description
	Alternate Function†	Function A 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	$\overline{\text{CSE2}}$	$\overline{\text{OCF}}$	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	$\overline{\text{CSH3}}$		56	60	I/O	I/O pin/A: chip select half output 3 goes low during memory accesses to 8000h–FFFFh
D2	$\overline{\text{CSH2}}$		—	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	$\overline{\text{R/W}}$	$\overline{\text{R/W}}$	54	57	I/O	I/O pin/A, B: read/write output pin
D5	$\overline{\text{CSPF}}$		—	56	I/O	I/O pin/A: chip select peripheral output for peripheral file goes low during memory accesses to 10C0h–10FFh
D6	$\overline{\text{CSH1}}$	$\overline{\text{EDS}}$	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	$\overline{\text{CSE1}}$	$\overline{\text{WAIT}}$	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	SPIO1		47	49	I/O	SPI slave output pin, master input pin/general-purpose bidirectional pin
SPISIMO	SPIO2		46	48	I/O	SPI slave input pin, master output pin/general-purpose bidirectional pin
SPICLK	SPIO3		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.





### 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.

## 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.

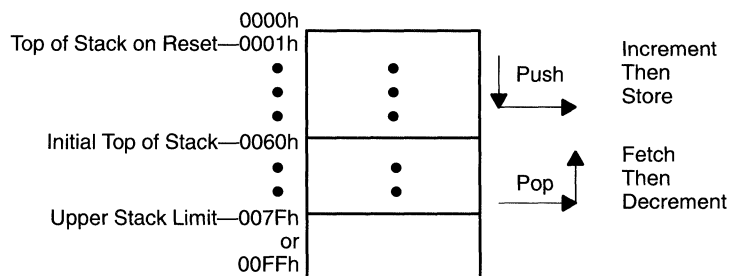
The stack pointer (SP) is an 8-bit CPU register that points to the last entry or top of the stack. The SP is automatically incremented *before* data is pushed onto the stack and decremented *after* data is popped from the stack.

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.

**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.



### 3.3 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

0000h 00FFh 0100h	256-Byte RAM (Register File/Stack)
	RAM Expansion (On-Chip)†
1000h 10BFh 10C0h	Peripheral File
10FFh 1100h	Peripheral File Expansion
1EFFh 1F00h	Data EEPROM Expansion (On-Chip)
	256-Byte Data EEPROM
1FFFh 2000h	32K-Byte Program Memory Start or Microprocessor Mode Memory Expansion‡
4000h	16K-Byte Program Memory Start or Microprocessor Mode Memory Expansion
6000h	8K-Byte Program Memory Start or Microprocessor Mode Memory Expansion
7000h	4K-Byte Program Memory Start or Microprocessor Mode Memory Expansion
7800h	2K-Byte Program Memory Start
7F00h 7FFFh	Interrupt and Reset Vectors; Trap Vectors
8000h FFFFh	Memory Expansion/External Memory

† 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.

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.

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.

### 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.

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.

### 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 pre-coded chip-select outputs.

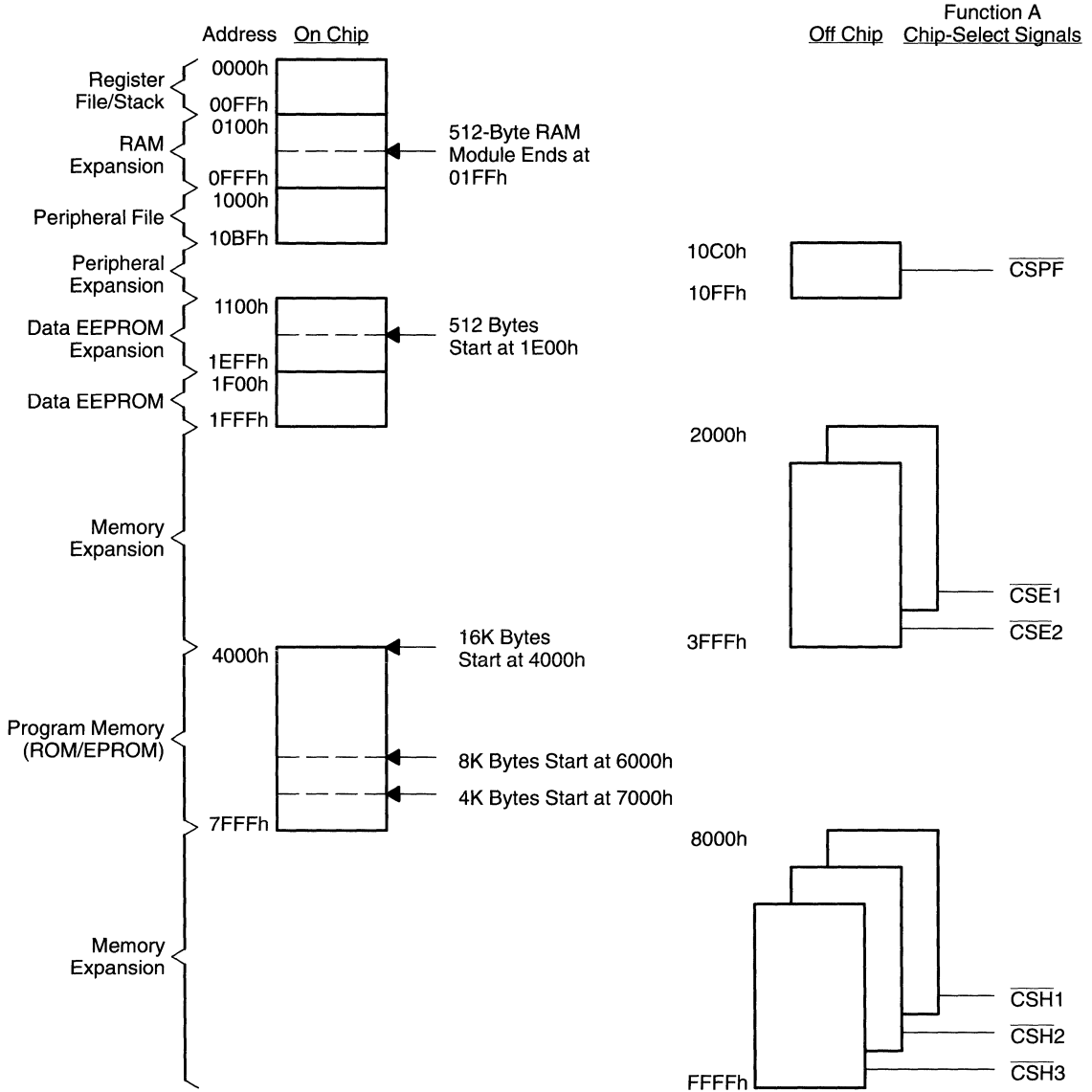
The port D outputs can be programmed on a pin-by-pin basis to provide direct memory/peripheral chip-select or chip-enable functions. Each port D pin can be individually set to function A, function B or general-purpose I/O.

#### 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  $\overline{CSE1}$  and  $\overline{CSE2}$ . Typically, an application that uses both  $\overline{CSE1}$  and  $\overline{CSE2}$  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  $\overline{CSH1}$ ,  $\overline{CSH2}$ , and  $\overline{CSH3}$  pins if they are enabled by the appropriate port control registers. The  $\overline{CSH1}$ ,  $\overline{CSH2}$ , and  $\overline{CSH3}$  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.

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.

### 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

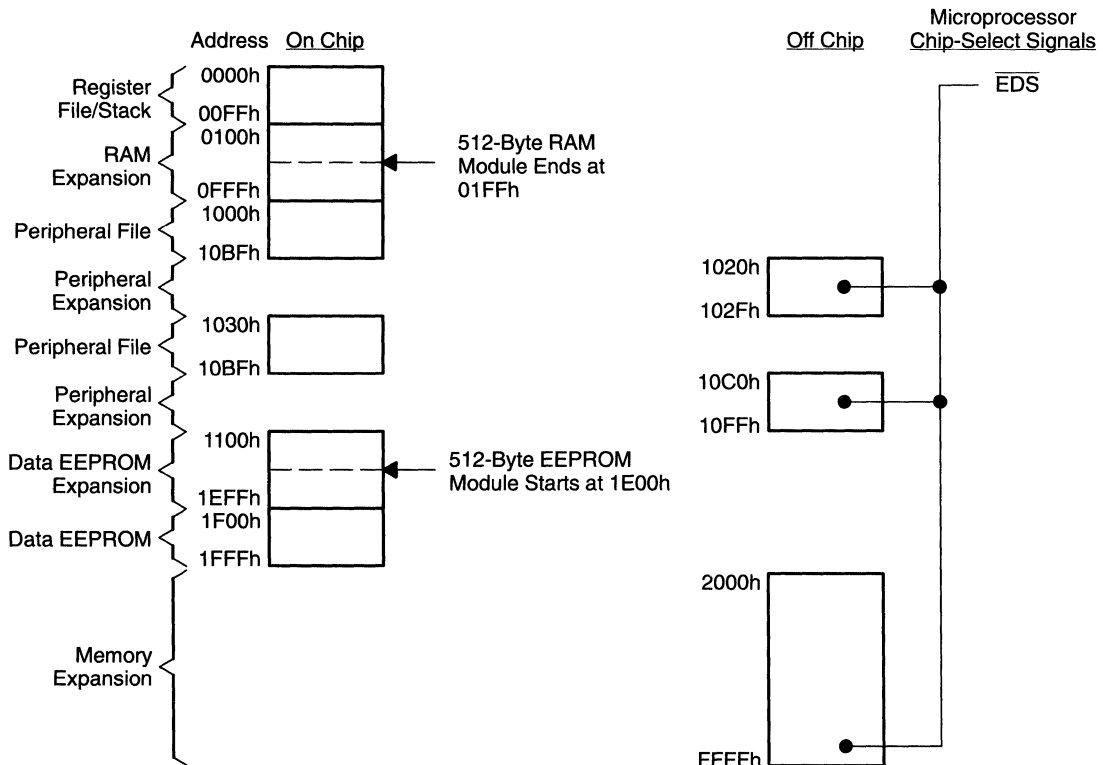
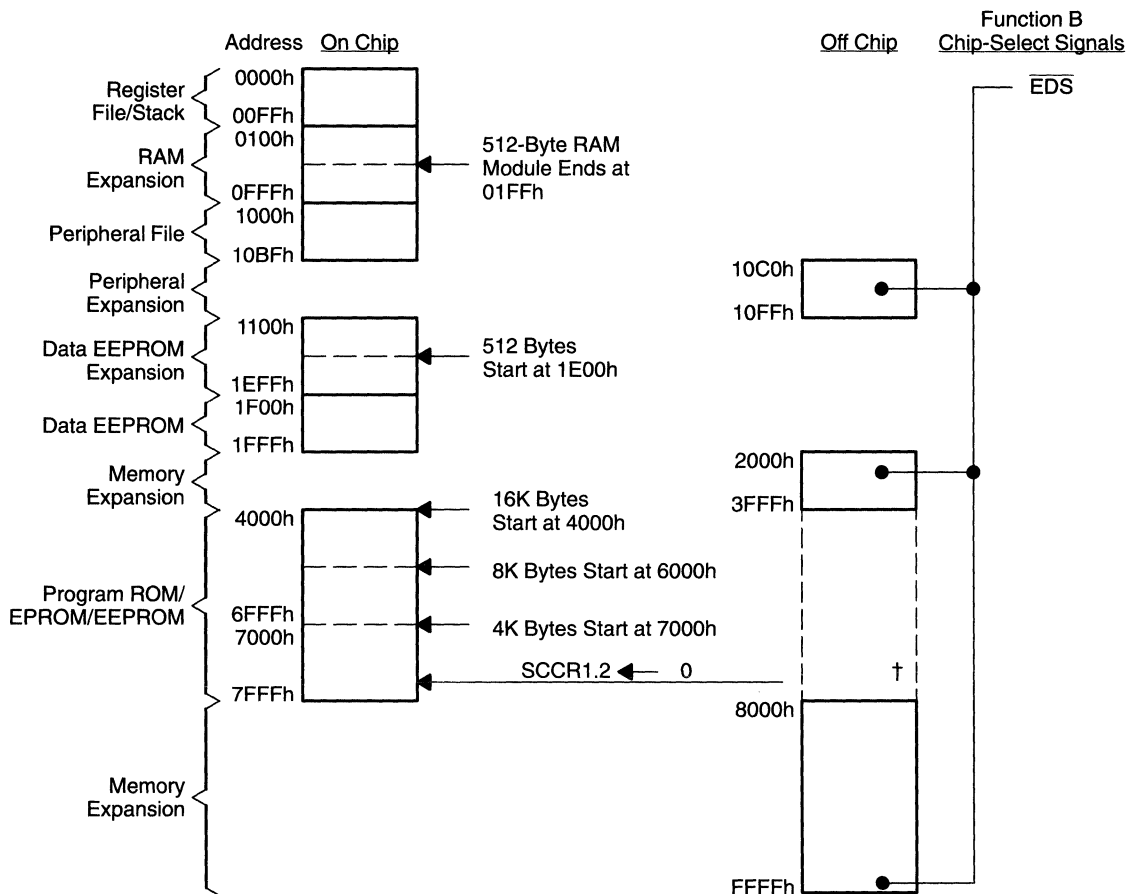




Figure 3–10. Microprocessor Mode With Internal Program Memory



† 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).

Figure 3–11. Memory Operating Modes

	Microcomputer Single-Chip Mode	Microcomputer With External Expansion	Microprocessor With Internal Program Memory	Microprocessor Mode
0000h	On Chip	On Chip	On Chip	On Chip
00FFh				
0100h	On-Chip Expansion	On-Chip Expansion	On-Chip Expansion	On-Chip Expansion
0FFFh				
1000h	On Chip	On Chip	On Chip‡	On Chip‡
10BFh				
10C0h	Not Available	External†	External	External
10FFh				
1100h	On-Chip Expansion	On-Chip Expansion	On-Chip Expansion	On-Chip Expansion
1EFFh				
1F00h	On Chip	On Chip	On Chip	On Chip
1FFFh				
2000h	On-Chip Expansion	External†	External	External
3FFFh				
4000h	On-Chip Expansion	On-Chip Expansion	On-Chip Expansion	External
6FFFh				
7000h	On Chip	On Chip	On Chip	External
7FFFh				
8000h	On-Chip Expansion	External†	External	External
FFFFh				

† Precoded chip-select outputs available on external expansion memory.

‡ 1020h–102Fh external.



## 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	PRIVILEGE 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.

### 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.

## 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		Mode Selected
PWRDWN/IDLE (SCCR2.6)	HALT/STANDBY (SCCR2.7)	
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.

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:

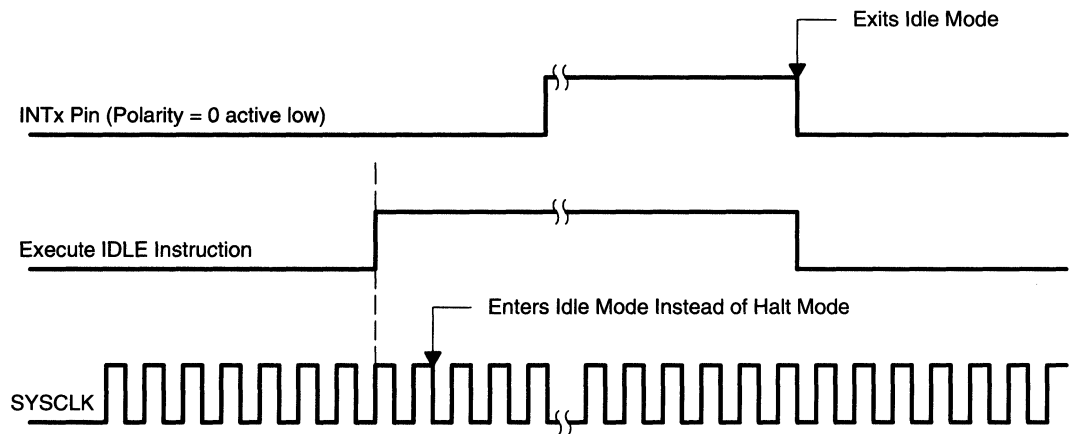
- Reset
- External Interrupt 1, 2, or 3 if enabled
- Low level on the SCIRXD pin if the SCI RX interrupt and receiver are enabled (described in Chapter 9)

## 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.

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*.



**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.

4

**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.

### 4.3.3 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

4

**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.

## 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.

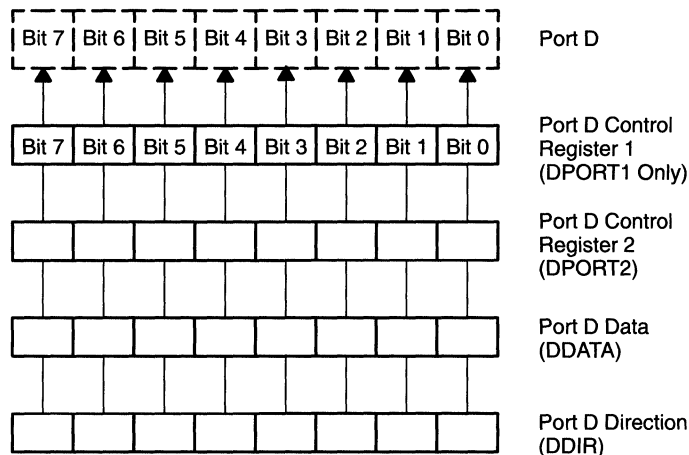
Table 4–5. Digital I/O Pins by Device

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 4–4. 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.

$\overline{\text{CSH3}}$	Chip-Select Half 3. This signal selects a third 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.
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.
4 $\overline{\text{EDS}}$	External Data Strobe. This signal goes low during external memory operations. The rising edge of $\overline{\text{EDS}}$ validates the read input data; the write data is available after the falling edge of $\overline{\text{EDS}}$ .
LOW ADDR/HI ADDR	External memory address bus. Output only.
$\text{R}/\overline{\text{W}}$	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.
$\overline{\text{OCF}}$	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).
$\overline{\text{WAIT}}$	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 $\overline{\text{EDS}}$ 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  $\overline{\text{CSH1}}$ ,  $\overline{\text{CSE1}}$ , and  $\overline{\text{CSPF}}$  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  $\text{R}/\overline{\text{W}}$ . The remaining bits of port D are used as I/O output.

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.







## 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 <sup>†</sup>	Vectors Total	Sources Total
TMS370Cx1x	4	6	13
TMS370Cx2x	4	8	16
TMS370Cx3x	4	23	25
TMS370Cx4x	4	9	22
TMS370Cx5x	4	10	23

<sup>†</sup> 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).

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

5

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
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.

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.

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.

## 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.

**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]

5

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.

### 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.

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	$\mu\text{P}/\mu\text{C}$ Mode	0	0	1
	MC PIN DATA	0	0	1
	COLD START	1	see Note 1	see Note 1
	OSC FLT FLAG	0	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	1	1	1
	TXRDY	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.

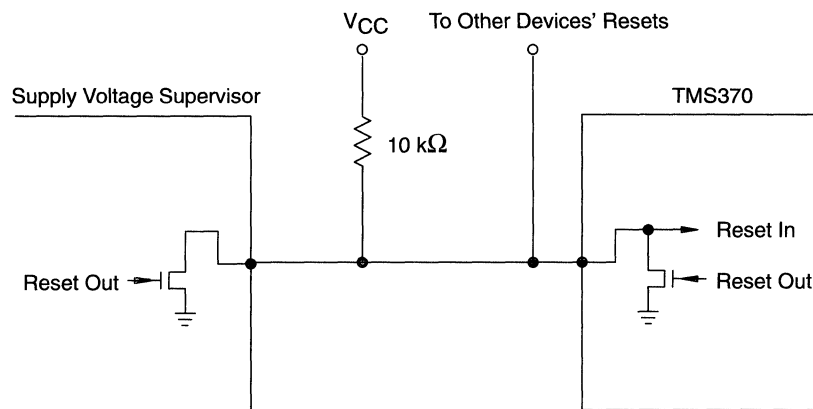
2) Refers to port control registers A, B, C, and D.

### 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  $V_{\text{CC}}$  is within operating range. Figure 5–5 shows a simple reset circuit that will hold  $\overline{\text{RESET}}$  low during power-up.



Figure 5–6. 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.



## 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.

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.

## 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.

## 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
            .
            .
            .

```

6

- 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 #2778 (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:
 
$$\text{loop count} = t_{W(PGM)B} / (\text{system cycle time} \times \text{delay loop cycle count})$$

$$\text{loop count} = 10 \text{ ms} / (200 \text{ ns} \times 18) = 10 \text{ ms} / 3.6 \mu\text{s} = 2778$$
- Note: Alternatively, a timer can be used for this delay.

## 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.**

## 6.4.2 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 ( $3X$  milliseconds duration).
- 10) Write to the EPCTL to clear the EXE and VPPS bits.



### 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.



## 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:

<b>System Requirements</b>	<b>Timer Resource</b>
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.

## 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.

7

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.

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:

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

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.

## 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.

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.

## 7.4 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 T1EDGE 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.



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:

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

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 <sup>16</sup>	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 <sup>18</sup>	0.524	0.262	0.105	0.052
1	0	1	2 <sup>20</sup>	2.10	1.05	0.419	0.210
1	1	0	2 <sup>22</sup>	8.39	4.19	1.68	0.839
1	1	1	2 <sup>24</sup>	33.6	16.8	6.71	3.355

† Time is given in seconds.

‡ Not applicable.

## 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.

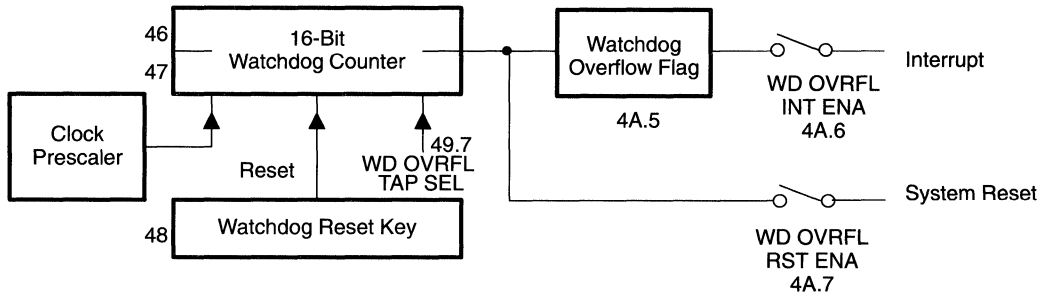
**Note:**

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.

## 7.7.1 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  $\overline{\text{WD OVRFL}}$  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.

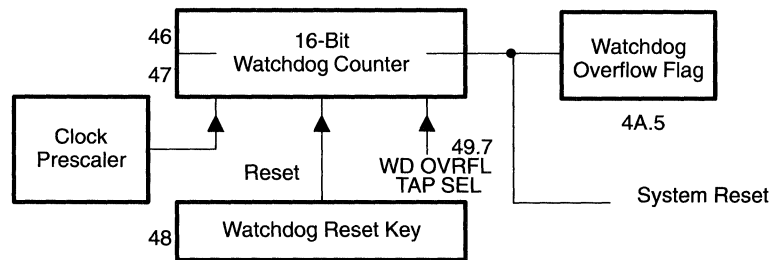
### 7.7.1.2 Nonwatchdog Mode

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



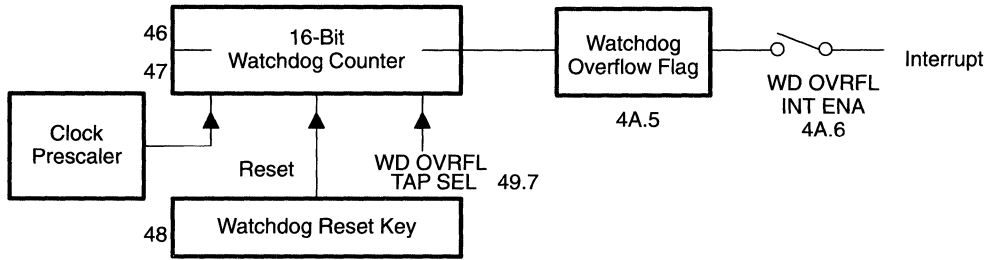
The hard watchdog provides additional system integrity. If the counter overflows or if the WD timer is reinitialized by an incorrect value, the hard watchdog generates a system reset, which pulls the  $\overline{\text{RESET}}$  pin low for eight system clock cycles. The required reinitialization frequency 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.

### 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.

## 7.8 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**

### 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.

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	T1 Counter MSbyte							Bit 8
T1CNTR	1041h	P041	T1 Counter LSbyte							Bit 0
T1C	1042h	P042	Compare Register MSbyte							Bit 8
T1C	1043h	P043	Compare Register LSbyte							Bit 0
T1CC	1044h	P044	Capture/Compare Register MSbyte							Bit 8
T1CC	1045h	P045	Capture/Compare Register LSbyte							Bit 0
WDCNTR	1046h	P046	Watchdog Counter MSbyte							Bit 8
WDCNTR	1047h	P047	Watchdog Counter LSbyte							Bit 0
WDRST	1048h	P048	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)
Dual Compare Mode										
T1CTL3	104Bh	P04B	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)
Dual Compare Mode										
T1CTL4	104Ch	P04C	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.

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



## 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

```

**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.

```

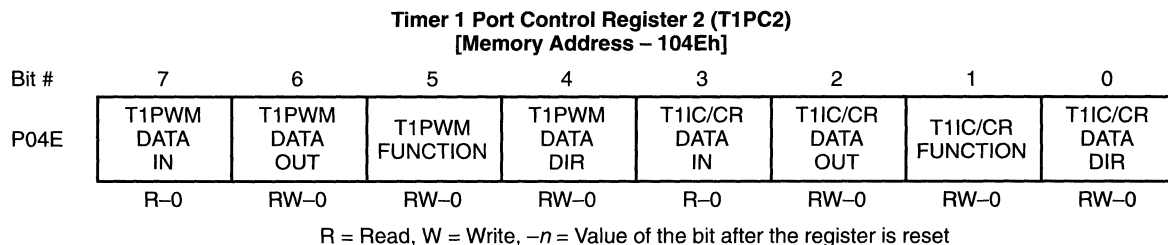
;Clearing the T1C1 INT FLAG
MOV    P04B,A
OR     #0E0h,A
AND    #0DFh,A
MOV    A,P04B
    
```

---

- 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.

### 7.9.5.2 Timer 1 Port Control Register 2 (T1PC2)

The T1PC2 register controls the I/O functions of the T1IC/CR and T1PWM pins.



**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.

### 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.



## 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

- ❑ **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.



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:

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

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)

8

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.

## 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

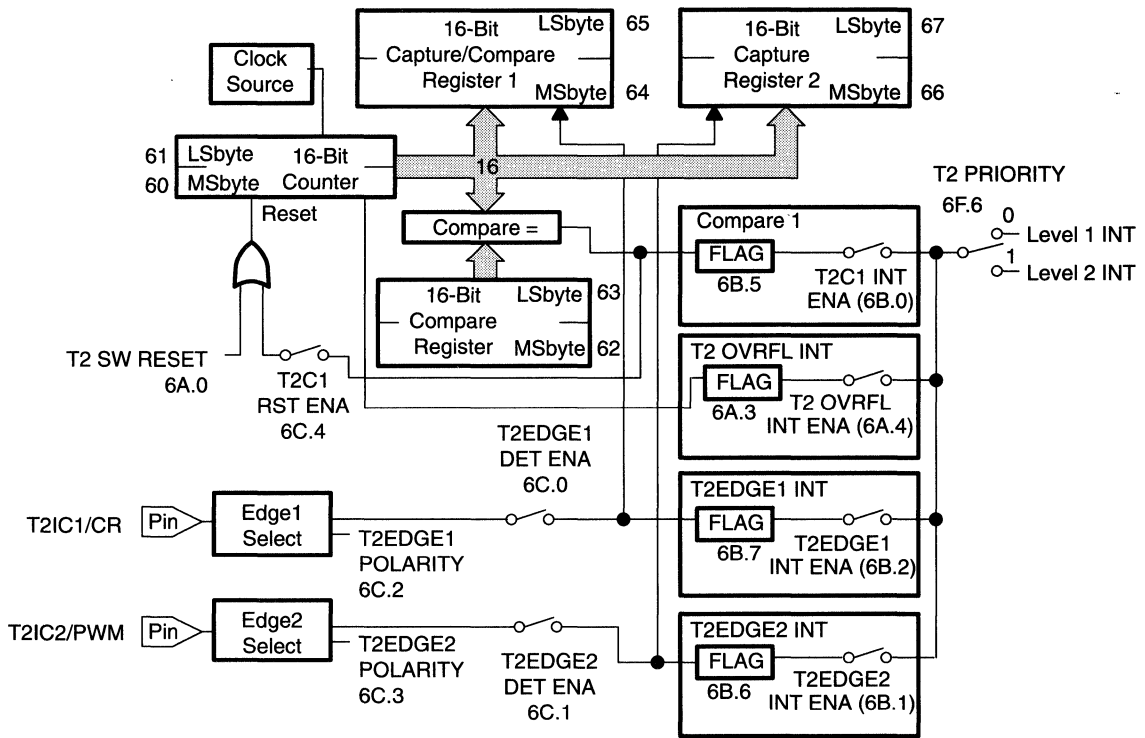
### 8.3.1 Dual Compare Mode

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 8–3. Dual Capture Mode



8

**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.

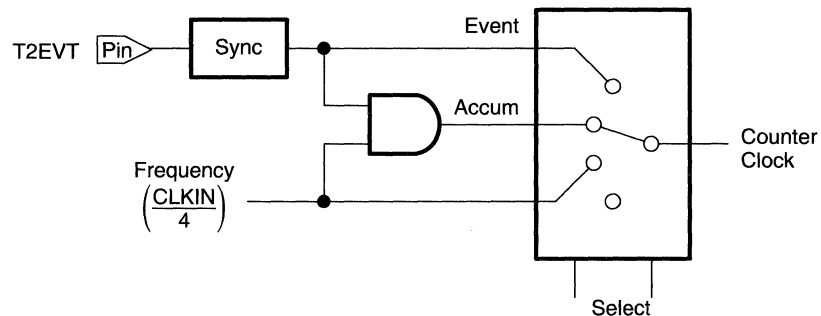
On the occurrence of a valid input on the T2IC1/CR or T2IC2/PWM pin, the current counter value is loaded into the 16-bit capture/compare register or 16-bit input capture register, respectively. In addition, the respective input capture INT FLAG bit is set, and a timer interrupt is generated if the respective INT ENA bit is set.

## 8.5 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:

$$\begin{aligned} \text{Maximum Counter Duration} &= 2^{16} \times \text{SYSCLK} \\ \text{Counter Resolution} &= \text{SYSCLK} \end{aligned}$$

where:  $\text{SYSCLK} = 4/\text{CLKIN}$

The external event frequency input to the module cannot exceed  $\text{CLKIN}/8$ . 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 ( $\text{CLKIN} = 20 \text{ MHz}$ ).

### 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.

## 8.7 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.

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)
T2CTL3	106Ch	P06C	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)
T2PC1	106Dh	P06D	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)
T2PC2	106Eh	P06E	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

### 8.8.2 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.

### 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.



### 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.

**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.



## 9.1 SCI Overview

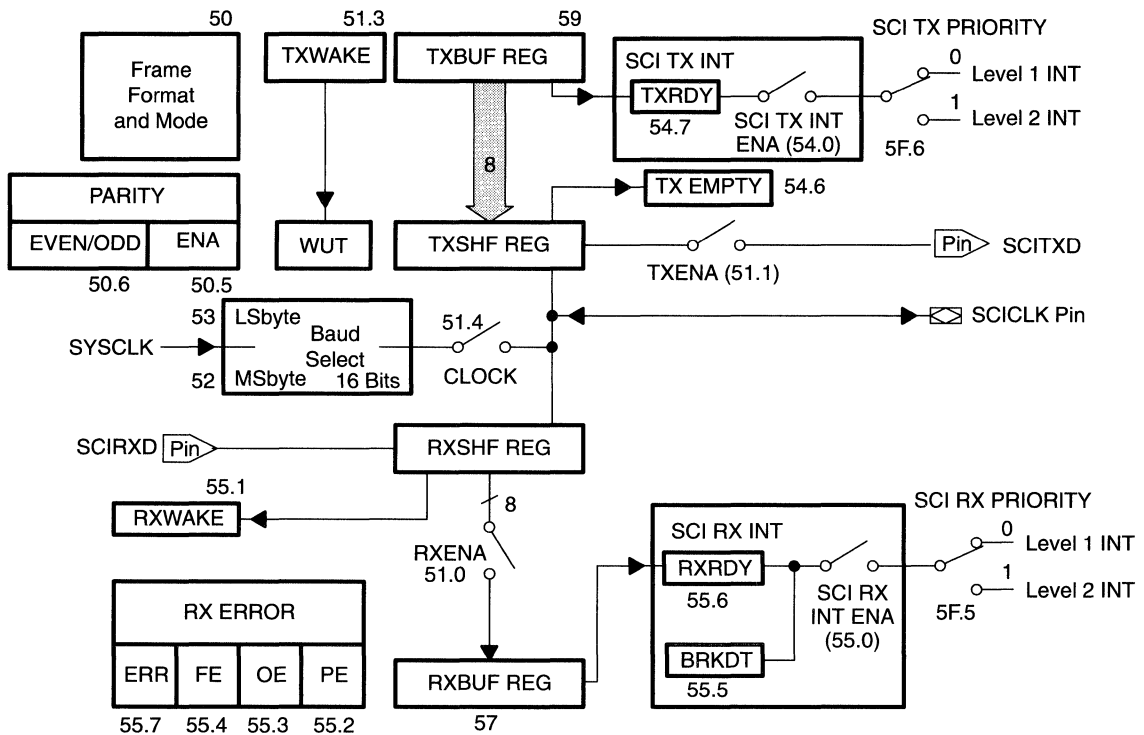
The serial communications interface (SCI) module is a programmable I/O port that facilitates digital communications between the TMS370 device and other asynchronous peripherals and uses the standard NRZ (nonreturn to zero) format. The SCI transmits and receives serial data, one bit at a time, at a programmable 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

Figure 9–1. SCI Block Diagram



## 9

### 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:

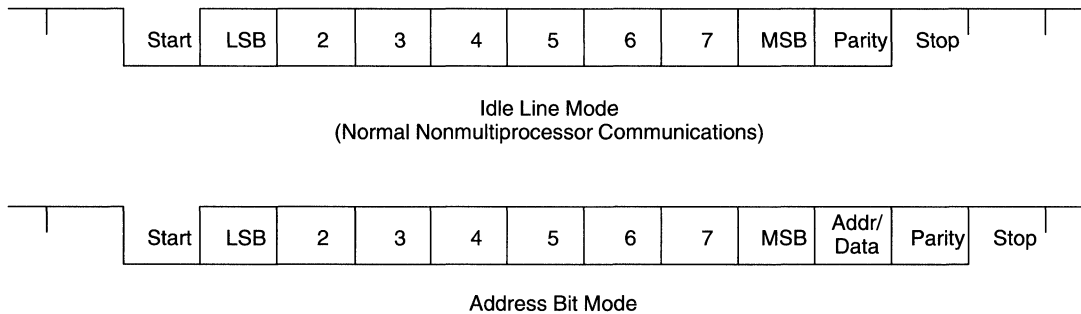
- 1 start bit,
- 1 to 8 data bits,
- An even/odd parity bit or no parity bit, and
- 1 or 2 stop bits.

## 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

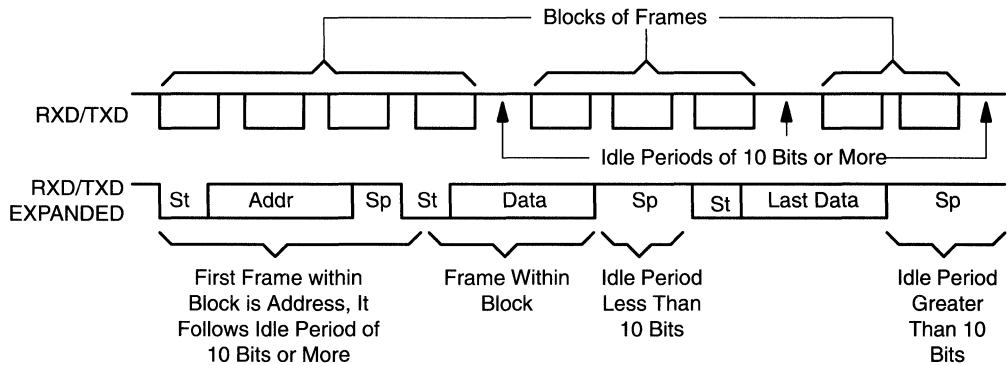
Bit Name	Designation	Function
SCI CHAR0–2	SCICCR.0–2	Select the character (data) length (1 to 8 bits). Refer to the bit listings on page 9-20 for additional information.
PARITY ENABLE	SCICCR.5	Enables the parity function if set to 1 or disables the parity function if cleared to 0.
EVEN/ODD PARITY	SCICCR.6	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.

- 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



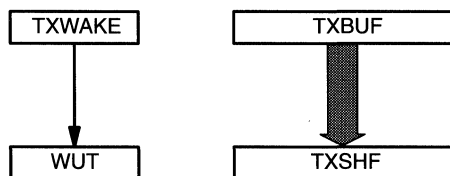
**Note:** In the figure, “St” = start and “Sp” = stop

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 from 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



## 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.

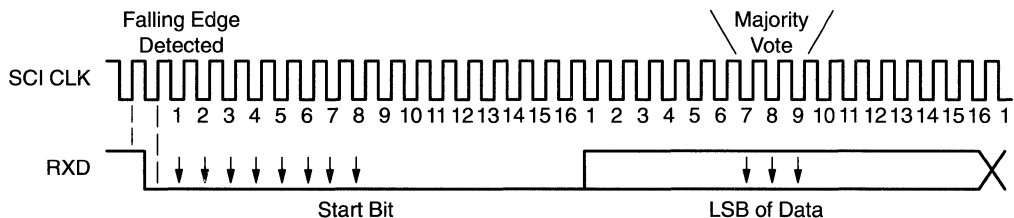
### 9.4.1 Asynchronous Communications Mode

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.

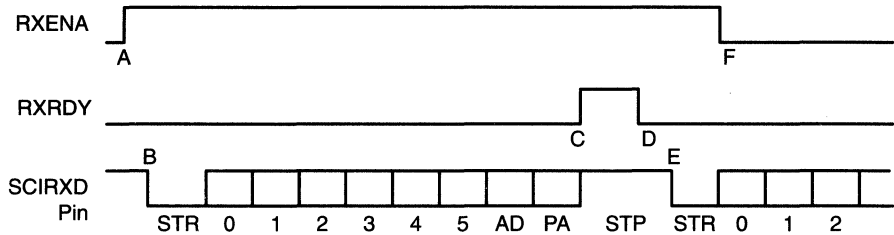
Figure 9–6. Asynchronous Communication Format



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.



Figure 9–8. SCI RX Signals in Communications Modes



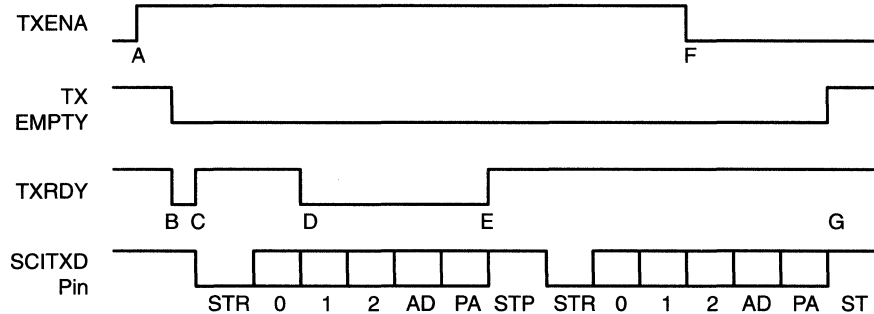
- A) RX ENA goes high to enable the receiver.
- B) Data arrives on the SCIRXD pin; start bit detected.
- C) RXRDY goes high to signal that a new character has been received; data is shifted to RXBUF; an interrupt is requested.
- D) The program reads the RXBUF register; RXRDY is automatically cleared.
- 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.

## 9.6 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 =  $\text{CLKIN} / [(\text{BAUD REG} + 1) \times 128]$   
 $\text{BAUD REG} = [\text{CLKIN} / (\text{Asynchronous Baud} \times 128)] - 1$
- Isosynchronous Baud =  $\text{CLKIN} / [(\text{BAUD REG} + 1)] \times 8]$   
 $\text{BAUD REG} = [\text{CLKIN} / (\text{Isosynchronous Baud} \times 8)] - 1$
- SCICLK frequency =  $\text{CLKIN} / [(\text{BAUD REG} + 1)] \times 8]$   
 $\text{BAUD REG} = [\text{CLKIN} / (\text{SCICLK frequency} \times 8)] - 1$

where

BAUD REG = The 16-bit value in the baud select registers.

Refer to Table 9–3.

## 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

```

```
                                ;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
         .
         .
         .
                                ;
RXEXIT   POP    P                ;Restore and exit
         RTI
```

### 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.

## 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.

		<b>SCI Control Register (SCICTL)</b> [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.

### 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:

$$\text{SCICLK frequency} = \text{CLKIN} / [(\text{BAUD REG} + 1) \times 8]$$

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:

$$\text{Asynchronous Baud} = \text{CLKIN} / [(\text{BAUD REG} + 1) \times 128]$$

$$\text{Isosynchronous Baud} = \text{CLKIN} / [(\text{BAUD REG} + 1) \times 8]$$

**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

### 9.8.5 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.

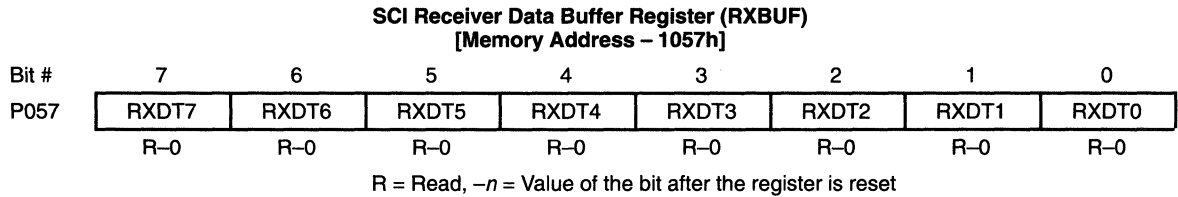
0 = No Overrun error detected.

1 = Overrun error detected.



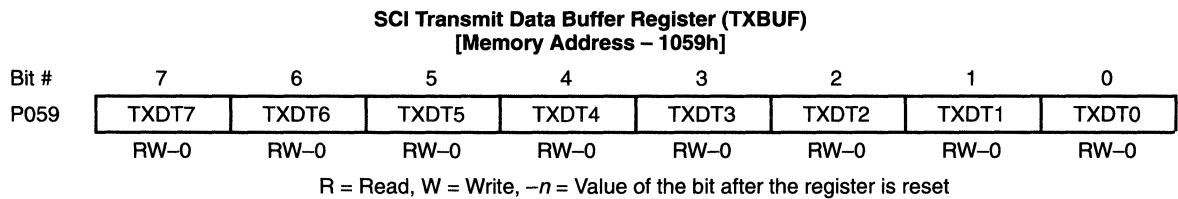
### 9.8.6 SCI Receiver Data Buffer Register (RXBUF)

The RXBUF register contains current data from the receiver shift register. RXBUF is cleared by a system reset.



### 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.



### 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.

### 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.



## 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

## 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.

## 10.3 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.

## 10.4 Data Format

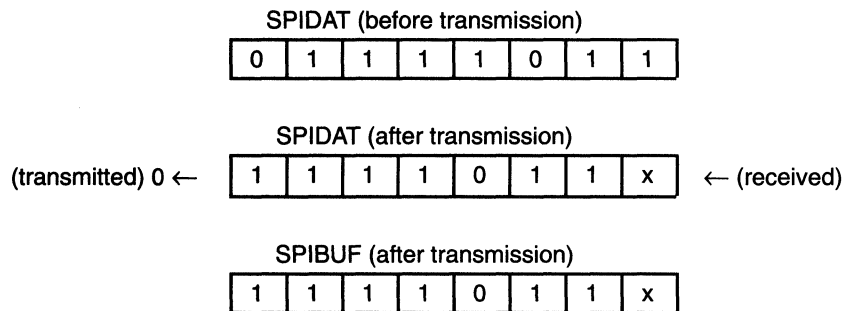
Three character-length bits (SPICCR.2–0) specify the number of bits in the data character (1–8 bits). This information directs the state control logic to count the number of bits received or transmitted to determine when a complete character has been processed.

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



## 10.6 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:

$$\text{SPI BAUD RATE} = \text{CLKIN} / (8 \times 2^b)$$

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

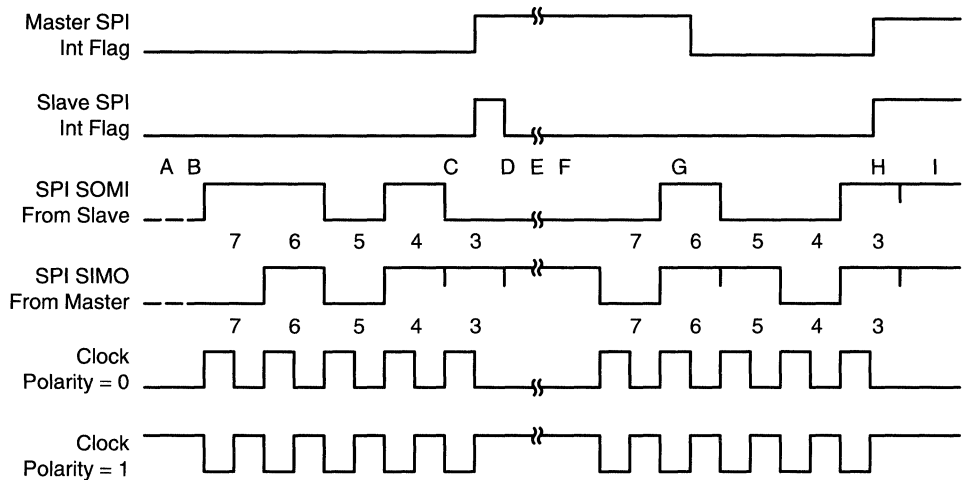
Crystal/Oscillator Frequency (MHz)						
SPI Value	Divide by	20 MHz	12 MHz	10 MHz	5 MHz	2 MHz
0	8	2500	1500	1250	625	250
1	16	1250	750	625	312.5	125
2	32	625	375	312.5	156.25	62.5
3	64	312.5	187.5	156.25	78.125	31.25
4	128	156.25	93.75	78.125	39.0625	15.625
5	256	78.125	46.875	39.0625	19.53125	7.8125
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.

## 10.8 SPI Example

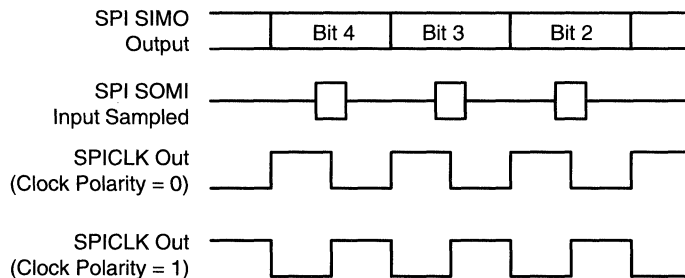
The following timing diagrams illustrate an SPI data transfer between two TMS370 devices using a character length of five bits. The lettered notes following the first diagram are keyed to the letter labels in the diagram.

### 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

### 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 <sup>†</sup>	SPI BIT RATE1 <sup>†</sup>	SPI BIT RATE0 <sup>†</sup>	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

<sup>†</sup> 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.

## 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.**

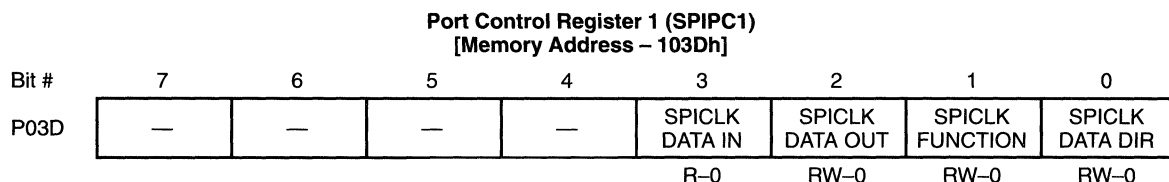
This bit is a read-only flag that the SPI hardware sets when a receive or transmit operation completes before the previous character has been read from the receive buffer. It indicates that the last received character has been overwritten and therefore has been lost. RECEIVER OVERRUN is cleared when the receiver buffer is read. It is also cleared by SPI SW RESET or by a system reset.

## 10.9.5 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.



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.

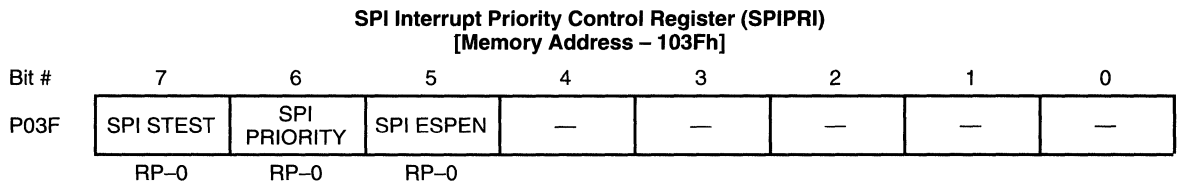
**Note:**

The SPICLK pin always functions as the SPICLK input pin in the slave mode (i.e., SPICLK2=0), even if SPICLK FUNCTION = 0.

- 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.



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.



## 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.



## 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  $\mu$ s for each kilohm of source output impedance or a minimum of 1  $\mu$ s for low-impedance sources.

## 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  $\mu$ s for each kilohm of source output impedance or a minimum of 1  $\mu$ s 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).

```

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

      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

```

### 11.4.1 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<sub>REF</sub>) 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 <sub>REF</sub> Source
0	0	0	V <sub>CC3</sub> <sup>†</sup>
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

<sup>†</sup> Pin AN0 cannot be selected as positive voltage reference.

## 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.

## 11.4.6 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.



## 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



## 12.1.2 Control Registers

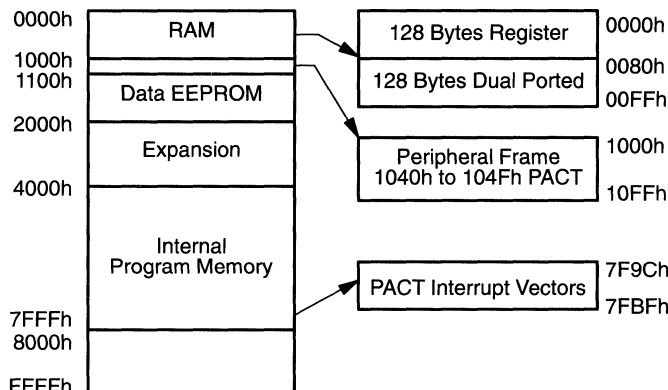
The PACT control registers are located at addresses 1040h to 104Fh and occupy peripheral file frame 4. The function of each is shown in Table 12–1.

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	RXBUF	PACT-SCI RX Data Register	Contains the data received by the SCI.
P047	TXBUF	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).

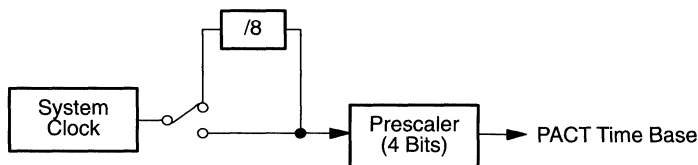
Figure 12–2. TMS370 Memory Map Highlighting PACT Areas



### 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.

PACT PRESCALE SELECT	Divide Rate		PACT PRESCALE SELECT	Divide Rate	
	FAST MODE SELECT			FAST MODE SELECT	
	1	0		1	0
3 2 1 0			3 2 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

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).

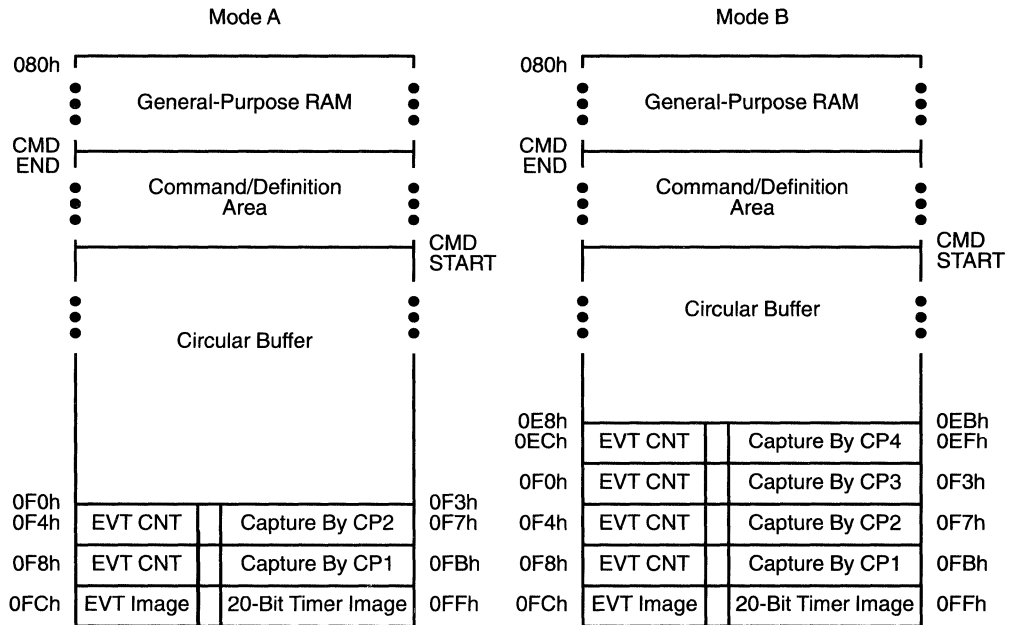
**Note:**

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.

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



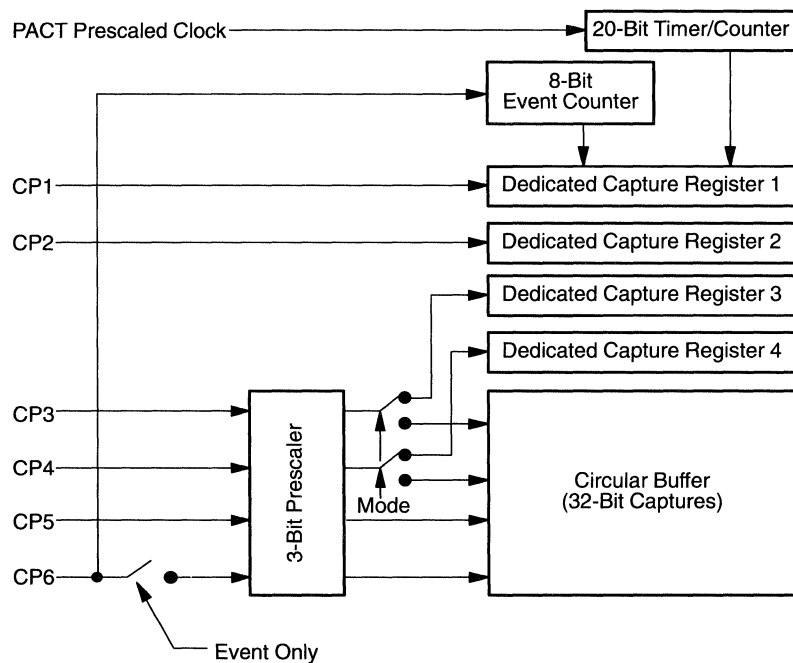
Mode A provides two, and Mode B provides four dedicated 32-bit storage locations, which follow a circular storage buffer. Modes A and B are described in Section 12.4.

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.

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.

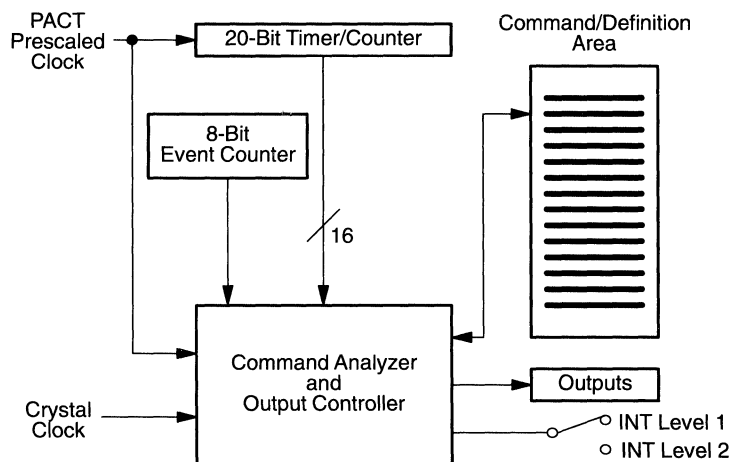
## 12.5 Control and Outputs

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



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.

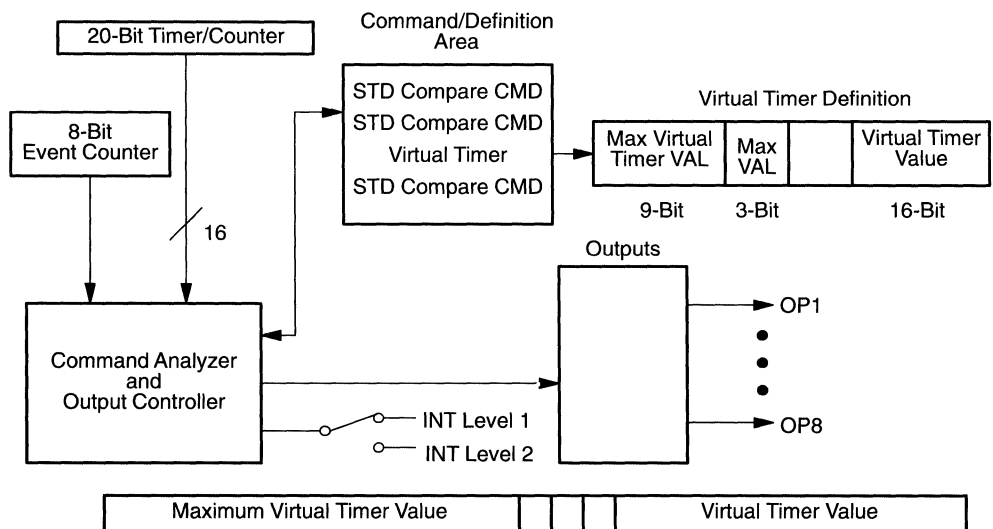
You cannot use this command to vary the period of the PWM.

## 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



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.

				Event 1 actions	Event 2 actions	Pin Select	Event 2 Compare Value	Event 1 Compare Value
--	--	--	--	--------------------	--------------------	------------	--------------------------	--------------------------

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.

Max Event Value				Actions		Virtual Timer Value
-----------------	--	--	--	---------	--	---------------------



## 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.

<b>Definitions</b>	<b>Time Slots</b>
Virtual timer definition	2
Offset timer definition	2/3
Baud rate timer definition	2

<b>Commands</b>	<b>Time Slots</b>
Standard Compare command	1
Conditional Compare command	1
Double Event Compare command	1

### 12.6.2 Offset Timer Definition-Time From Last Event

Maximum Event Counter Value										Virtual Timer Value										1
-----------------------------	--	--	--	--	--	--	--	--	--	---------------------	--	--	--	--	--	--	--	--	--	---

D31

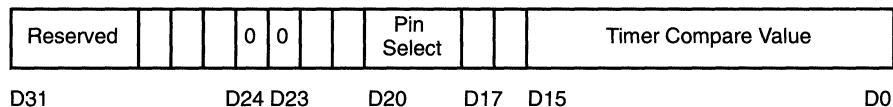
D19

D0

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.

## 12.6.4 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.**

- 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.

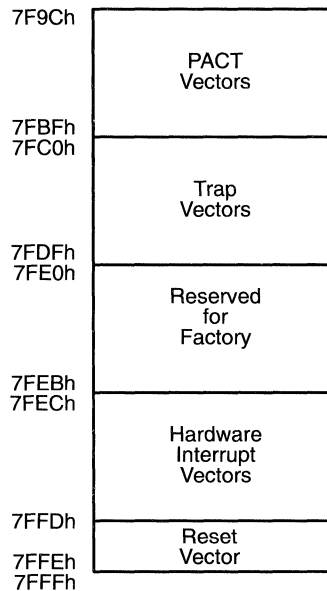
## 12.7 Interrupts

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.

## 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.

## 12.10 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  $\mu$ s, 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  $\mu$ s 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.

## 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
0011 # .byte 0,0,1,0
0012     virtmr     1000,enable      ;period = 1000 uSec
0013 # .byte 0,0,52,31
0014     stdcmp     800,op1,opp_act|enable ;80% duty, pin 1
0015 # .byte 32,3,0,10
0016 len.equ  $-table
0017
0018     .text 6000h
0019 cmd_st.equ 0EBh
0020 start mov #7,P04F ;disable the watchdog
0021 ;copy PACT commands/def. into ram
0022     mov #len,b ;length of cmd/def area
0023     movw #(cmd_st-len+1),r3 ;R2:R3 points to area
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 #07,p04f ;set to mode B
0031     mov #cmd_st,p041 ;cmd/def start at 0ebh
0032     mov #(cmd_st-len+1),p042 ;cmd/def end = 0E0h
0033 ;set prescale to 5, 1 usec res, enable cmd/def area
0034 .. mov #034h,p040
0035
0036 ;PWM running without processor intervention
0037     idle
0038     .end

```



Table 12–5. Peripheral File Frame 4: PACT Control Registers

Designation	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
RXBUF	1046h	P046	PACT RXDT7	PACT RXDT6	PACT RXDT5	PACT RXDT4	PACT RXDT3	PACT RXDT2	PACT RXDT1	PACT RXDT0
TXBUF	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

- 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.

### 12.11.3 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.

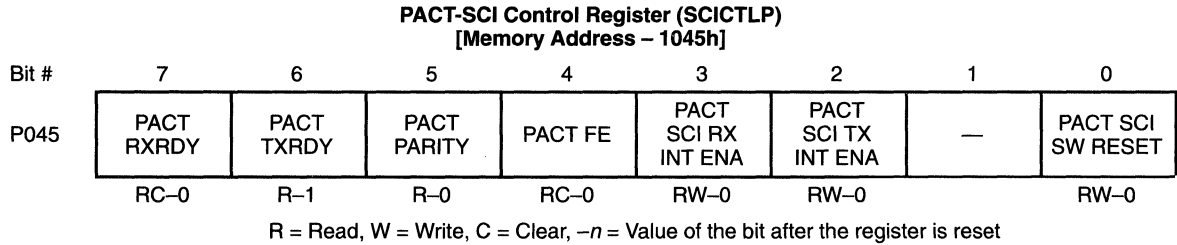
These bits define the end address of the command/definition area. There are 32 possible locations for the command/definition area end. The address is the same as if you consider bit 7 of this register to be set to 1, and bit 1 and bit 0 to be set to 0. A table of the bits and the corresponding addresses 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.

### 12.11.5 PACT-SCI Control Register (SCICTLP)

The SCICTLP register controls the functions of the mini-SCI.



- 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.

### 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.
    
```

### 12.11.10 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.

### 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.

**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.

### 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.



### 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.

### 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.



## 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:

<u>MNEMONIC</u>	<u>SOURCE</u>	<u>DESTINATION</u>
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.

<u>LABEL</u>	<u>INSTRUCTION</u>	<u>OPERANDS</u>	<u>COMMENT</u>
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.

### 13.3 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.

### 13.3.1.2 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:

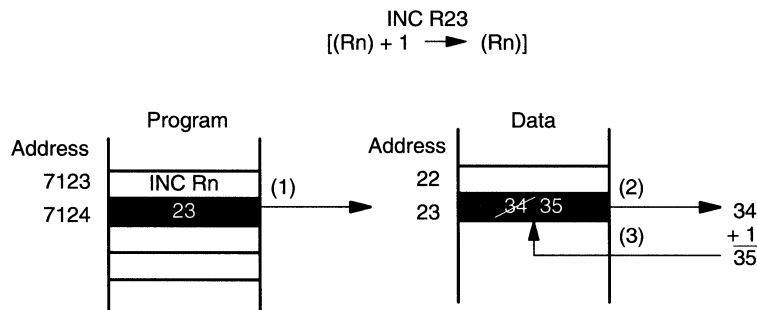
```

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.

### 13.3.1.4 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:

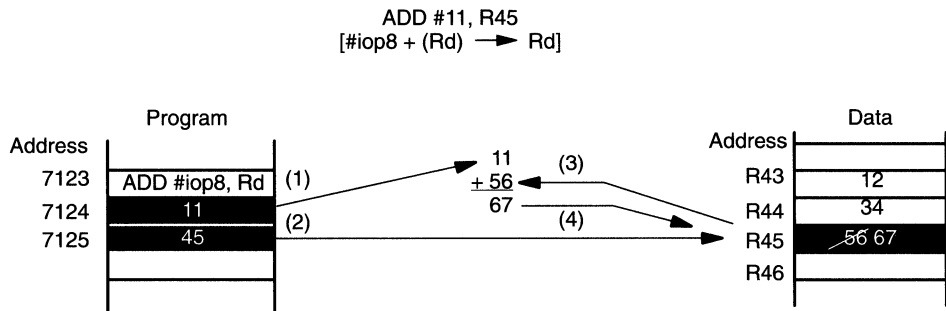
```

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.

### 13.3.1.6 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.

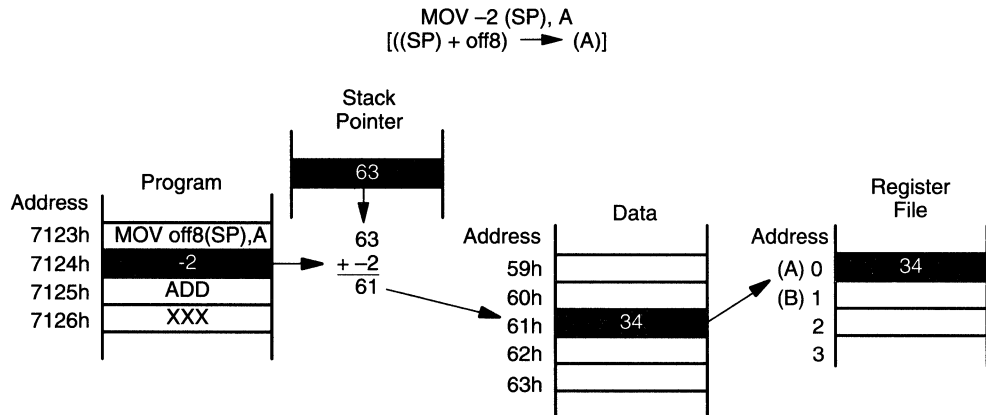
This addressing mode is useful in accessing arguments that are passed to a subroutine on the stack. You must insure that the resulting address location is within the implemented register file, because overflows or underflows will execute without warning.

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

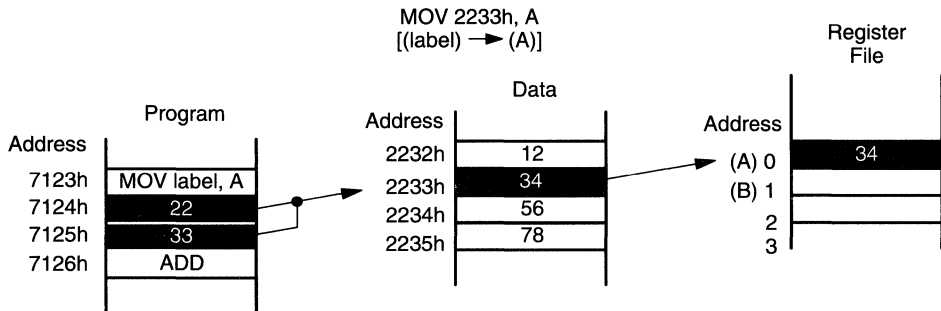




### 13.3.2.1 Direct Addressing Modes

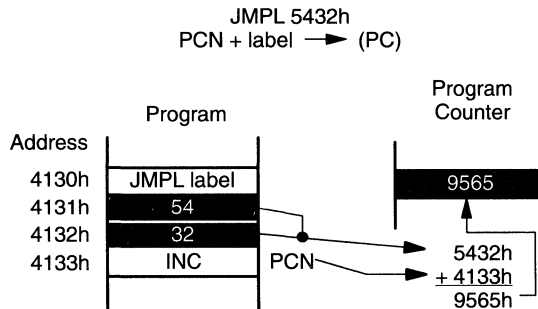
**Direct addressing** mode instructions use an address as the operand. The 16-bit address is written as either a constant value or a label and immediately follows the opcode in the source code. The **absolute direct addressing** mode acts upon the address itself as shown in Figure 13–7.

Figure 13–7. Absolute Direct Addressing Mode



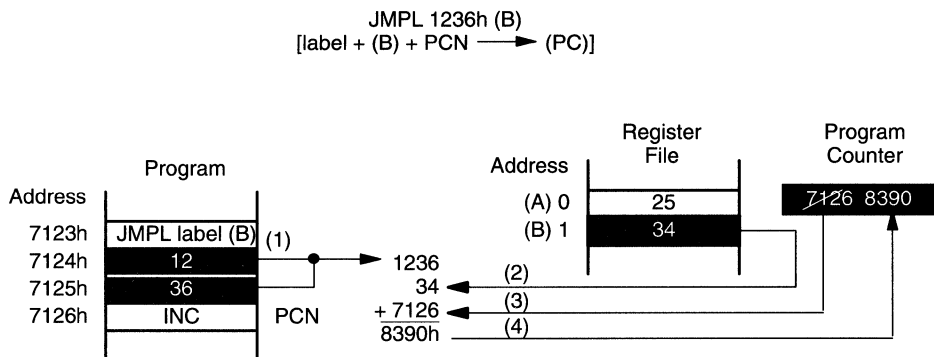
The **relative direct addressing** mode (Figure 13–8) adds the address of the next instruction to the 16-bit operand to produce the address of the succeeding instruction. If a label is used in the instruction, the assembler automatically calculates the offset to use as the operand.

Figure 13–8. Relative Direct Addressing Mode



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



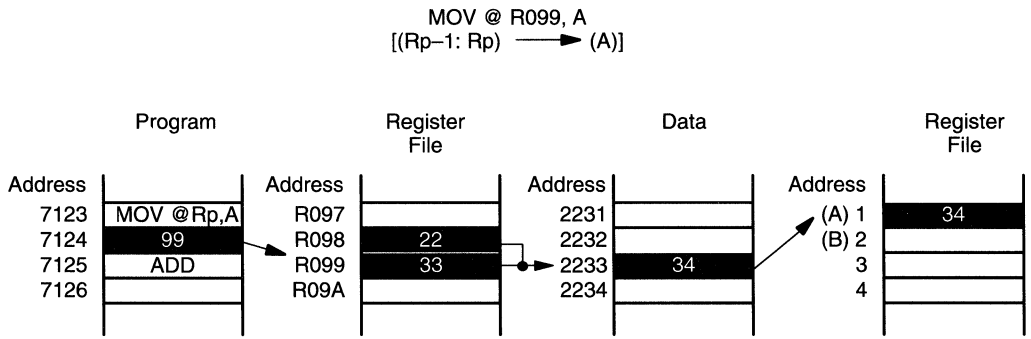
**Note:** Numbers in parentheses represent order of execution.

### 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.

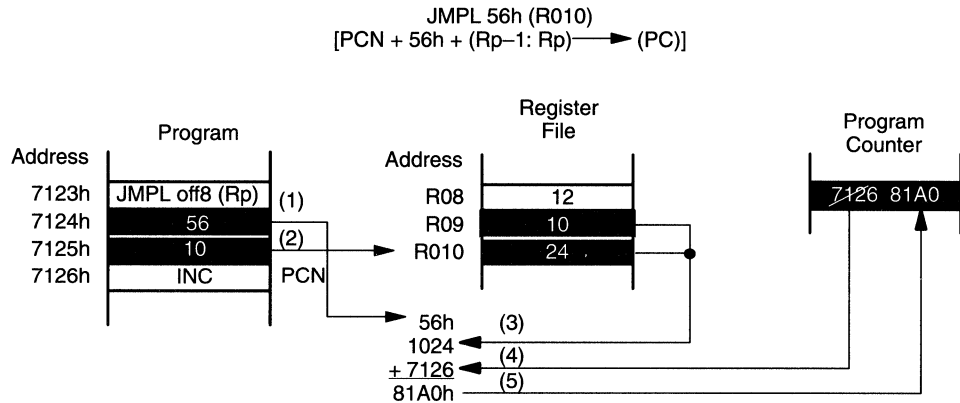
Figure 13–11 shows how the **absolute indirect addressing** mode uses the register pair in the calculation.

Figure 13–11. Absolute Indirect Addressing Mode



The **relative 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)*

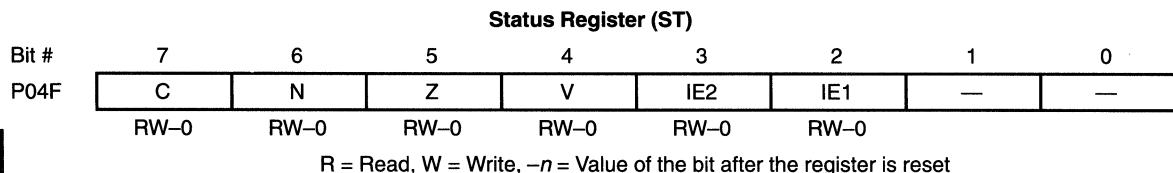


Table 13–3. TMS370 Family Instruction Overview (Continued)

Mnemonic	Opcode	Bytes	Cycles t <sub>c</sub>	Status C N Z V	Operation Description	
BTJZ†	A,Pd,off8 B,A,off8 B,Pd,off8 Rs,A,off8 Rs,B,off8 Rs,Rd,off8 #iop8,A,off8 #iop8,B,off8 #iop8,Rd,off8 #iop8,Pd,off8	87 67 97 17 37 47 27 57 77 A7	3 2 3 3 3 4 3 3 4 4	10 10 10 9 9 11 8 8 10 11	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.
CALL	label @Rp label(B) off8(Rp)	8E 9E AE F4 EE	3 2 3 4	13 12 15 20	– – – –	Push PC MSbyte, PC LSbyte, XADDR → (PC)
CALLR	label @Rp label(B) off8(Rp)	8F 9F AF F4 EF	3 2 3 4	15 14 17 22	– – – –	Call relative Push PC MSbyte, PC LSbyte, PCN + (XADDR) → (PC)
CLR	A B Rd	B5 C5 D5	1 1 2	8 8 6	0 0 1 0	0 → (Rd) Clear the destination operand.
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 @Rp,A label(B),A off8(Rp),A off8(SP),A B,A Rs,A Rs,B Rs,Rd #iop8,A #iop8,B #iop8,Rd	8D 9D AD F4 ED F3 6D 1D 3D 4D 2D 5D 7D	3 2 3 4 2 1 2 2 3 2 2 3	11 10 13 18 8 8 7 7 9 6 6 8	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.
CMPBIT	Rname Pname	75 A5	3 3	8 10	0 x x 0	Complement bit; invert the bit
COMPL	A B Rn	BB CB DB	1 1 2	8 8 6	x x x 0	2s complement; 00h – (s) → (d)

† 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 <sub>c</sub>	Status C N Z V	Operation Description
INV A B Rn	B4 C4 D4	1 1 2	8 8 6	0 x x 0	NOT(d) → (d) 1s complement the destination operand.
JBIT0† Rname,off8 Pname,off8	77 A7	4 4	10 11	0 x x 0	Jump if bit = 0
JBIT1† Rname,off8 Pname,off8	76 A6	4 4	10 11	0 x x 0	Jump if bit = 1
JMP off8	00	2	7	– – – –	PCN + off8 → (PC) Jump unconditionally using an 8-bit offset.
JMPL label @Rp label(B) off8(Rp)	89 99 A9 F4 E9	3 2 3 4	9 8 11 16	– – – –	PCN + D → (PC) Jump unconditionally using a 16-bit offset.
Jcnd† JC JEQ JG JGE JHS JL JLE JLO JN JNC JNE JNV JNZ JP JPZ JV JZ	03 02 0E 0D 0B 09 0A 0F 01 07 06 0C 06 04 05 08 02	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	– – – –	Conditional jump Carry Jump equal Greater than, signed Greater than or equal, signed Higher or same, unsigned Less than, signed Less than or equal, signed Lower value, unsigned Negative, signed No carry Jump not equal No overflow, signed Not zero Positive, signed Positive or zero, signed Overflow, signed 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.

Table 13–3. TMS370 Family Instruction Overview (Continued)

Mnemonic	Opcode	Bytes	Cycles t <sub>C</sub>	Status C N Z V	Operation Description
OR A,Pd B,A B,Pd Rs,A Rs,B Rs,Rd #iop8,A #iop8,B #iop8,Rd #iop8,Pd	84 64 94 14 34 44 24 54 74 A4	2 1 2 2 2 3 2 2 3 3	9 8 9 7 7 9 6 6 8 10	0 x x 0	(s) OR (d) → (d)  Logically OR the source and destination operands, and store the results at the destination address.
POP A B Rd ST	B9 C9 D9 FC	1 1 2 1	9 9 7 8	0 x x 0   x x x x	((SP)) → (d) (SP) - 1 → (SP)
PUSH A B Rs ST	B8 C8 D8 FB	1 1 2 1	9 9 7 8	0 x x 0   - - - -	(SP) + 1 → (SP) (s) → ((SP)) Copy the operand onto the stack. Copy the status register onto the stack.
RL A B Rn	BE CE DE	1 1 2	8 8 6	x x x 0	Bit(n) → Bit(n + 1) Bit(7) → Bit(0) and Carry
RLC A B Rn	BF CF DF	1 1 2	8 8 6	x x x 0	Bit(n) → Bit(n + 1) Carry → Bit(0) Bit(7) → Carry
RR A B Rn	BC CC DC	1 1 2	8 8 6	x x x 0	Bit(n + 1) → Bit(n) Bit(0) → Bit(7) and Carry
RRC A B Rn	BD CD DD	1 1 2	8 8 6	x x x 0	Bit(n + 1) → Bit(n) Carry → Bit(7) 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 Rs,A Rs,B Rs,Rd #iop8,A #iop8,B #iop8,Rd	6B 1B 3B 4B 2B 5B 7B	1 2 2 3 2 2 3	8 7 7 9 6 6 8	x x x x	(d) - (s) - 1 + (C) → (d) Subtract with borrow. Destination minus source minus 1 plus carry; stored at the destination address.
SBIT0 Rname Pname	73 A3	3 3	8 10	0 x x 0	Set bit to 0
SBIT1 Rname Pname	74 A4	3 3	8 10	0 x x 0	Set bit to 1

**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 13–4. TMS370 Family Opcode/Instruction Map

		MSN															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
L S N	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,Rd 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,Rd 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,Rd 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

## **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**                    **ADD** *s, Rd*

**Execution**                 $(s) + (Rd) \rightarrow (Rd)$

Options	inst	operands	bytes	cycles	opcode	operation
	ADD	B,A	1	8	68	$(B)+(A) \rightarrow (A)$
	ADD	Rs,A	2	7	18	$(Rs)+(A) \rightarrow (A)$
	ADD	Rs,B	2	7	38	$(Rs)+(B) \rightarrow (B)$
	ADD	Rs,Rd	3	9	48	$(Rs)+(Rd) \rightarrow (Rd)$
	ADD	#iop8,A	2	6	28	$iop8+(A) \rightarrow (A)$
	ADD	#iop8,B	2	6	58	$iop8+(B) \rightarrow (B)$
	ADD	#iop8,Rd	3	8	78	$iop8+(Rd) \rightarrow (Rd)$

**Status Bits Affected**    **C** Set to 1 on carry-out of  $(s) + (Rd)$   
**Z** Set on result  
**N** Set on result  
**V**  $(C \text{ XOR } N) \text{ AND } (\text{Source [bit 7] XNOR Destination [bit 7]})$

**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****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**
**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**
**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
```

<b>Syntax</b>	<b>CLRC</b>										
<b>Execution</b>	Set status bits										
<b>Options</b>	<table><thead><tr><th>inst</th><th>operands</th><th>bytes</th><th>cycles</th><th>opcode</th></tr></thead><tbody><tr><td>CLRC</td><td>none</td><td>1</td><td>9</td><td>B0</td></tr></tbody></table>	inst	operands	bytes	cycles	opcode	CLRC	none	1	9	B0
inst	operands	bytes	cycles	opcode							
CLRC	none	1	9	B0							
<b>Status Bits Affected</b>	<b>C</b> ← 0 <b>N</b> Set on value of register A <b>Z</b> Set on value of register A <b>V</b> ← 0										
<b>Description</b>	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.										
<b>Example</b>	<code>LABEL CLRC ;Clear the carry bit</code>										

**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**                    **DEC Rn**

**Execution**                 $(Rn) - 1 \rightarrow (Rn)$

<b>Options</b>	<b>inst</b>	<b>operands</b>	<b>bytes</b>	<b>cycles</b>	<b>opcode</b>	<b>operation</b>
	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** `DIV Rs, A`

**Execution** `A:B/(Rs) → A(=quo), B(=rem)`

<b>Options</b>	<b>inst</b>	<b>operands</b>	<b>bytes</b>	<b>cycles</b>	<b>opcode</b>	<b>operation</b>
	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**                    **DSB** *s,Rd*
**Execution**                 $(Rd) - (s) - 1 + (C) \rightarrow (Rd)$  (decimal result)

Options	inst	operands	bytes	cycles	opcode	operation
	DSB	B,A	1	10	6F	$(A)-(B)-1+(C) \rightarrow (A)$
	DSB	Rs,A	2	9	1F	$(A)-(Rs)-1+(C) \rightarrow (A)$
	DSB	Rs,B	2	9	3F	$(B)-(Rs)-1+(C) \rightarrow (B)$
	DSB	Rs,Rd	3	11	4F	$(Rd)-(Rs)-1+(C) \rightarrow (Rd)$
	DSB	#iop8,A	2	8	2F	$(A)-iop8-1+(C) \rightarrow (A)$
	DSB	#iop8,B	2	8	5F	$(B)-iop8-1+(C) \rightarrow (B)$
	DSB	#iop8,Rd	3	10	7F	$(Rd)-iop8-1+(C) \rightarrow (Rd)$

**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
    
```

<b>Syntax</b>	<b>EINTH</b>										
<b>Execution</b>	04h → (ST)										
<b>Options</b>	<table><thead><tr><th><b>inst</b></th><th><b>operands</b></th><th><b>bytes</b></th><th><b>cycles</b></th><th><b>opcode</b></th></tr></thead><tbody><tr><td>EINTH</td><td>none</td><td>2</td><td>6</td><td>F0 04</td></tr></tbody></table>	<b>inst</b>	<b>operands</b>	<b>bytes</b>	<b>cycles</b>	<b>opcode</b>	EINTH	none	2	6	F0 04
<b>inst</b>	<b>operands</b>	<b>bytes</b>	<b>cycles</b>	<b>opcode</b>							
EINTH	none	2	6	F0 04							
<b>Status Bits Affected</b>	<b>C</b> ← 0 <b>N</b> ← 0 <b>Z</b> ← 0 <b>V</b> ← 0 <b>IE1</b> ← 1 <b>IE2</b> ← 0										
<b>Description</b>	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.										
<b>Example</b>	<pre>LABEL      EINTH          ;All level 1 interrupts are enabled.</pre>										

<b>Syntax</b>	<b>IDLE</b>										
<b>Execution</b>	(PC) + 1 → (PC) after return from interrupt										
<b>Options</b>	<table><thead><tr><th><b>inst</b></th><th><b>operands</b></th><th><b>bytes</b></th><th><b>cycles</b></th><th><b>opcode</b></th></tr></thead><tbody><tr><td>IDLE</td><td>none</td><td>1</td><td>6 (minimum)</td><td>F6</td></tr></tbody></table>	<b>inst</b>	<b>operands</b>	<b>bytes</b>	<b>cycles</b>	<b>opcode</b>	IDLE	none	1	6 (minimum)	F6
<b>inst</b>	<b>operands</b>	<b>bytes</b>	<b>cycles</b>	<b>opcode</b>							
IDLE	none	1	6 (minimum)	F6							
<b>Status Bits Affected</b>	None										
<b>Description</b>	<p>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.</p> <ul style="list-style-type: none"><li><input type="checkbox"/> In standby mode, the on-chip oscillator and timer 1 module remain active.</li><li><input type="checkbox"/> In halt mode, the oscillator is off, and the chip consumes the least amount of power.</li></ul> <p>If you execute an IDLE instruction when low-power modes are disabled through a programmable contact (mask-ROM devices only), the device will <i>always</i> enter the idle mode.</p> <p>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.</p>										
<b>Examples</b>	<pre>LABEL      IDLE                ;Enter idle mode and                                 ;wait for interrupt</pre>										

**Syntax** `INCW #iop8,Rp`

**Execution** `(Rp) + #iop8 → (Rp)`

<b>Options</b>	<b>inst</b>	<b>operands</b>	<b>bytes</b>	<b>cycles</b>	<b>opcode</b>	<b>operation</b>
	INCW	#iop8,Rp	3	11	70	<code>iop8+(Rp-1:Rp) → (Rp-1:Rp)</code> 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 (Rp))`

**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**                    **JBIT0** *name, off8*

**Execution**                If bit (name) = 0, then PCN + off8 → (PC), else PCN → PC

<b>Options</b>	<b>inst</b>	<b>operands</b>	<b>bytes</b>	<b>cycles</b>	<b>opcode</b>
	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**
**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)
Jump if Greater	JG	0E	x	–	–	–	—Signed Operation— 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.

<b>Syntax</b>	<b>JMP</b> <i>off8</i>					
<b>Execution</b>	PCN + off8 → (PC)					
<b>Options</b>	<b>inst</b>	<b>operands</b>	<b>bytes</b>	<b>cycles</b>	<b>opcode</b>	<b>operation</b>
	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
```



<b>Syntax</b>	<b>LDSP</b>										
<b>Execution</b>	(B) → (SP)										
<b>Options</b>	<table><thead><tr><th><b>inst</b></th><th><b>operands</b></th><th><b>bytes</b></th><th><b>cycles</b></th><th><b>opcode</b></th></tr></thead><tbody><tr><td>LDSP</td><td>none</td><td>1</td><td>7</td><td>FD</td></tr></tbody></table>	<b>inst</b>	<b>operands</b>	<b>bytes</b>	<b>cycles</b>	<b>opcode</b>	LDSP	none	1	7	FD
<b>inst</b>	<b>operands</b>	<b>bytes</b>	<b>cycles</b>	<b>opcode</b>							
LDSP	none	1	7	FD							
<b>Status Bits Affected</b>	None										
<b>Description</b>	LDSP copies the contents of register B to the stack pointer (SP). Use LDSP to initialize the stack pointer.										
<b>Example</b>	<pre>MOV      #080h,B          ;Register B = SP value.  LABEL   LDSP             ;Copy register B to the stack                            ;pointer.</pre>										

**Syntax**                    **MOV** *s,d*
**Execution**                (*s*) → (*d*)

<b>Options</b>	<b>inst</b>	<b>operands</b>	<b>bytes</b>	<b>cycles</b>	<b>opcode</b>	<b>operation</b>
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

**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)
    
```

<b>Syntax</b>	<b>NOP</b>										
<b>Execution</b>	(PC) + 1 → (PC)										
<b>Options</b>	<table><thead><tr><th><b>inst</b></th><th><b>operands</b></th><th><b>bytes</b></th><th><b>cycles</b></th><th><b>opcode</b></th></tr></thead><tbody><tr><td>NOP</td><td>none</td><td>1</td><td>7</td><td>FF</td></tr></tbody></table>	<b>inst</b>	<b>operands</b>	<b>bytes</b>	<b>cycles</b>	<b>opcode</b>	NOP	none	1	7	FF
<b>inst</b>	<b>operands</b>	<b>bytes</b>	<b>cycles</b>	<b>opcode</b>							
NOP	none	1	7	FF							
<b>Status Bits Affected</b>	None										
<b>Description</b>	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.										
<b>Example</b>	LABEL      NOP										

**Syntax**                    **POP** *d*

**Execution**                 $((SP)) \rightarrow (d)$   
 $(SP) - 1 \rightarrow (SP)$   
(Move value then decrement SP)

**Options**

inst	operands	bytes	cycles	opcode	operation
POP	A	1	9	B9	$((SP)) \rightarrow (A); (SP) - 1 \rightarrow (SP)$
POP	B	1	9	C9	$((SP)) \rightarrow (B); (SP) - 1 \rightarrow (SP)$
POP	Rd	2	7	D9	$((SP)) \rightarrow (Rd); (SP) - 1 \rightarrow (SP)$
POP	ST	1	8	FC	$((SP)) \rightarrow (ST); (SP) - 1 \rightarrow (SP)$

**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**

**RL** *Rn*

**Execution**

Bit(*n*) → Bit(*n*+1)  
 Bit(7) → Bit(0) and carry

**Options**

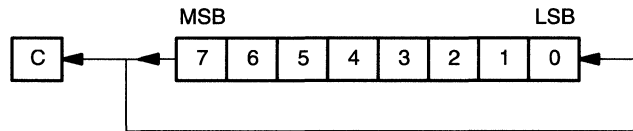
inst	operands	bytes	cycles	opcode
RL	A	1	8	BE
RL	B	1	8	CE
RL	Rn	2	6	DE

**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
```

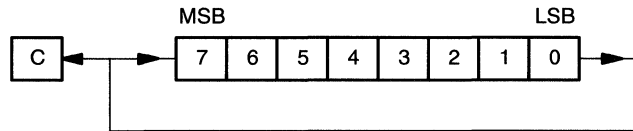
**Syntax** **RR** *Rn*

**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****RTI****Execution**

((SP)) → (PC LSbyte)  
(SP) - 1 → (SP)  
((SP)) → (PC MSbyte)  
(SP) - 1 → (SP)  
((SP)) → (ST)  
(SP) - 1 → (SP)

**Options**

<b>inst</b>	<b>operands</b>	<b>bytes</b>	<b>cycles</b>	<b>opcode</b>
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**                    **SBB** *s,Rd*
**Execution**                 $(Rd) - (s) - 1 + (C) \rightarrow (Rd)$ 

Options	inst	operands	bytes	cycles	opcode	operation
	SBB	B,A	1	8	6B	$(A) - (B) - 1 + (C) \rightarrow (A)$
	SBB	Rs,A	2	7	1B	$(A) - (Rs) - 1 + (C) \rightarrow (A)$
	SBB	Rs,B	2	7	3B	$(B) - (Rs) - 1 + (C) \rightarrow (B)$
	SBB	Rs,Rd	3	9	4B	$(Rd) - (Rs) - 1 + (C) \rightarrow (Rd)$
	SBB	#iop8,A	2	6	2B	$(A) - iop8 - 1 + (C) \rightarrow (A)$
	SBB	#iop8,B	2	6	5B	$(B) - iop8 - 1 + (C) \rightarrow (B)$
	SBB	#iop8,Rd	3	8	7B	$(Rd) - iop8 - 1 + (C) \rightarrow (Rd)$

**Status Bits Affected**

- C** Set to 1 if no borrow; 0 otherwise
- N** Set on result
- Z** Set on result
- V**  $((C \text{ XOR } N) \text{ AND } (\text{Source}[\text{Bit } 7] \text{ XOR } \text{Destination}[\text{Bit } 7]))$

**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**                    **SBIT1** *name***Execution**                1 → <name>

<b>Option</b>	<b>inst</b>	<b>operands</b>	<b>bytes</b>	<b>cycles</b>	<b>opcode</b>	<b>operation</b>
	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
```

<b>Syntax</b>	<b>STSP</b>										
<b>Execution</b>	(SP) → (B)										
<b>Options</b>	<table><thead><tr><th>inst</th><th>operands</th><th>bytes</th><th>cycles</th><th>opcode</th></tr></thead><tbody><tr><td>STSP</td><td>none</td><td>1</td><td>8</td><td>FE</td></tr></tbody></table>	inst	operands	bytes	cycles	opcode	STSP	none	1	8	FE
inst	operands	bytes	cycles	opcode							
STSP	none	1	8	FE							
<b>Status Bits Affected</b>	None										
<b>Description</b>	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.										
<b>Example</b>	<pre>LABEL      STSP          ;Copy the contents of stack pointer            ;to register B</pre>										

**Syntax****SWAP** *Rn***Execution**

Bits (7,6,5,4, / 3,2,1,0) → Bits (3,2,1,0, / 7,6,5,4)

**Options**

<b>inst</b>	<b>operands</b>	<b>bytes</b>	<b>cycles</b>	<b>opcode</b>
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
```

<b>Syntax</b>	<b>TST {A   B}</b>															
<b>Execution</b>	C,N,Z,V bits affected															
<b>Options</b>	<table><thead><tr><th><b>inst</b></th><th><b>operands</b></th><th><b>bytes</b></th><th><b>cycles</b></th><th><b>opcode</b></th></tr></thead><tbody><tr><td>TST</td><td><b>A</b></td><td>1</td><td>9</td><td>B0</td></tr><tr><td>TST</td><td><b>B</b></td><td>1</td><td>10</td><td>C6</td></tr></tbody></table>	<b>inst</b>	<b>operands</b>	<b>bytes</b>	<b>cycles</b>	<b>opcode</b>	TST	<b>A</b>	1	9	B0	TST	<b>B</b>	1	10	C6
<b>inst</b>	<b>operands</b>	<b>bytes</b>	<b>cycles</b>	<b>opcode</b>												
TST	<b>A</b>	1	9	B0												
TST	<b>B</b>	1	10	C6												
<b>Status Bits Affected</b>	<b>C</b> ← 0 <b>N</b> Set or cleared based on operand <b>Z</b> Set or cleared based on operand <b>V</b> ← 0															
<b>Description</b>	TST sets the status bits according to the value in register A or B. This allows conditional jumps on the value in the register.															
<b>Example</b>	<pre>LABEL    TST A        ;Check for zero and negative            ;conditions in register A             TST B        ;Check for zero and negative            ;conditions in register B</pre>															

**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
    
```



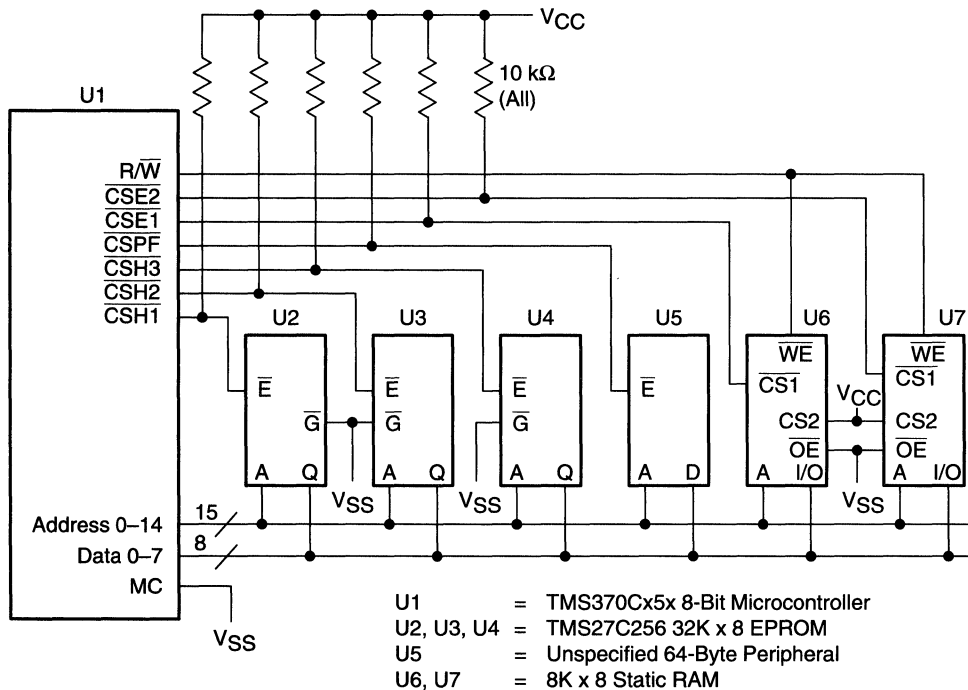
### 14.1 Microcomputer Interface Example

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



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
$t_w(\text{COL})$	CLKOUT low pulse duration	$0.5 t_c - 25$	$0.5 t_c$	ns
$t_w(\text{COH})$	CLKOUT high pulse duration	$0.5 t_c$	$0.5 t_c + 20$	ns
$t_d(\text{COL-A})$	Delay time, CLKOUT low to address R/W, and $\overline{\text{OCF}}$ valid		$0.25 t_c + 75$	ns
$t_v(\text{A})$	Address valid to $\overline{\text{EDS}}$ , $\overline{\text{CSE1}}$ , $\overline{\text{CSE2}}$ , $\overline{\text{CSH1}}$ , $\overline{\text{CSH2}}$ , $\overline{\text{CSH3}}$ , and $\overline{\text{CSPF}}$ low	$0.5 t_c - 90$		ns
$t_{su}(\text{D})$	Write data set-up time to $\overline{\text{EDS}}$ high	$0.75 t_c - 80^\ddagger$		ns
$t_h(\text{EH-A})$	Address, R/W, and $\overline{\text{OCF}}$ hold time from $\overline{\text{EDS}}$ , $\overline{\text{CSE1}}$ , $\overline{\text{CSE2}}$ , $\overline{\text{CSH1}}$ , $\overline{\text{CSH2}}$ , $\overline{\text{SH3}}$ , and $\overline{\text{CSPF}}$ high	$0.5 t_c - 60$		ns
$t_h(\text{EH-D})\text{W}$	Write data hold time from $\overline{\text{EDS}}$ high	$0.75 t_c + 15$		ns
$t_d(\text{DZ-EL})$	Delay time, data bus high impedance to $\overline{\text{EDS}}$ low (read cycle)	$0.25 t_c - 35$		ns
$t_d(\text{EH-D})$	Delay time, $\overline{\text{EDS}}$ high to data bus enable (read cycle)	$1.25 t_c - 40$		ns
$t_d(\text{EL-DV})$	Delay time, $\overline{\text{EDS}}$ low to read data valid		$t_c - 95^\ddagger$	ns
$t_h(\text{EH-D})\text{R}$	Read data hold time from $\overline{\text{EDS}}$ high	0		ns
$t_{su}(\text{WT-COH})$	$\overline{\text{WAIT}}$ set-up time to CLKOUT high	$0.25 t_c + 70^\S$		ns
$t_h(\text{COH-WT})$	$\overline{\text{WAIT}}$ hold time from CLKOUT high	0		ns
$t_d(\text{ED-WTV})$	Delay time, $\overline{\text{EDS}}$ low to $\overline{\text{WAIT}}$ valid		$0.5 t_c - 60$	ns
$t_w$	Pulse duration, $\overline{\text{EDS}}$ , $\overline{\text{CSE1}}$ , $\overline{\text{CSE2}}$ , $\overline{\text{CSH1}}$ , $\overline{\text{CSH2}}$ , $\overline{\text{CSH3}}$ , and $\overline{\text{CSPF}}$ low	$t_c - 80^\ddagger$	$t_c + 40^\ddagger$	ns
$t_d(\text{AV-DV})\text{R}$	Delay time, address valid to read data valid		$1.5 t_c - 115^\ddagger$	ns
$t_d(\text{AV-WTV})$	Delay time, address valid to $\overline{\text{WAIT}}$ valid		$t_c - 115$	ns
$t_d(\text{AV-EH})$	Delay time, address valid to $\overline{\text{EDS}}$ high (end of write)	$1.5 t_c - 85^\ddagger$		ns

$^\dagger t_c$  = system clock cycle time =  $4/\text{CLKIN}$ .

$^\ddagger$  If wait states, PFWait, or the autowait feature is used, add  $t_c$  to this value for each wait state invoked.

$^\S$  If the autowait feature is enabled, the  $\overline{\text{WAIT}}$  input may assume a "don't care" condition until the third cycle of the access. The  $\overline{\text{WAIT}}$  signal must be synchronized with the high pulse of the CLKOUT signal while still conforming to the minimum set-up time.

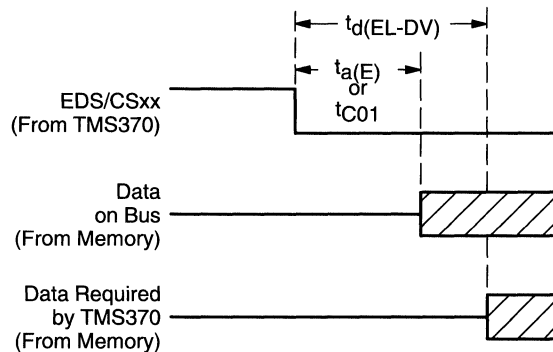
The HM6264-15 RAM can extend the TMS370's minimum address-to-data set-up time with no wait states. When you access external RAM comparable to the Hitachi device, you can turn off the autowait feature to speed up the system.

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 ( $\overline{CS_{xx}}$  or  $\overline{EDS}$ ) must be used with external memory to validate the memory cycle. Connecting the chip-select pin ( $\overline{CS_{xx}}$ ) 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

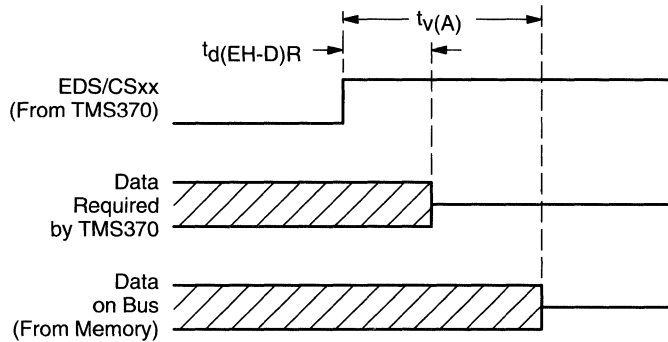


Name	Description	Formula	Time
$t_d(EL-DV)$	TMS370 (0 wait) requires data	$t_{c-95}$	105 ns(too fast)
$t_d(EL-DV)$	TMS370 (1 wait) requires data	$2 t_{c-95}$	305 ns(ok)
$t_d(EL-DV)$	TMS370 (pf wait) requires data	$3 t_{c-95}$	505 ns(ok)
$t_a(E)$	TMS27C256-25 provides data		250 ns(ok)
$t_{C01}$	HM6264-15 provides data		150 ns(ok)

**14.1.1.4 Read Data Hold After Chip Select High Requirements**

The high transition of the chip-select signal ( $\overline{CH_{xx}}$ ) 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



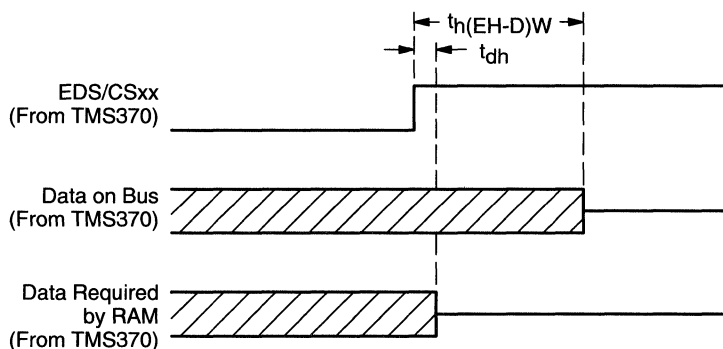
Name	Description	Formula	Time
$t_d(EH-D)R$	TMS370 (all) needs data	—	0 ns
$t_v(A)$	TMS27C256-25 data	—	0 ns
$t_{HZ}^1$	HM6264-15 holds data	—	0 ns

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.

### 14.1.4 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 through 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
          AND    #0EFh,P011    ;autowait cycles
          ;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.

```

## 14.2 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 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 EEPORG 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)
PROGRAM  MOV  A,TEMP2           ; x1010    x1100
                                ;          Save data
                                ;          Read current data
MOV      @ADDR1,A             ; x1100
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    EEPORG                ;          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    EEPORG                ;          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

```

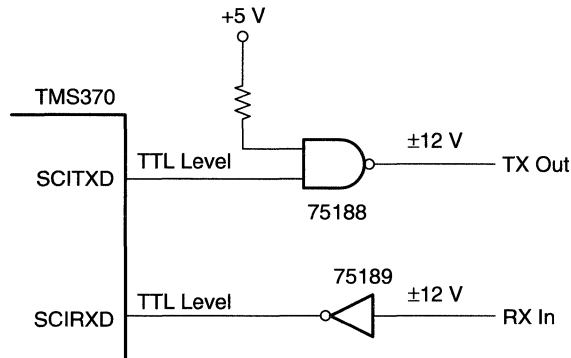
### 14.3.2 SCI Port Interfacing

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:

<u>Parameter</u>	<u>Options</u>
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.



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*.

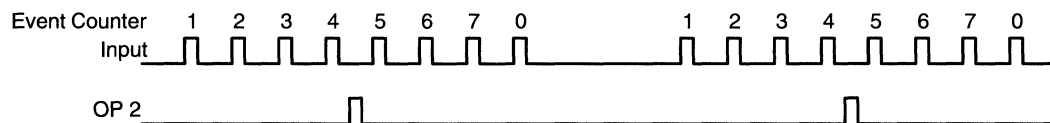
## 14.5 PACT Module

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).

### **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.

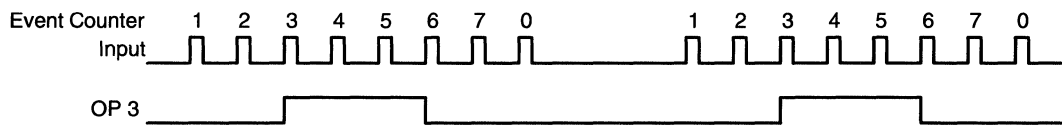
Lines 37 and 38 clear the event counter and set the event prescaler to no prescale. The rising edge of CP6 is enabled to perform captures to the circular buffer and increment the event counter. In a real application, you must ensure that the event counter is not initialized in the middle of a series of eight input pulses.

In line 40, the PACT prescale value is set to five, and the command/definition area is enabled.

## 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.

### 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:

$$\begin{aligned} \text{Max Virtual Timer Value} &= \frac{1}{(9600) (4) (10^{-6})} - 2 \\ &= 24 \end{aligned}$$

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.

## 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.
```

### 14.6.3 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.

<u>Register</u>	<u>Before</u>	<u>After No Error</u>	<u>After Error</u>
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

```

```

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

```



## 14.6.7 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.

<u>Register</u>	<u>Before</u>	<u>After</u>	<u>Function</u>
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.

## 14.6.9 Bubble Sort

This routine sorts up to 256 bytes using the bubble sort method. Longer tables can be sorted using the indirect addressing mode.

		<u>Register</u>	<u>Function</u>	
		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	MOV	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

### 14.6.11 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 XLYH1
*   + 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

```

```

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

```

## 14.6.14 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

```



## 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 real-time 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.

## 15.2 The Assembler

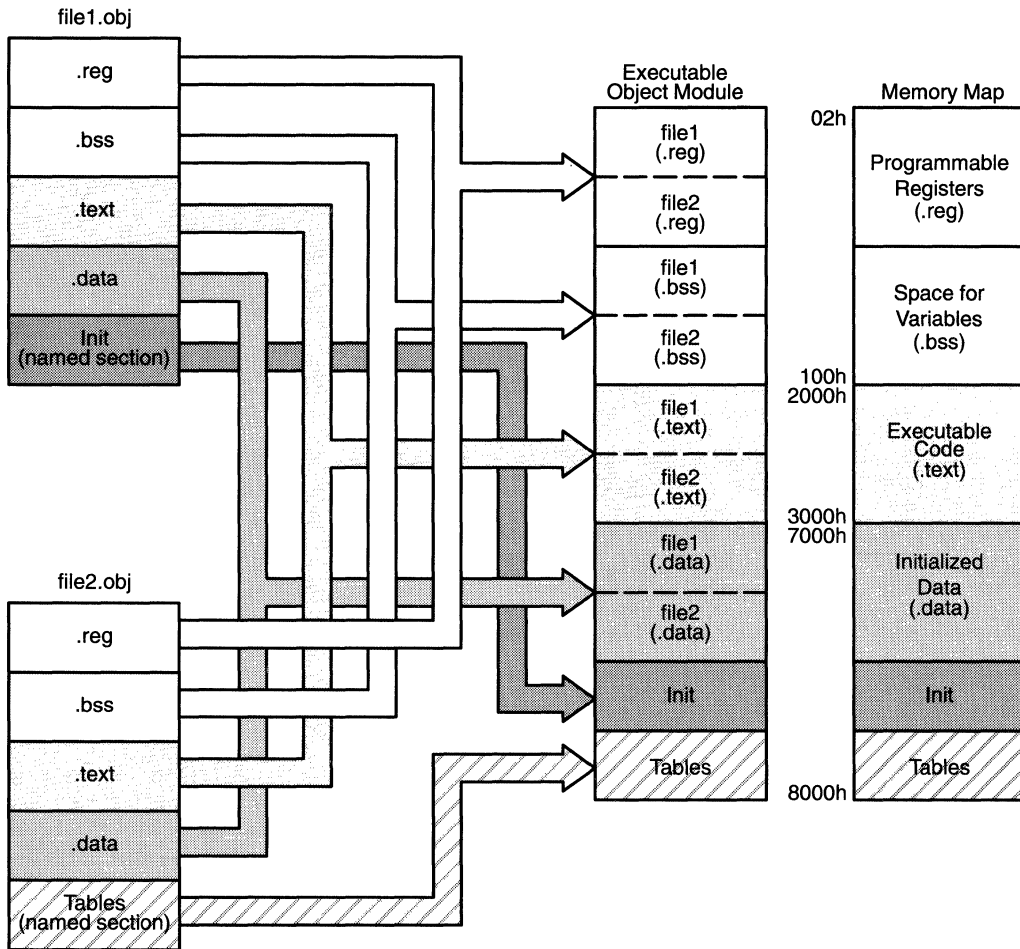
The TMS370 assembler translates assembly language source files into machine language object files. Source files can contain instructions, assembler directives, and macro directives. The assembler directives control various aspects of the assembly process, such as the source listing format, symbol definition, 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.



Figure 15–2. Linker Output Generation



## 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:

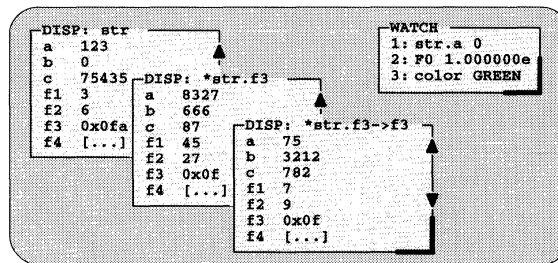
- Conformance with the ANSI C specification
- Highly efficient code—the C compiler incorporates state-of-the-art generic and target-specific optimizations
- ANSI standard runtime-support library
- A C shell program that facilitates one-step translation from C source to executable code
- ROM-able, relocatable, 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

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.

## 15.7 Breakpoint, Trace, and Timing Functions

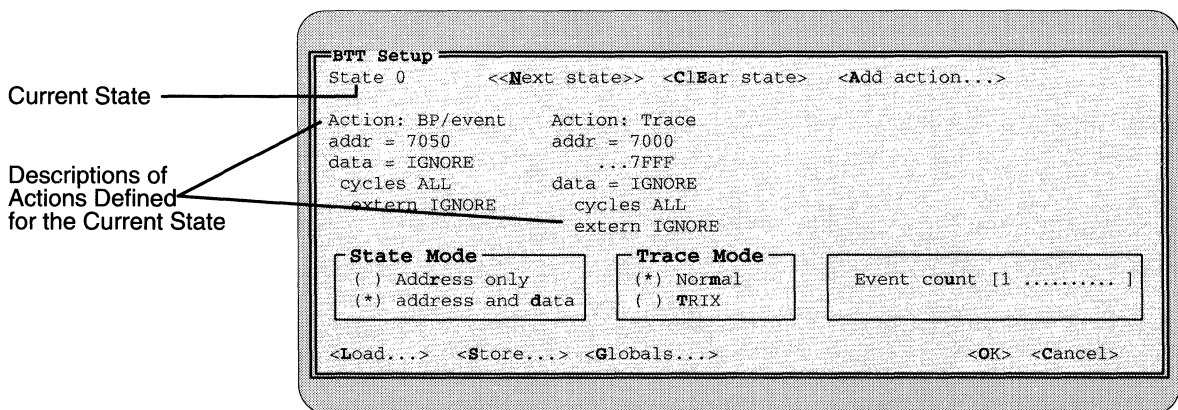
Included with the XDS/22 emulation system is a separate board called the BTT board, which provides breakpoint, trace, and timing features. The BTT monitors the TMS370 CPU; when a preselected pattern of bus activity is detected, the BTT performs an action such as executing a hardware breakpoint or storing 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



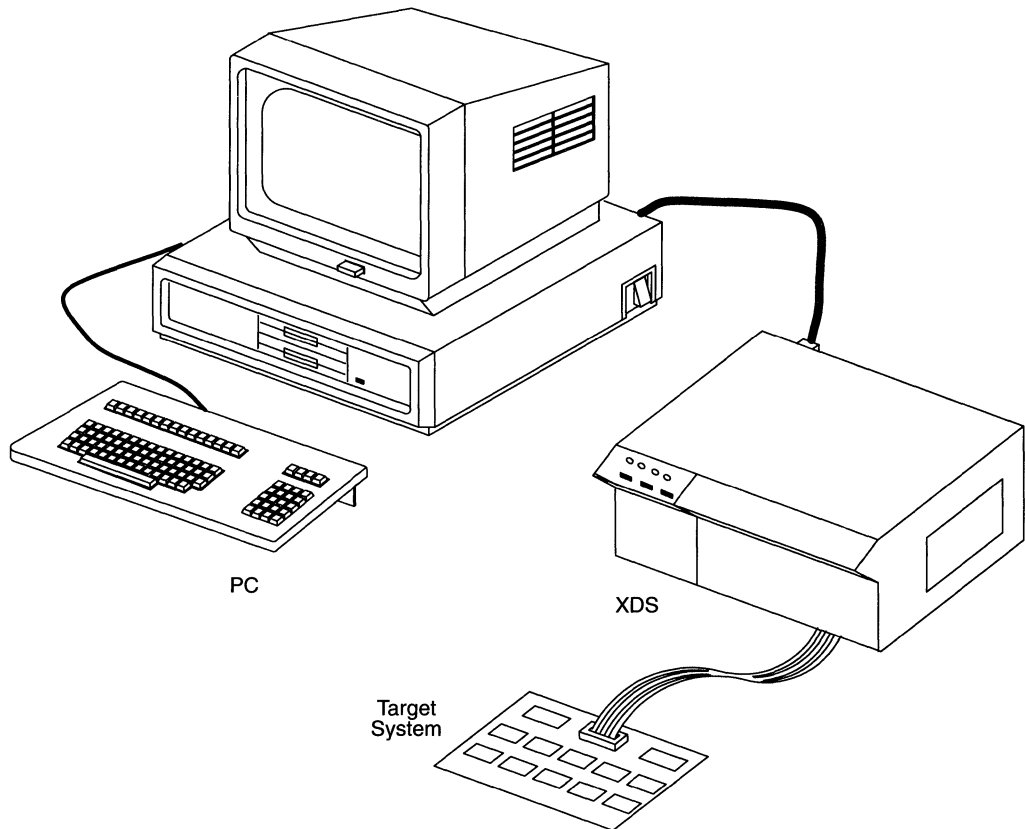
- ❑ **Complete timing analysis.** The BTT supports two timers that you can start and stop on a variety of conditions. The BTT reports the total time for each of these timers; it also reports the average for one of the timers. Additionally, the BTT collects timing information that relates specifically 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.

## 15.8.2 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



## 15.9 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.

## 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

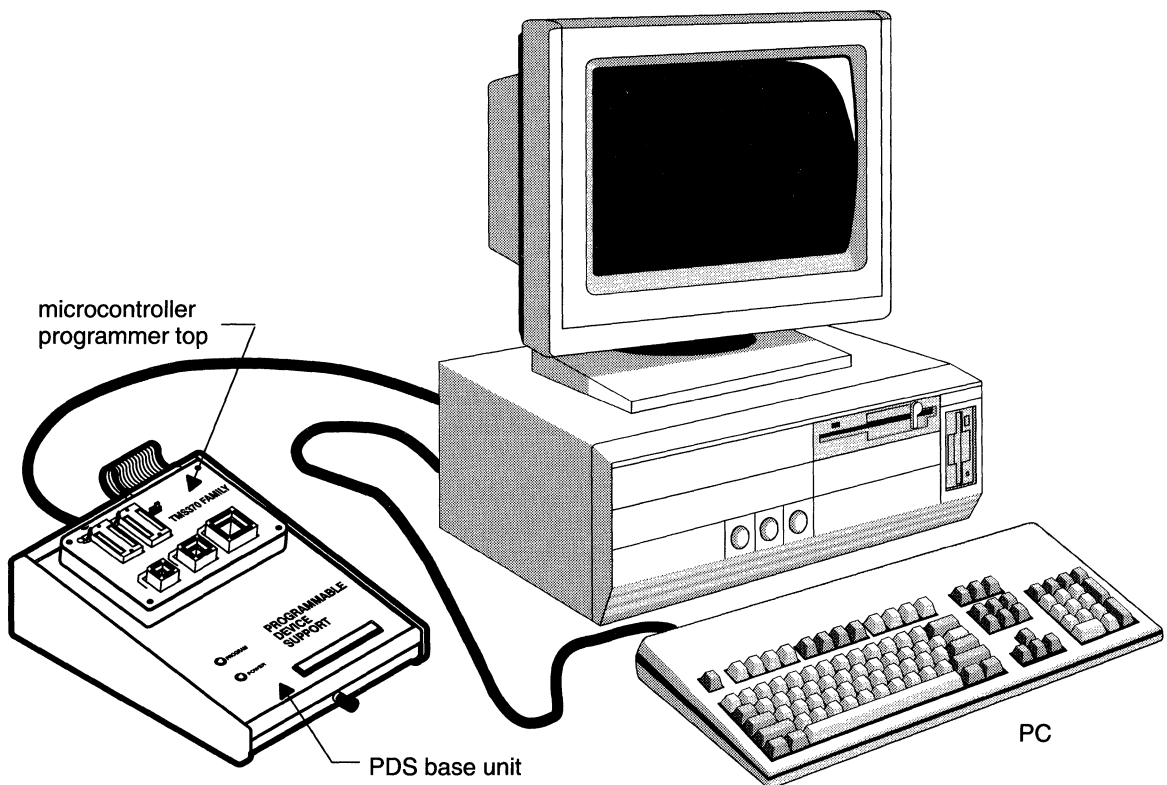


## 15.11 The Microcontroller Programmer

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



## 15.12 The Gang Programmer

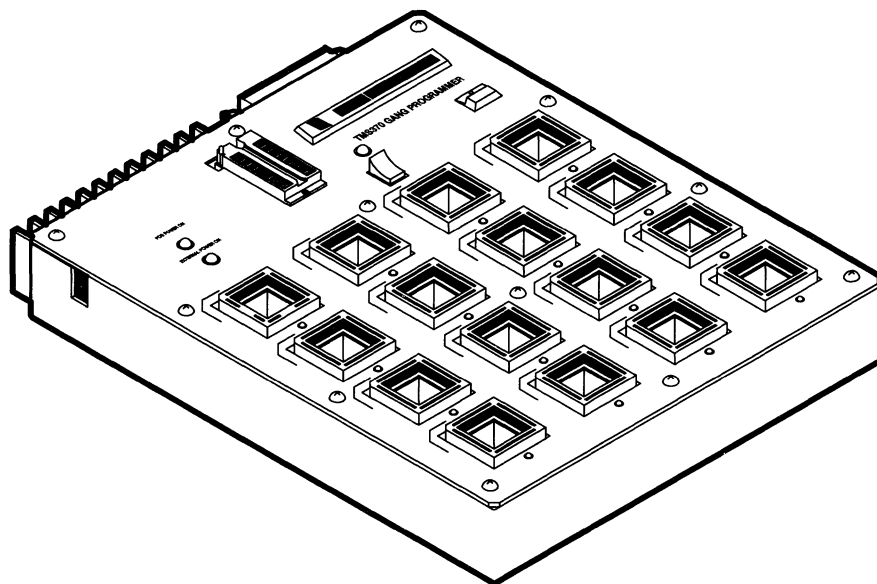
The TMS370 gang programmer is an interactive, menu-driven system that provides programming support for on-chip EEPROM or EPROM of TMS370 microcontrollers in a production environments.

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





## 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	$\overline{\text{EDS}}$	SPC	SPICLK
FE	Final	TXD	SCITXD
IE	Initial	W	Write
PGM	Program	WT	$\overline{\text{WAIT}}$

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

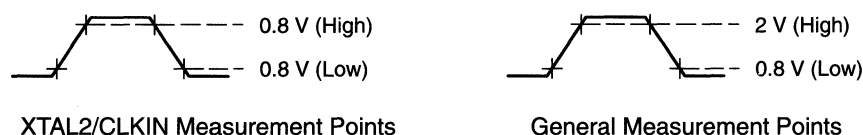
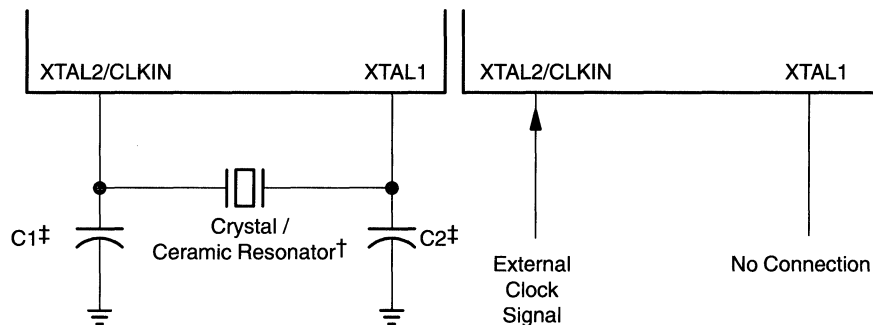


Figure 16–2 illustrates how to connect the crystal/ceramic resonator and the external clock signal. This figure is valid for all TMS370 family devices.

16

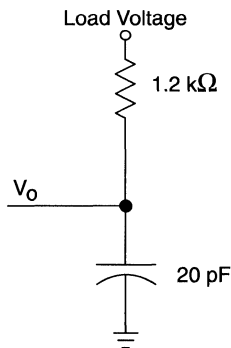
Figure 16–2. Recommended Crystal/Clock Connections



† 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:  $V_O = V_{OH} = 2.4 \text{ V}$ ; Load Voltage = 0 V  
 Case 2:  $V_O = V_{OL} = 0.4 \text{ V}$ ; 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.

## 16.6 TMS370Cx1xA Specifications

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 <sub>CC</sub>	Supply voltage (see Note 1)	4.5	5	5.5	V
V <sub>CC</sub>	RAM data retention supply voltage (see Note 2)	3		5.5	V
V <sub>IL</sub>	Low-level input voltage	All pins except MC		0.8	V
		MC, normal operation		0.3	
V <sub>IH</sub>	High-level input voltage	All pins except MC, XTAL2/CLKIN, and RESET		V <sub>CC</sub>	V
		XTAL2/CLKIN		0.8 V <sub>CC</sub>	
		RESET		0.7 V <sub>CC</sub>	
V <sub>MC</sub>	MC (mode control) voltage	EEPROM write protect override (WPO)		13	V
		EPROM programming voltage (V <sub>PP</sub> )		13	
		Microcomputer		0.3	
T <sub>A</sub>	Operating free-air temperature	A version		85	°C
		L version		70	°C

- Notes:**
- 1) Unless otherwise noted, all voltages are with respect to V<sub>SS</sub>.
  - 2) To guarantee RAM data retention from 3.0 V to 4.5 V, **RESET** must be externally asserted and released only while V<sub>CC</sub> is within the recommended operating range of 4.5 V to 5.5 V.

## 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	$t_w(\text{Cl})$ XTAL2/CLKIN pulse duration‡	20			ns
2	$t_r(\text{Cl})$ XTAL2/CLKIN rise time			30	ns
3	$t_f(\text{Cl})$ 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

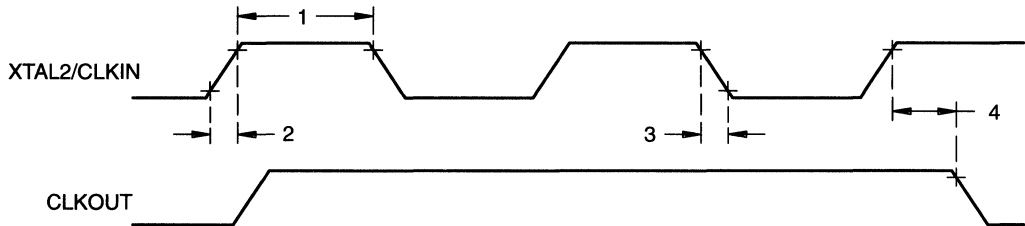


Table 16–9. Switching Characteristics and Timing Requirements†

No.	Parameter	Min	Max	Unit
4	$t_d(\text{CIH-COL})$ Delay time, XTAL2/CLKIN rise to CLKOUT fall		100	ns
5	$t_c$ CLKOUT (system clock) cycle time	200	2000	ns
6	$t_w(\text{COL})$ CLKOUT low pulse duration	$0.5t_c - 20$	$0.5t_c$	ns
7	$t_w(\text{COH})$ CLKOUT high pulse duration	$0.5t_c$	$0.5t_c + 20$	ns

†  $t_c$  = system clock cycle time =  $4/\text{CLKIN}$ .

Figure 16–6. CLKOUT Timing

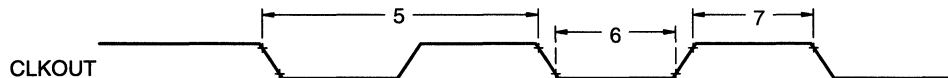




Table 16–11. Electrical Characteristics Over Full Ranges of Recommended Operating Conditions

16

Parameter		Test Conditions	Min	Typ	Max	Unit
V <sub>OL</sub>	Low-level output voltage	I <sub>OL</sub> = 1.4 mA			0.4	V
V <sub>OH</sub>	High-level output voltage	I <sub>OH</sub> = –50 μA	0.9 V <sub>CC</sub>			V
		I <sub>OH</sub> = –2 mA	2.4			
I <sub>I</sub>	Input current	MC	0 V < V <sub>I</sub> ≤ 0.3 V		10	μA
			0.3 V < V <sub>I</sub> ≤ 13 V		650	
		Note 1 12 V ≤ V <sub>I</sub> ≤ 13 V		50	mA	
		I/O pins	0 V ≤ V <sub>I</sub> ≤ V <sub>CC</sub>		± 10	μA
I <sub>OL</sub>	Low-level output current	V <sub>OL</sub> = 0.4 V	1.4			mA
I <sub>OH</sub>	High-level output current	V <sub>OH</sub> = 0.9 V <sub>CC</sub>	–50			μA
		V <sub>OH</sub> = 2.4 V	–2			mA
I <sub>CC</sub>	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 <sub>CC</sub>	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 <sub>CC</sub>	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 <sub>CC</sub>	Supply current (halt mode)	Note 2 XTAL2/CLKIN < 0.2 V		2	30	μA

- Notes:**
- 1) Input current I<sub>pp</sub> 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<sub>CC</sub> – 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.

## 16.8 TMS370Cx3x Specifications

16

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	

- 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.

## 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	$t_w(\text{Cl})$ XTAL2/CLKIN pulse duration‡	20			ns
2	$t_r(\text{Cl})$ XTAL2/CLKIN rise time			30	ns
3	$t_f(\text{Cl})$ 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

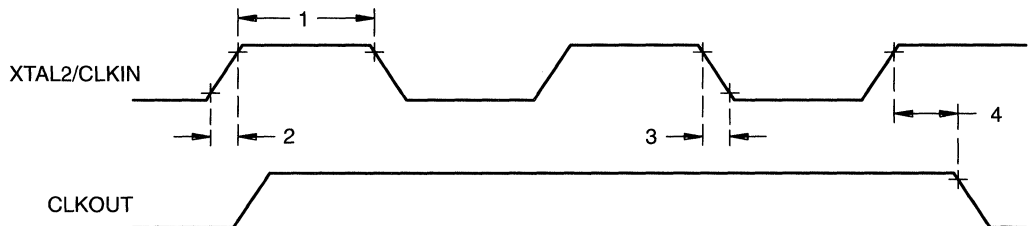


Table 16–17. Switching Characteristics and Timing Requirements†

No.	Parameter	Min	Max	Unit
4	$t_d(\text{CIH-COL})$ Delay time, XTAL2/CLKIN rise to CLKOUT fall		100	ns
5	$t_c$ CLKOUT (system clock) cycle time	200	2000	ns
6	$t_w(\text{COL})$ CLKOUT low pulse duration	$0.5t_c - 20$	$0.5t_c$	ns
7	$t_w(\text{COH})$ CLKOUT high pulse duration	$0.5t_c$	$0.5t_c + 20$	ns

†  $t_c$  = system clock cycle time =  $4/\text{CLKIN}$ .

Figure 16–10. CLKOUT Timing

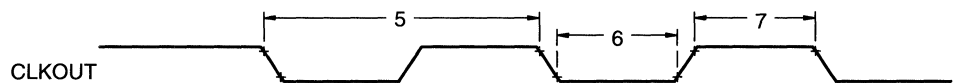


Table 16–19. Electrical Characteristics Over Full Ranges of Recommended Operating Conditions

16

Parameter		Test Conditions	Min	Typ	Max	Unit
$V_{OL}$	Low-level output voltage	$I_{OL} = 1.4 \text{ mA}$			0.4	V
$V_{OH}$	High-level output voltage	$I_{OH} = -50 \mu\text{A}$	0.9 $V_{CC}$			V
		$I_{OH} = -2 \text{ mA}$	2.4			
$I_I$	Input current	MC	$0 \text{ V} < V_I \leq 0.3 \text{ V}$		10	$\mu\text{A}$
			$0.3 \text{ V} < V_I \leq 13 \text{ V}$		650	
			Note 1 $12 \text{ V} \leq V_I \leq 13 \text{ V}$		50	mA
		I/O pins	$0 \text{ V} \leq V_I \leq V_{CC}$		$\pm 10$	$\mu\text{A}$
$I_{OL}$	Low-level output current	$V_{OL} = 0.4 \text{ V}$	1.4			mA
$I_{OH}$	High-level output current	$V_{OH} = 0.9 V_{CC}$	-50			$\mu\text{A}$
		$V_{OH} = 2.4 \text{ V}$	-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	$\mu\text{A}$

- 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  $\leq 0.2 \text{ V}$  or  $\geq V_{CC} - 0.2 \text{ V}$ .
  - 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 =  $.01 \text{ mA} \times (\text{total load capacitance} + \text{crystal capacitance in pF})$ .
  - 4) Maximum operating current for TMS370Cx4xA =  $1.90 (\text{CLKIN}) + 7 \text{ mA}$ .
  - 5) When OSC POWER bit = 0, maximum standby current for TMS370Cx4xA =  $0.75 (\text{CLKIN}) + 2 \text{ mA}$ .
  - 6) When OSC POWER bit = 1, maximum standby current for TMS370Cx4xA =  $0.56 (\text{CLKIN}) + 1.9 \text{ mA}$ . Bit is valid only from 2 MHz to 12 MHz.

## 16.10 TMS370Cx5xA Specifications

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  $V_{CC} < 4.5$  V 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 16–23. 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)	$I_{OL} = 1.4 \text{ mA}$			0.4	V	
$V_{OH}$	High-level output voltage	$I_{OH} = -50 \mu\text{A}$		$0.9 V_{CC}$		V	
		$I_{OH} = -2 \text{ mA}$		2.4			
$I_I$	Input current	MC	$0 \text{ V} < V_I \leq 0.3 \text{ V}$			10	$\mu\text{A}$
			$0.3 \text{ V} < V_I < V_{CC} - 0.3$			50	
			$V_{CC} - 0.3 \text{ V} \leq V_I \leq V_{CC} + 0.3 \text{ V}$			10	
			$V_{CC} + 0.3 \text{ V} < V_I \leq 13 \text{ V}$			650	
		Note 2; $12 \text{ V} \leq V_I \leq 13 \text{ V}$			50	mA	
	I/O pins	$0 \text{ V} \leq V_I \leq V_{CC}$			$\pm 10$	$\mu\text{A}$	
$I_{OL}$	Low-level output current (see Note 1)	$V_{OL} = 0.4 \text{ V}$	1.4			mA	
$I_{OH}$	High-level output current	$V_{OH} = 0.9 V_{CC}$	-50			$\mu\text{A}$	
		$V_{OH} = 2.4 \text{ V}$	-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
					35	56	
		TMS370Cx50A, TMS370Cx52A	Notes 3 and 4; CLKIN = 12 MHz		20	30	mA
					25	36	
		TMS370Cx50A, TMS370Cx52A	Notes 3 and 4; CLKIN = 2 MHz		5	11	mA
					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	$\mu\text{A}$	

- 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 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  $\leq 0.2 \text{ V}$  or  $\geq V_{CC} - 0.2 \text{ V}$ .
  - 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 =  $.01 \text{ mA} \times (\text{total load capacitance} + \text{crystal capacitance in pF})$ .
  - 5) Maximum operating current for TMS370Cx50A and TMS370Cx52A =  $1.9 (\text{CLKIN}) + 7 \text{ mA}$ . Maximum operating current for TMS370Cx56A =  $2.5 (\text{CLKIN}) + 5.8 \text{ mA}$ .
  - 6) When OSC POWER bit = 0, maximum standby current for TMS370Cx5xA =  $0.75 (\text{CLKIN}) + 2 \text{ mA}$ .
  - 7) When OSC POWER bit = 1, maximum standby current for TMS370Cx5xA =  $0.56 (\text{CLKIN}) + 1.9 \text{ mA}$ . Bit is valid only from 2 MHz to 12 MHz.

Table 16–25. Switching Characteristics and Timing Requirements for External Read and Write†

16

No.	Parameter	Min	Max	Unit
5	$t_c$ CLKOUT (system clock) cycle time	200	2000	ns
6	$t_w(\text{COL})$ CLKOUT low pulse duration	$0.5 t_c - 25$	$0.5 t_c$	ns
7	$t_w(\text{COH})$ CLKOUT high pulse duration	$0.5 t_c$	$0.5 t_c + 20$	ns
8	$t_d(\text{COL-A})$ Delay time, CLKOUT low to address R/W, and OCF valid		$0.25 t_c + 75$	ns
9	$t_v(\text{A})$ Address valid to $\overline{\text{EDS}}$ , $\overline{\text{CSE1}}$ , $\overline{\text{CSE2}}$ , $\overline{\text{CSH1}}$ , $\overline{\text{CSH2}}$ , $\overline{\text{CSH3}}$ , and CSPF low	$0.5 t_c - 90$		ns
10	$t_{su}(\text{D})$ Write data set-up time to $\overline{\text{EDS}}$ high	$0.75 t_c - 80^\ddagger$		ns
11	$t_h(\text{EH-A})$ Address, R/W, and $\overline{\text{OCF}}$ hold time from $\overline{\text{EDS}}$ , $\overline{\text{CSE1}}$ , $\overline{\text{CSE2}}$ , $\overline{\text{CSH1}}$ , $\overline{\text{CSH2}}$ , $\overline{\text{SH3}}$ , and CSPF high	$0.5 t_c - 60$		ns
12	$t_h(\text{EH-D})\text{W}$ Write data hold time from $\overline{\text{EDS}}$ high	$0.75 t_c + 15$		ns
13	$t_d(\text{DZ-EL})$ Delay time, data bus high impedance to $\overline{\text{EDS}}$ low (read cycle)	$0.25 t_c - 35$		ns
14	$t_d(\text{EH-D})$ Delay time, $\overline{\text{EDS}}$ high to data bus enable (read cycle)	$1.25 t_c - 40$		ns
15	$t_d(\text{EL-DV})$ Delay time, $\overline{\text{EDS}}$ low to read data valid		$t_c - 95^\ddagger$	ns
16	$t_h(\text{EH-D})\text{R}$ Read data hold time from $\overline{\text{EDS}}$ high	0		ns
17	$t_{su}(\text{WT-COH})$ $\overline{\text{WAIT}}$ set-up time to CLKOUT high	$0.25 t_c + 70^\S$		ns
18	$t_h(\text{COH-WT})$ $\overline{\text{WAIT}}$ hold time from CLKOUT high	0		ns
19	$t_d(\text{ED-WTV})$ Delay time, $\overline{\text{EDS}}$ low to $\overline{\text{WAIT}}$ valid		$0.5 t_c - 60$	ns
20	$t_w$ Pulse duration, $\overline{\text{EDS}}$ , $\overline{\text{CSE1}}$ , $\overline{\text{CSE2}}$ , $\overline{\text{CSH1}}$ , $\overline{\text{CSH2}}$ , $\overline{\text{CSH3}}$ , and CSPF low	$t_c - 80^\ddagger$	$t_c + 40^\ddagger$	ns
21	$t_d(\text{AV-DV})\text{R}$ Delay time, address valid to read data valid		$1.5 t_c - 115^\ddagger$	ns
22	$t_d(\text{AV-WTV})$ Delay time, address valid to $\overline{\text{WAIT}}$ valid		$t_c - 115$	ns
23	$t_d(\text{AV-EH})$ Delay time, address valid to $\overline{\text{EDS}}$ high (end of write)	$1.5 t_c - 85^\ddagger$		ns

†  $t_c$  = system clock cycle time =  $4/\text{CLKIN}$ .‡ 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  $\overline{\text{WAIT}}$  input may assume a “don’t care” condition until the third cycle of the access. The  $\overline{\text{WAIT}}$  signal must be synchronized with the high pulse of the CLKOUT signal while still conforming to the minimum set-up time.

## 16.11 SCI Timings

16

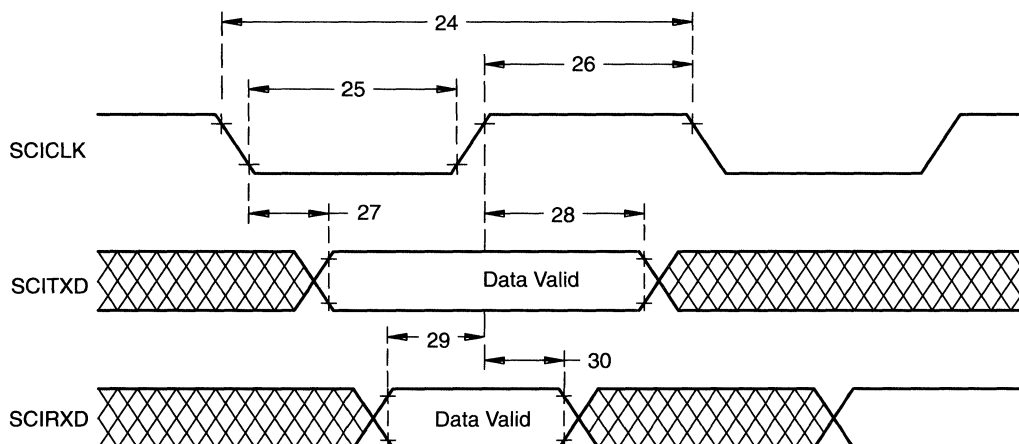
This section contains timing tables and figures for devices that have the serial communications interface (SCI) module.

Table 16–26. SCI Isosynchronous Mode Timing Characteristics and Requirements for Internal Clock<sup>†</sup>

No.	Parameter	Min	Max	Unit
24	$t_c(\text{SCC})$ SCICLK cycle time	$2 t_c$	$131,072 t_c$	ns
25	$t_w(\text{SCCL})$ SCICLK low pulse duration	$t_c - 45$	$0.5 t_c(\text{SCC}) + 45$	ns
26	$t_w(\text{SCCH})$ SCICLK high pulse duration	$t_c - 45$	$0.5 t_c(\text{SCC}) + 45$	ns
27	$t_d(\text{SCCL-TXDV})$ Delay time, SCITXD valid after SCICLK low	- 50	60	ns
28	$t_v(\text{SCCH-TXD})$ SCITXD data valid after SCICLK high	$t_w(\text{SCCH}) - 50$		ns
29	$t_{su}(\text{RXD-SCCH})$ SCIRXD set-up time to SCICLK high	$0.25 t_c + 145$		ns
30	$t_v(\text{SCCH-RXD})$ SCIRXD data valid after SCICLK high	0		ns

<sup>†</sup>  $t_c$  = system clock cycle time =  $4/\text{CLKIN}$ .

Figure 16–16. SCI Isosynchronous Mode Timing Diagram for Internal Clock





## 16.12 SPI Timings

16

This section contains timing tables and figures for devices that have the serial peripheral interface (SPI) module.

**Note:**

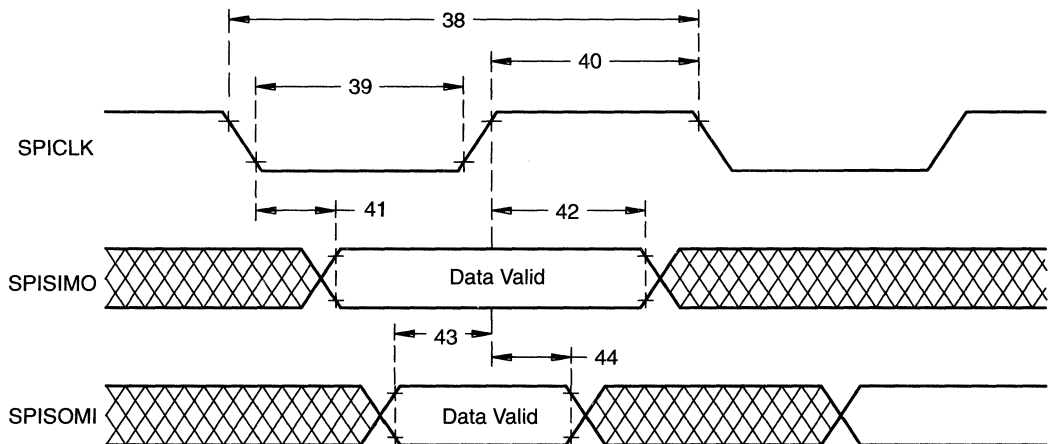
Some SPI electrical specifications and timings differ for TMS370Cxxx devices. Refer to subsection A.6.2, page A-6.

Table 16–28. SPI Master External Timing Characteristics and Requirements†

No.	Parameter	Min	Max	Unit
38	$t_c(\text{SPC})$ SPICLK cycle time	$2 t_c$	$256 t_c$	ns
39	$t_w(\text{SPCL})$ SPICLK low pulse duration	$t_c - 45$	$0.5 t_c(\text{SPC}) + 45$	ns
40	$t_w(\text{SPCH})$ SPICLK high pulse duration	$t_c - 55$	$0.5 t_c(\text{SPC}) + 45$	ns
41	$t_d(\text{SPCL-SIMOV})$ Delay time, SPISIMO valid after SPICLK low (polarity = 1)	-65	50	ns
42	$t_v(\text{SPCH-SIMO})$ SPISIMO data valid after SPICLK high (polarity = 1)	$t_w(\text{SPCH}) - 50$		ns
43	$t_{su}(\text{SOMI-SPCH})$ SPISOMI set-up time to SPICLK high (polarity = 1)	$0.25 t_c + 150$		ns
44	$t_v(\text{SPCH-SOMI})$ 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.

## 16.13 A/D Converter Specifications

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 <sup>†</sup>	2.5	$V_{CC3}$	$V_{CC3} + 0.1$	V
Analog input for conversion	$V_{SS3}$		$V_{ref}$	V

<sup>†</sup>  $V_{ref}$  must be stable, within  $\pm 1/2$  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 <sup>†</sup>	$V_{CC3} = 5.5$ V, $V_{ref} = 5.1$ V			$\pm 1.5$	LSB
Differential/integral linearity error <sup>†‡</sup>	$V_{CC3} = 5.5$ V, $V_{ref} = 5.1$ V			$\pm 0.9$	LSB
$I_{CC3}$ Analog supply current	Converting			2	mA
	Nonconverting			5	$\mu$ A
$I_I$ Input current, AN0–AN7	$0$ V $\leq V_I \leq 5.5$ V			2	$\mu$ A
$V_{ref}$ input charge current				1	mA
$Z_{ref}$ Source impedance of $V_{ref}$	$XTAL2/CLKIN \leq 12$ MHz			24	k $\Omega$
	$12$ MHz $< XTAL2/CLKIN \leq 20$ MHz			10	k $\Omega$

<sup>†</sup> 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.

<sup>‡</sup> Excluding quantization error of  $1/2$  LSB.





## 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.

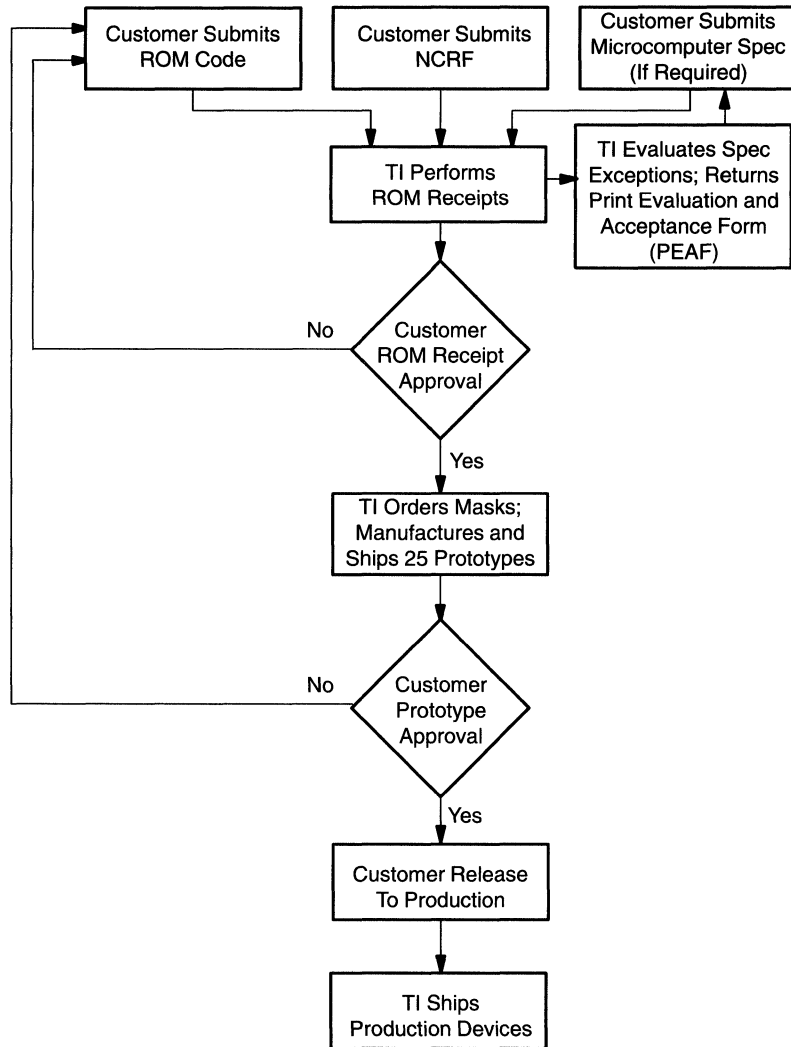
- 1) **Customer-required information.** For TI to accept the receipt of your ROM algorithm, you must follow these steps:
  - a) Complete and submit a new code release form (NCRF—available from a TI Field Sales Office and shown in Example 17-1, page 17-5) describing the custom features of the device (for 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.

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.

## 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.

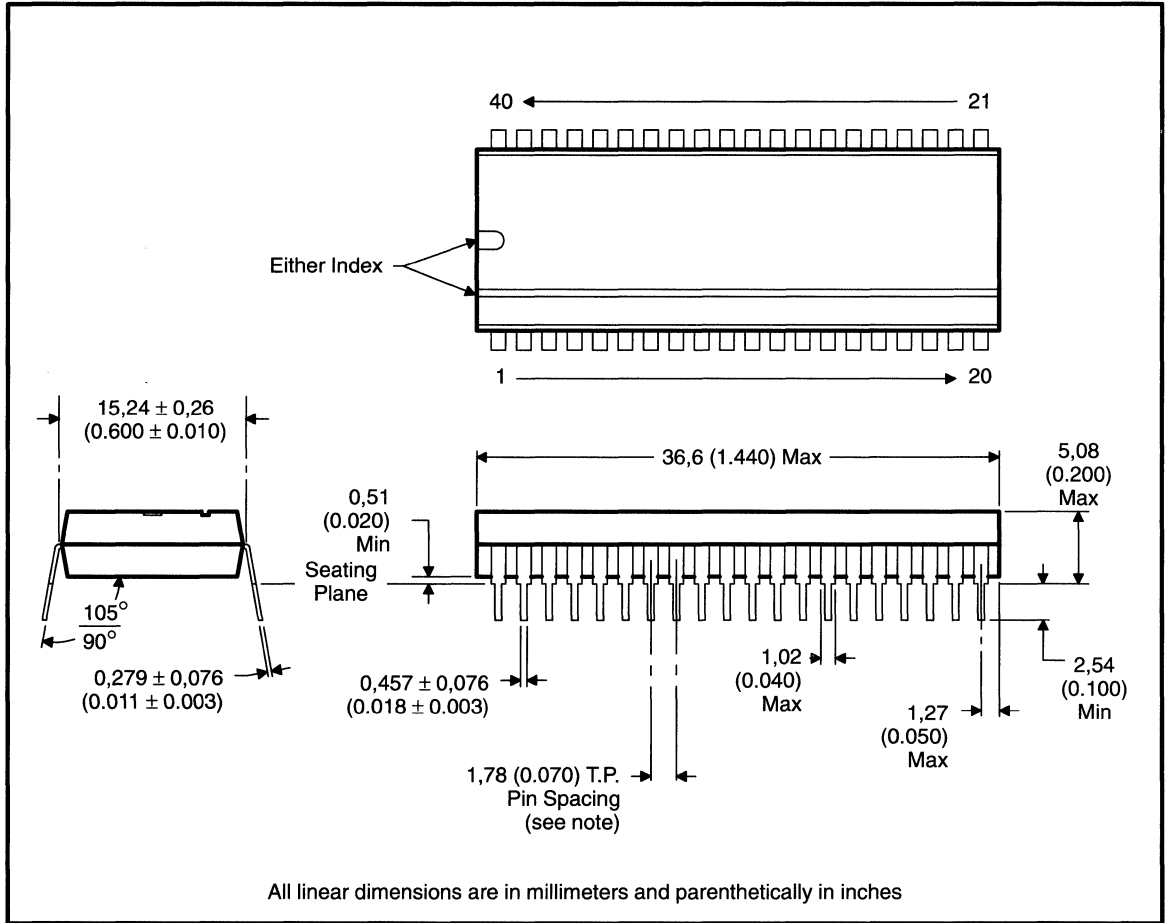
*Table 17–1. Package Types*

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 17-3. 40-Pin Plastic Dual-Inline Shrink Package (N2 Suffix)

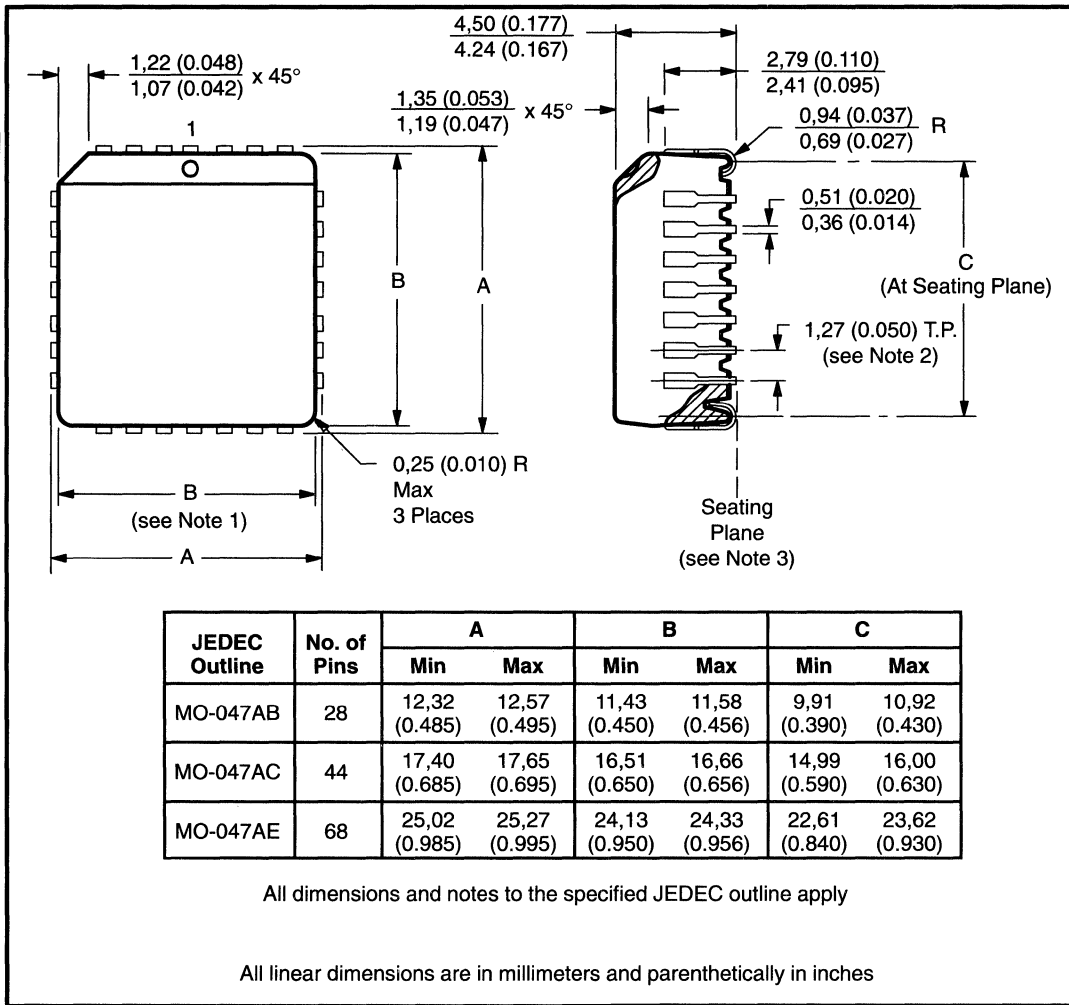
17



**Note:** Each pin centerline is located within 0,26 (0.010) of its true longitudinal position.

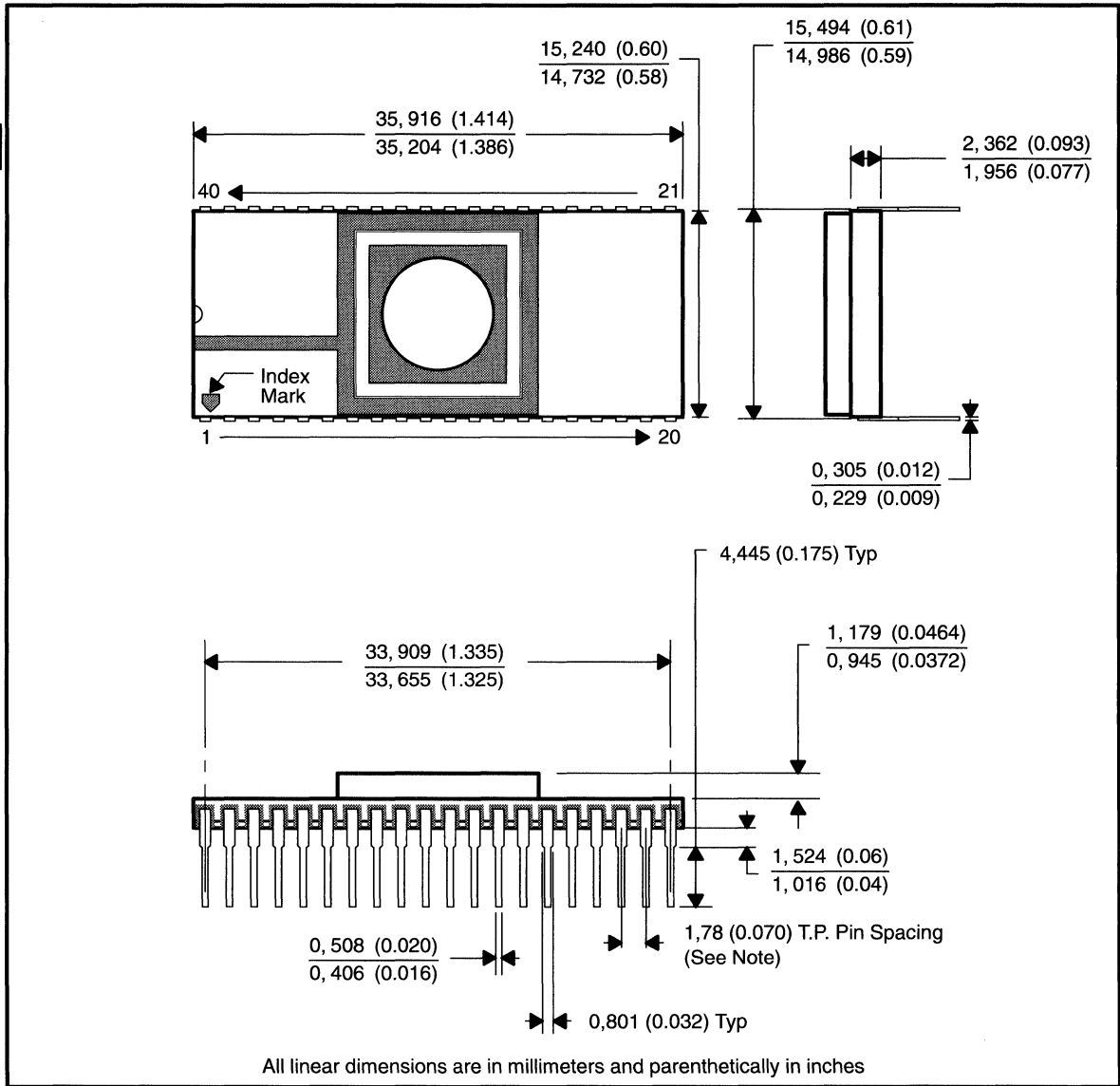


Figure 17-5. Plastic-Leaded Chip Carrier Package (FN Suffix)



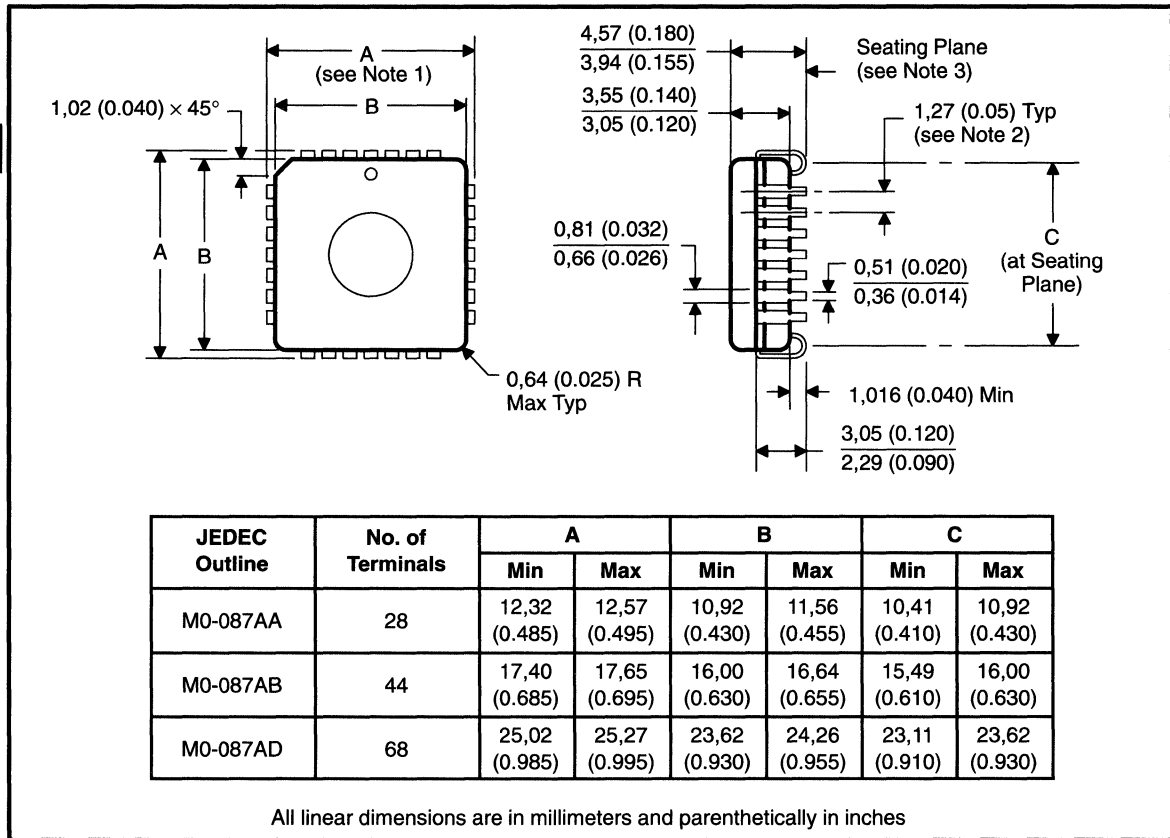
- Notes:**
- 1) Centerline of center pin on each side is within 0,10 (0.004) of package centerline as determined by dimension B.
  - 2) Location of each pin is within 0,127 (0.005) of true position with respect to center pin on each side.
  - 3) The lead contact points are planar within 0,10 (0.004).

Figure 17-7. 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 17-9. Ceramic-Leaded Chip Carrier Package (FZ Suffix)



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).

### 17.3.2 Support Device Prefix Designators

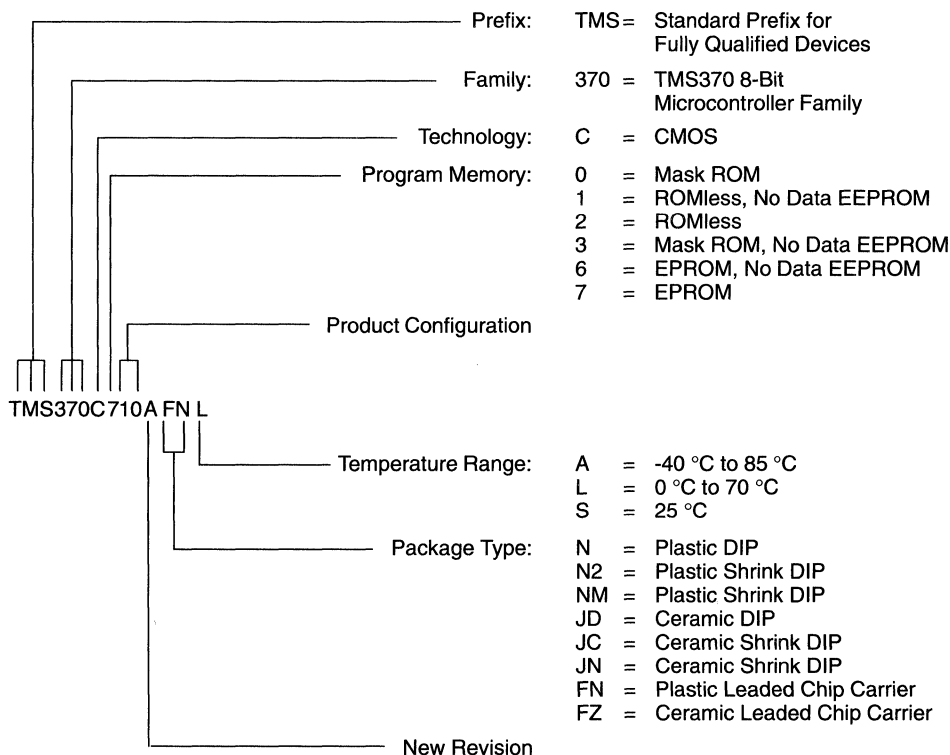
The SE prefix designation is given to the system evaluator devices that are used for prototyping purposes. This designation applies only to the prototype members of the TMS370 family (SE370C710, SE370C722, SE370C732, SE370C742, SE370C756, and SE370C758). The SE devices are shipped against the following disclaimer:

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



- **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

Line 1:	(a)		(c) R1X00XXXN
Line 2:			(f) WAYYWW
Line 3:			(i) 12345678
Line 4:	(g) ©1986TI		(k) Philippines

- **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)

Line 1:	(a)		(d) TMS370CXXXN
Line 2:			(f) WAYYWW
Line 3:			(g) ©1988TI
Line 4:	(i) 12345678		(k) Philippines

- **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

Line 1:	(e) SE370C7XXFZX
Line 2:	(j) 980 (f) WAYYWW
Line 3:	(i) 12345678
Line 4:	(g) ©1988TI
Line 5:	(k) USA

### 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 <sub>AC</sub> (target cables sold separately)
TMDS3762281	TMS370 XDS/22 Emulator—220 V <sub>AC</sub> (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.



## 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 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



### 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

### A.4 V<sub>CC1</sub> and V<sub>CC2</sub> Pins

The V<sub>CC1</sub> and V<sub>CC2</sub> 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<sub>SS1</sub> and V<sub>SS2</sub>) 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.

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 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

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
$t_d(\text{SCCL-TXDV})$	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
$t_w(\text{SPCH})$	SPICLK high pulse duration	$t_c - 55$	$0.5 t_c(\text{SPC}) + 45$	ns	$t_c - 45$	$0.5 t_c(\text{SPC}) + 45$
$t_d(\text{SPCL-SIMOV})$	Delay time, SPISIMO valid after SPICLK low (polarity = 1)	-65	50	ns	-50	50
$t_d(\text{SPCL-SOMIV})_S$	Delay time, SPISOMI valid after SPICLK low (polarity = 1)		$3.25 t_c + 130$	ns		$3.25 t_c + 125$

---

A

## 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.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.5 Peripheral File Frame 4: PACT Control Registers

Designation	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)
RXBUF	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)
TXBUF	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.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	T2 Counter MSbyte							Bit 8
T2CNTR	1061h	P061	T2 Counter LSbyte							Bit 0
T2C	1062h	P062	Compare Register MSbyte							Bit 8
T2C	1063h	P063	Compare Register LSbyte							Bit 0
T2CC	1064h	P064	Capture/Compare Register MSbyte							Bit 8
T2CC	1065h	P065	Capture/Compare Register LSbyte							Bit 0
T2IC	1066h	P066	Capture Register 2 MSbyte							Bit 8
T2IC	1067h	P067	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)
Dual Compare Mode										
T2CTL2	106Bh	P06B	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)
Dual Compare Mode										
T2CTL3	106Ch	P06C	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)	—	—	—	—	—	—

For more information about these registers and control bits, refer to Section 8.8.

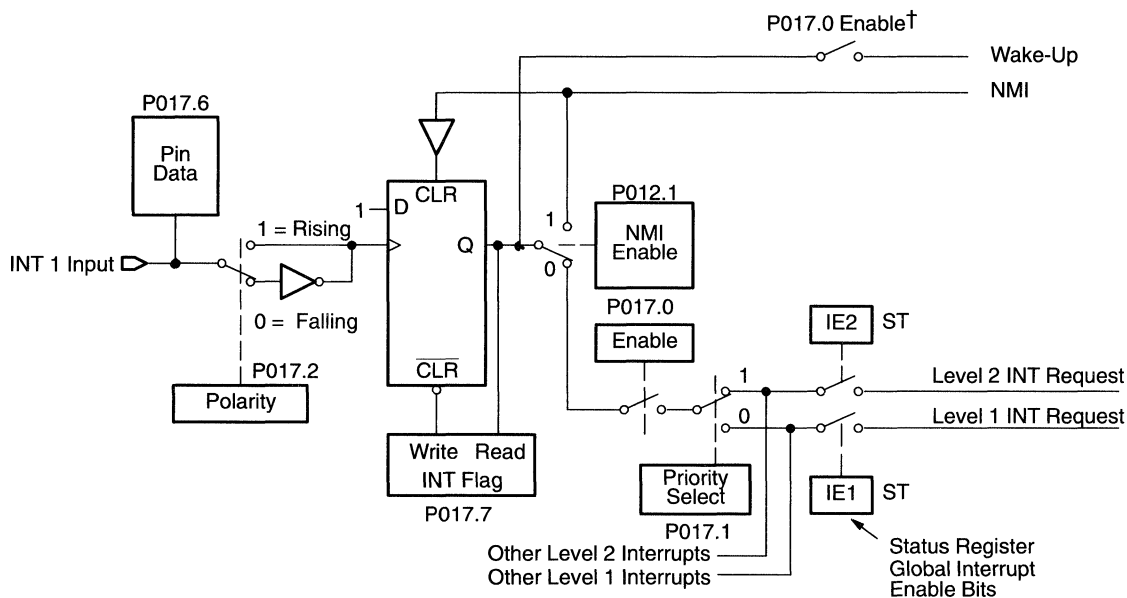




## C.1 Interrupts

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

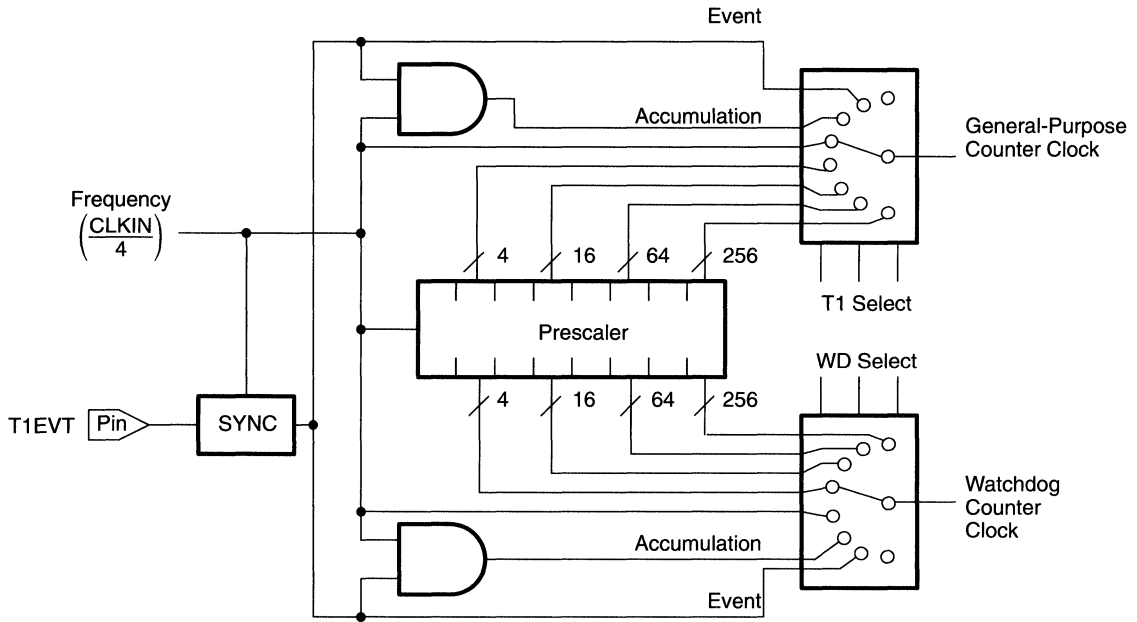


† This bit is ignored if you are using the hard watchdog option.

## 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.

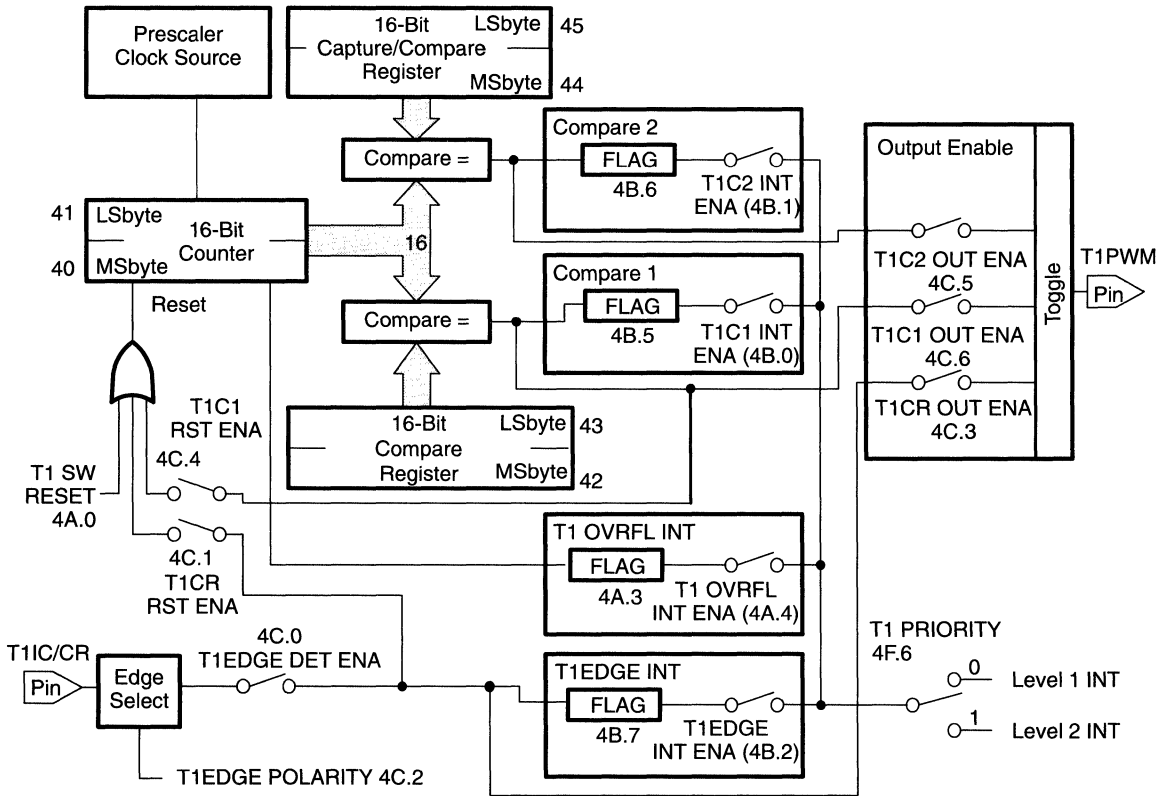
The timer 1 module has two operating modes:

- 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

C



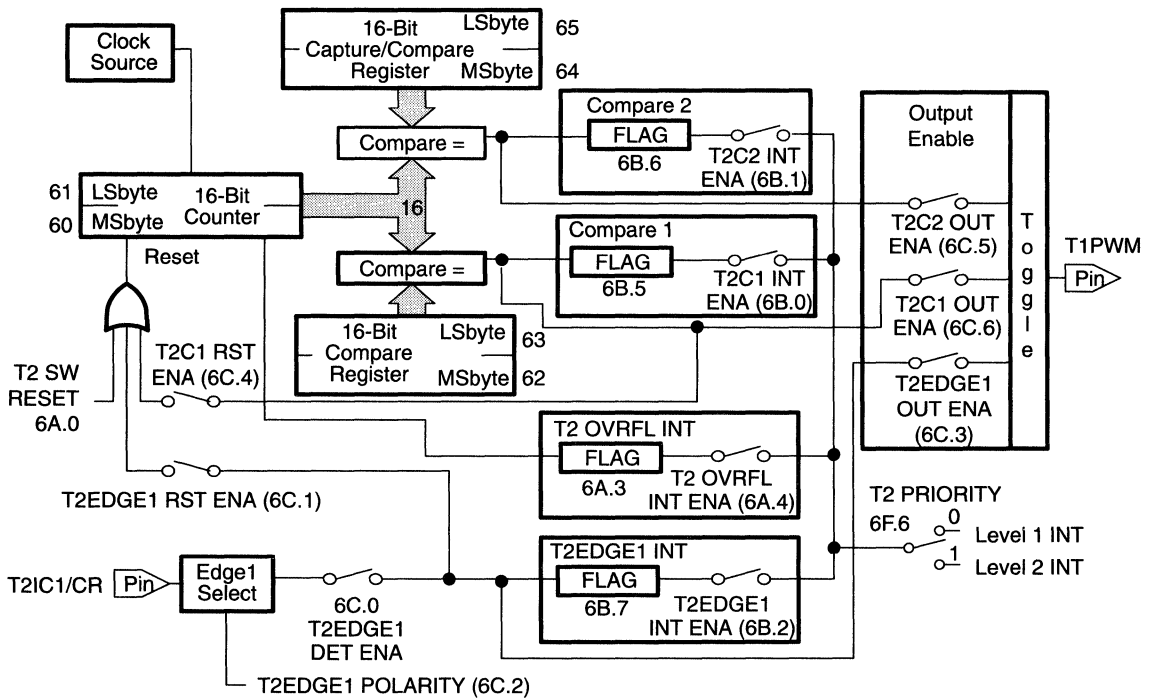
### C.3 Timer 2 Module

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.

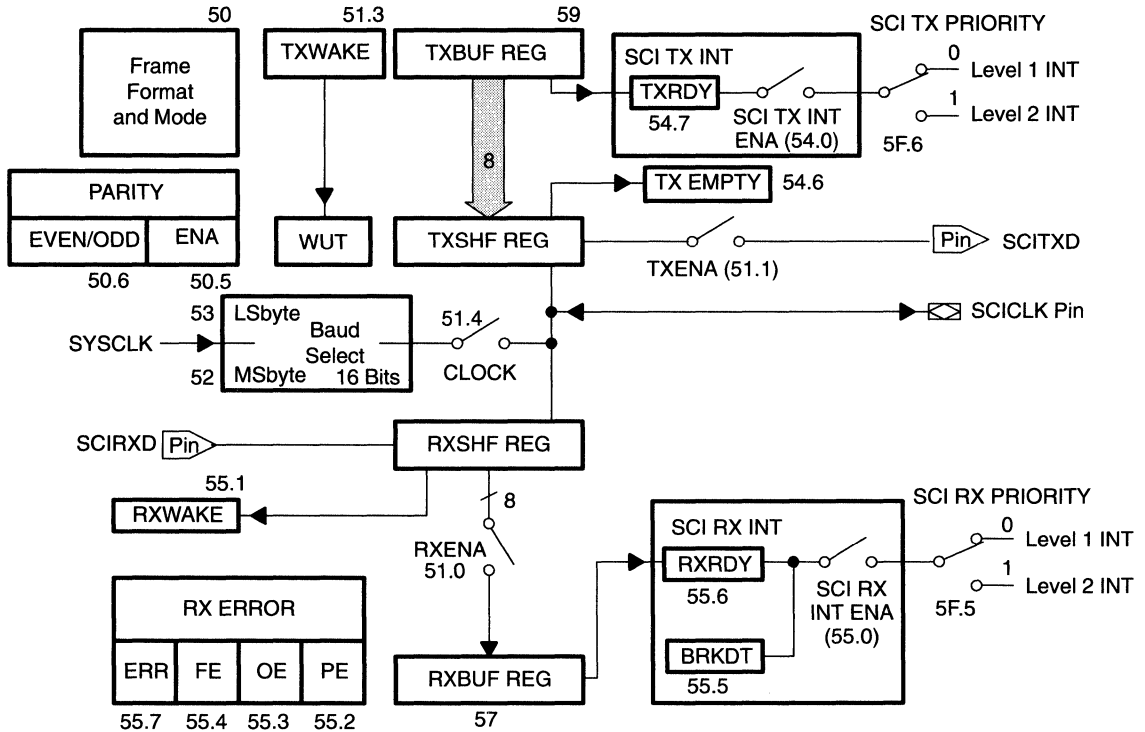
**C** Figure C-7. Timer 2: Dual Compare Mode Block Diagram



### 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







D



Table E-1. TMS370 Family Opcode/Instruction Map

		MSN																
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
L S N	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 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 inst2 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 B 1/8	COMPL B 1/8	COMPL Rn 2/6	TRAP 4 1/14	PUSH ST 1/8	





E

## 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.

## 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

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	<<	Ar	Aw																
59	ADC #n,B	OC	<<	O1	<<	Br	Bw																
79	ADC #n,Rd	OC	<<	O1	<<	O2	<<	Rr	Rw														
69	ADC B,A	OC	<<	PO	<<	Br	IC	Ar	Aw														
19	ADC Rs,A	OC	<<	O1	<<	Rr	Ar	Aw															
39	ADC Rs,B	OC	<<	O1	<<	Rr	Br	Bw															
49	ADC Rs,Rd	OC	<<	O1	<<	Rr	O2	<<	Rr	Rw													
28	ADD #n,A	OC	<<	O1	<<	Ar	Aw																
58	ADD #n,B	OC	<<	O1	<<	Br	Bw																
78	ADD #n,Rd	OC	<<	O1	<<	O2	<<	Rr	Rw														
68	ADD B,A	OC	<<	PO	<<	Br	IC	Ar	Aw														
18	ADD Rs,A	OC	<<	O1	<<	Rr	Ar	Aw															
38	ADD Rs,B	OC	<<	O1	<<	Rr	Br	Bw															
48	ADD Rs,Rd	OC	<<	O1	<<	Rr	O2	<<	Rr	Rw													
23	AND #n,A	OC	<<	O1	<<	Ar	Aw																
53	AND #n,B	OC	<<	O1	<<	Br	Bw																
A3	AND #n,Pd	OC	<<	O1	<<	O2	<<	Pr	<<	Pw	<<												
73	AND #n,Rd	OC	<<	O1	<<	O2	<<	Rr	Rw														
83	AND A,Pd	OC	<<	O1	<<	Ar	Pr	<<	Pw	<<													
63	AND B,A	OC	<<	PO	<<	Br	IC	Ar	Aw														
93	AND B,Pd	OC	<<	O1	<<	Br	Pr	<<	Pw	<<													
13	AND Rs,A	OC	<<	O1	<<	Rr	Ar	Aw															
33	AND Rs,B	OC	<<	O1	<<	Rr	Br	Bw															
43	AND Rs,Rd	OC	<<	O1	<<	Rr	O2	<<	Rr	Rw													
9C	BR @Rp	OC	<<	O1	<<	Rr	Rr	IC	IC														
8C	BR lab	OC	<<	O1	<<	O2	<<	IC	IC	IC													
AC	BR lab(B)	OC	<<	O1	<<	O2	<<	Br	IC	IC	IC												
F4 EC	BR n(Rp)	PF	<<	OC	<<	IC	IC	O1	<<	O2	<<	Rr	Rr	IC	IC	IC	IC						
26	BTJO #n,A,ra	OC	<<	O1	<<	Ar	IC	O2	<<	RJ	RJ												
56	BTJO #n,B,ra	OC	<<	O1	<<	Br	IC	O2	<<	RJ	RJ												
A6	BTJO #n,Pd,ra	OC	<<	O1	<<	O2	<<	Pr	<<	IC	O3	<<	RJ	RJ									
76	BTJO #n,Rd,ra	OC	<<	O1	<<	O2	<<	Rr	IC	O3	<<	RJ	RJ										
86	BTJO A,Pd,ra	OC	<<	O1	<<	Ar	Pr	<<	IC	O2	<<	RJ	RJ										
66	BTJO B,A,ra	OC	<<	PO	<<	Br	IC	Ar	IC	O1	<<	RJ	RJ										
96	BTJO B,Pd,ra	OC	<<	O1	<<	Br	Pr	<<	IC	O2	<<	RJ	RJ										
16	BTJO Rs,A,ra	OC	<<	O1	<<	Rr	Ar	IC	O2	<<	RJ	RJ											
36	BTJO Rs,B,ra	OC	<<	O1	<<	Rr	Br	IC	O2	<<	RJ	RJ											
46	BTJO Rs,Rd,ra	OC	<<	O1	<<	Rr	O2	<<	Rr	IC	O3	<<	RJ	RJ									
27	BTJZ #n,A,ra	OC	<<	O1	<<	Ar	IC	O2	<<	RJ	RJ												
57	BTJZ #n,B,ra	OC	<<	O1	<<	Br	IC	O2	<<	RJ	RJ												
A7	BTJZ #n,Pd,ra	OC	<<	O1	<<	O2	<<	Pr	<<	IC	O3	<<	RJ	RJ									
77	BTJZ #n,Rd,ra	OC	<<	O1	<<	O2	<<	Rr	IC	O3	<<	RJ	RJ										
87	BTJZ A,Pd,ra	OC	<<	O1	<<	Ar	Pr	<<	IC	O2	<<	RJ	RJ										
67	BTJZ B,A,ra	OC	<<	PO	<<	Br	IC	Ar	IC	O1	<<	RJ	RJ										
97	BTJZ B,Pd,ra	OC	<<	O1	<<	Br	Pr	<<	IC	O2	<<	RJ	RJ										
17	BTJZ Rs,A,ra	OC	<<	O1	<<	Rr	Ar	IC	O2	<<	RJ	RJ											
37	BTJZ Rs,B,ra	OC	<<	O1	<<	Rr	Br	IC	O2	<<	RJ	RJ											
47	BTJZ Rs,Rd,ra	OC	<<	O1	<<	Rr	O2	<<	Rr	IC	O3	<<	RJ	RJ									
9E	CALL @Rp	OC	<<	O1	<<	Rr	Rr	IC	SH	IC	SH	IC	IC										
8E	CALL lab	OC	<<	O1	<<	O2	<<	IC	IC	SH	IC	SH	IC	IC									
AE	CALL lab(B)	OC	<<	O1	<<	O2	<<	Br	IC	IC	IC	SH	IC	SH	IC	IC							
F4 EE	CALL n(Rp)	PF	<<	OC	<<	IC	IC	O1	<<	O2	<<	Rr	Rr	IC	IC	IC	SH	IC	SH	IC	IC		

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	.	.	.	.	.	.	.	.
77	JBIT0	Rname,ra	OC	<<	O1	<<	O2	<<	Rr	IC	O3	<<	RJ	RJ	.	.	.	.	.	.	.	.	.
A6	JBIT1	Pname,ra	OC	<<	O1	<<	O2	<<	Pr	<<	IC	O3	<<	RJ	RJ	.	.	.	.	.	.	.	.
76	JBIT1	Rname,ra	OC	<<	O1	<<	O2	<<	Rr	IC	O3	<<	RJ	RJ	.	.	.	.	.	.	.	.	.
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	Rr	Rr	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	<<	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	.	.	.	.	.	.	.	.	.	.	.
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	<<	.	.	.	.	.	.	.	.	.	.	.
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	IC	.	.	.	.	.	.	.
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
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	Aw												
C7	SWAP B	OC	<<	PO	<<	Br	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	.	.	.	.	.	.	.	.	.	.	.	.	.

- 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.

Figure G–1. Pinouts for TMS370Cx1x Devices (Top View)

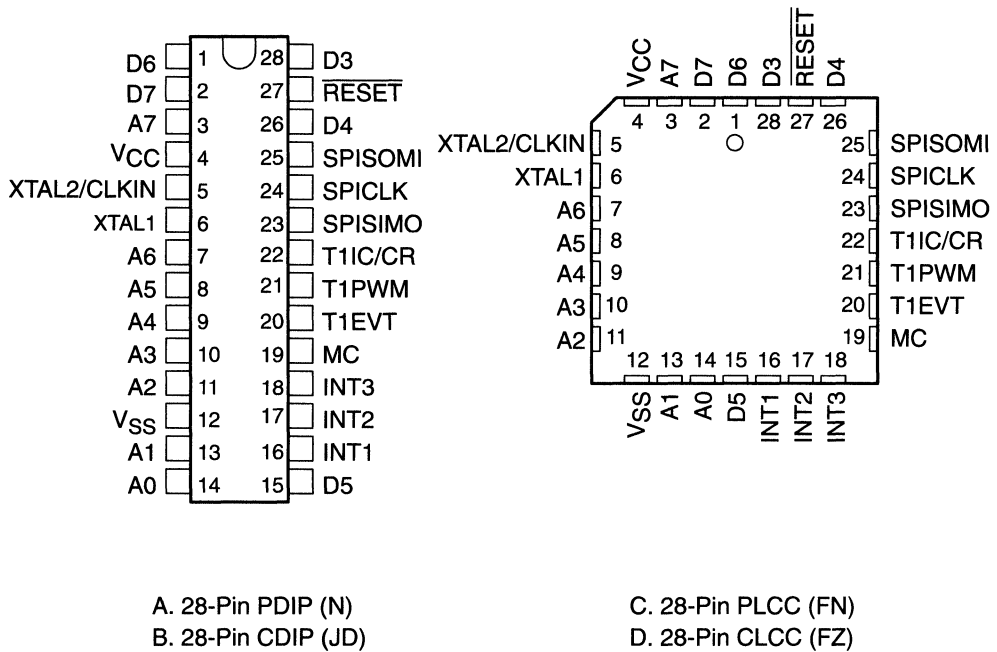


Figure G–2. Pinouts for TMS370Cx2x Devices (Top View)

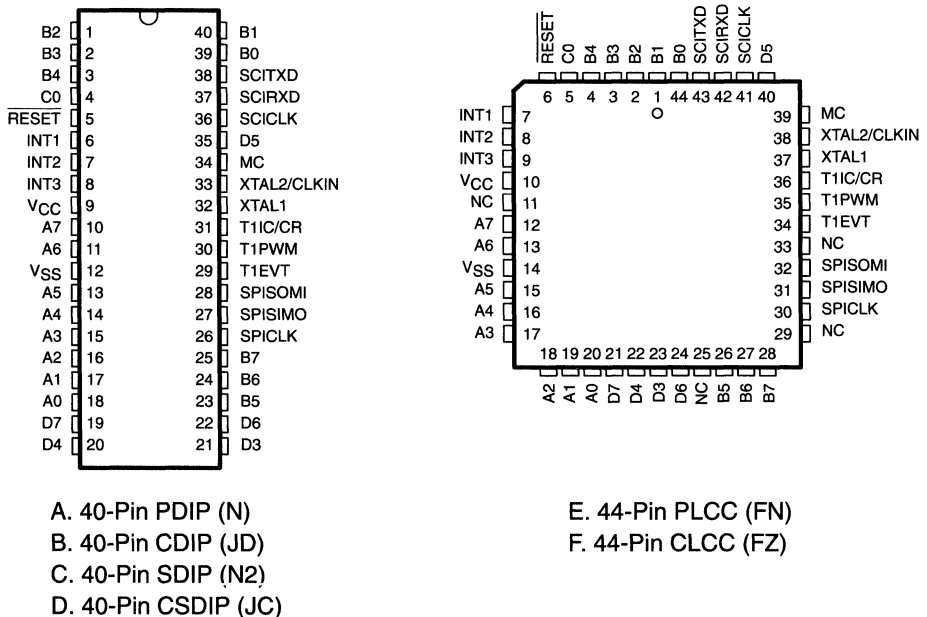
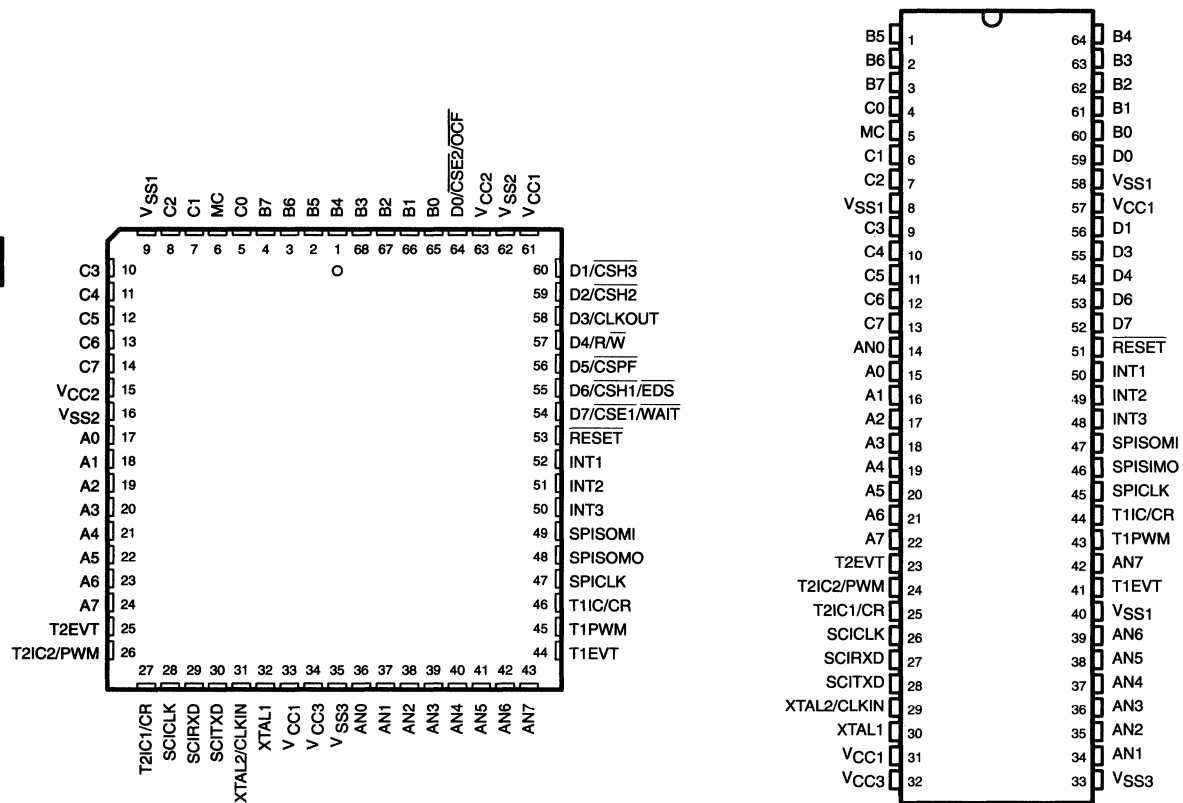


Figure G–5. Pinouts for TMS370Cx5x Devices (Top View)

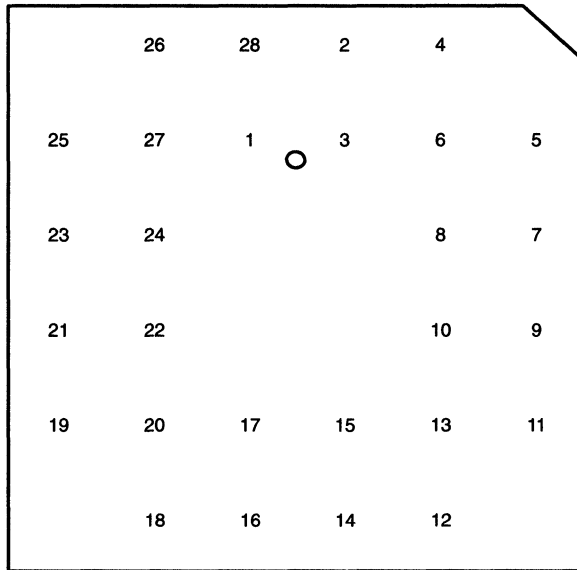


- A. 68-Pin PLCC (FN)
- B. 68-Pin CLCC (FZ)

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



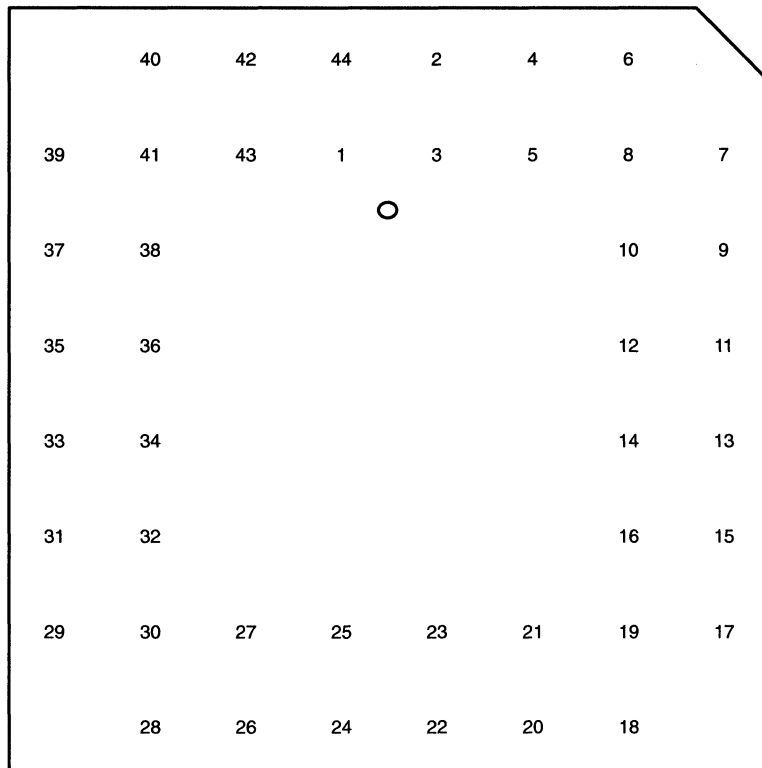
Figure H-1. 28-Pin PGA Pinout



BOTTOM VIEW

H

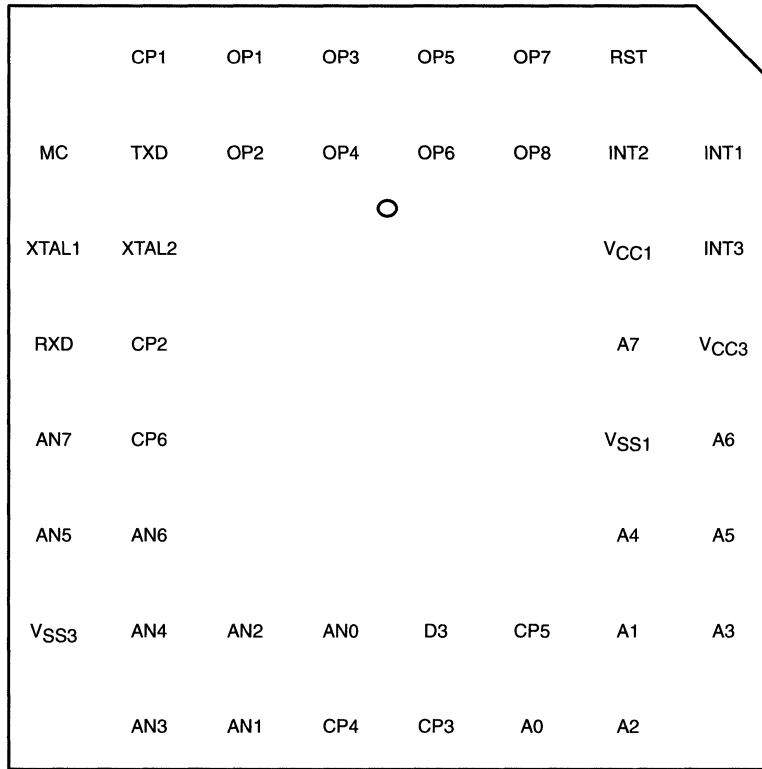
Figure H-3.44-Pin PGA Pinout



BOTTOM VIEW

H

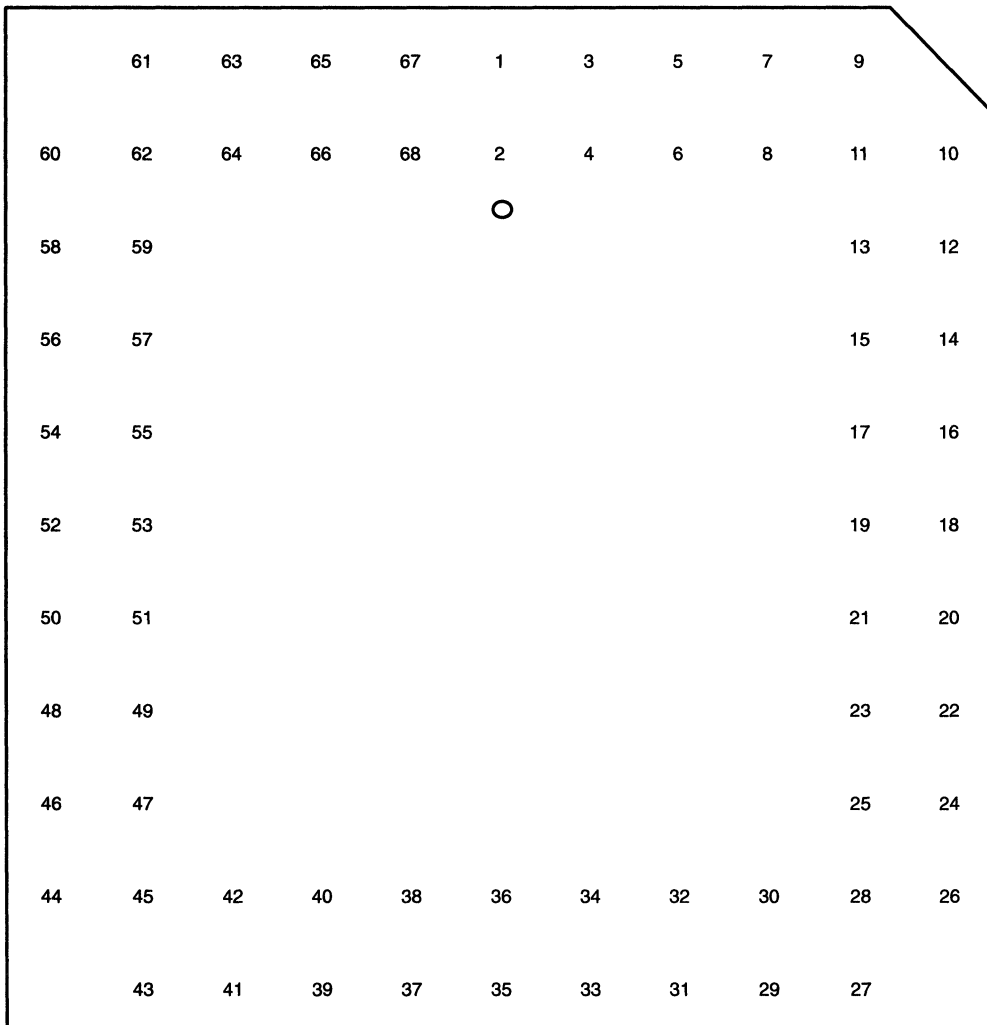
Figure H-5. TMS370Cx3x Device PGA Pinout



BOTTOM VIEW

H

Figure H-7.68-Pin PGA Pinout



BOTTOM VIEW

H

H

## 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.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.

```

;
; 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  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    .MACROperiod,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

```

```

;           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

```

**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.

**J****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.

**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.

**J**

**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.

**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.

**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.

## 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  $V_{REF}$ .

**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.

**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  $\overline{RESET}$  signal returns high, then the processor enters the microcomputer mode. If the MC pin is high when the  $\overline{RESET}$  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.

**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.

- ADC (add with carry) instruction 13-27
  - setting the V bit of ST 3-5
- ADCTL (analog control) register 11-10 to 11-11
  - AD INPUT SELECT0–2 bits 11-10
  - CONVERT START bit 11-6, 11-11
  - REF VOLT SELECT0–2 bits 11-10
  - SAMPLE START bit 11-6, 11-11
- ADD instruction 13-28
  - setting the V bit of ST 3-5
- ADDATA (analog conversion data) register 11-12
  - DATA0–7 bits 11-12
- addition instructions
  - ADC 13-27
  - ADD 13-28
  - DAC 13-41
  - INC 13-51
  - INCW 13-52
- address bit mode. *See* SCI, multiprocessor communications, address bit mode
- address memory
  - during microcomputer mode 3-18
- ADDRESS/IDLE WUP bit (SCICCR register) 9-7, 9-20
- addressing modes 13-4 to 13-16
  - additional 13-16
  - definition J-1
  - extended 13-11 to 13-16
    - absolute* 13-11
      - direct 13-12
      - indexed 13-13
      - indirect 13-14
      - offset indirect 13-15
    - definition* J-4
    - overview* 13-4
    - relative* 13-11
      - direct 13-12
      - indexed 13-14
      - indirect 13-15
      - offset indirect 13-16
  - general 13-5 to 13-10
    - definition* J-4
    - immediate* 13-8
    - implied* 13-5
    - overview* 13-4
    - peripheral* 13-7
    - program counter relative* 13-9
    - register* 13-6
    - stack pointer relative* 13-10
- ADENA (analog port E input enable) register 11-13
  - PORT E INPUT ENA0–7 bits 11-13
- ADIN (analog port E data input) register 11-13
  - PORT E DATA AN0–7 bits 11-13
- ADPRI (analog interrupt priority) register 11-14
  - AD ESPEN bit 11-14
  - AD PRIORITY bit 11-5, 11-14
  - AD STEST bit 11-14
- ADSTAT (analog status and interrupt) register 11-12
  - AD INT ENA bit 11-5, 11-12
  - AD INT FLAG bit 11-5, 11-12
  - AD READY bit 11-12
- amplitude detector 4-4
  - See also* oscillator
- AN0–AN7 pins 11-4
- analog pins. *See* AN0–AN7 pins
- analog-to-digital converter. *See* A/D converter module
- AND (logical AND) instruction 13-29
- ANSI C
  - definition J-1
- AP bit (DEECTL register) 6-5
- applications 1-3
  - sample 14-1 to 14-44
    - See also design aids, sample routines*
- architecture
  - overview 1-5 to 1-8, 3-2 to 3-3
  - summary of components (by device) 1-8 to 1-10
- archiver 15-2, 15-7
  - definition J-1
- ASCII table D-1
- assembler 15-4
  - definition J-1
  - ordering information 17-19
  - symbolic debugging 15-4
- assembly language
  - definition J-1
  - instructions
    - See also individual instruction name listings*
    - addressing modes* 13-4 to 13-16
      - See also* addressing modes
    - alphabetical summary* 13-27 to 13-86
    - bus activity table* F-4 to F-10
      - bus cycles F-5
      - internal cycles F-5
    - cross-reference*
      - instruction/opcode F-2 to F-3
      - opcode/instruction 13-24 to 13-25, E-1 to E-3



# C

- C bit (ST register) 3-6
  - See also* carry bit (C)
- C compiler. *See* compiler
- C programming language
  - definition J-2
- CALL instruction 13-33
- CALLR (call relative) instruction 13-34
- capture register
  - See also* timer 2, registers, capture definition J-2
- capture/compare mode. *See* timer 1, operating modes
- carry bit (C) 3-6
  - clearing using CLRC 3-6, 13-36
  - setting using SETC 3-6, 13-79
- category. *See* TMS370 family, categories
- CDEND (command/definition area end) register 12-40
  - CMD/DEF AREA END BIT 2–6 bits 12-40
- CDFLAGS (command/definition entry flags) register 12-45
  - CMD/DEF INT 0–7 FLAG bits 12-45
- CDIP 17-6, 17-11
- CDSTART (command/definition area start) register 12-39
  - CMD/DEF AREA INT ENA bit 12-39
  - CMD/DEF AREA START BIT 2–5 bits 12-39
- CDT370 15-2, 15-18
  - capabilities 15-18
  - definition J-3
  - hardware 15-18
  - ordering information 17-20
- character sets D-1
- chip select
  - definition J-2
- circuit
  - constructing with expansion memory 4-5
  - reset 5-16 to 5-17
    - with low-voltage detection* 5-17 to 5-18
- circular buffer. *See* PACT, circular buffer
- CLCC 17-6, 17-14
- CLKOUT signal
  - definition 4-20
- CLOCK bit (SCICTL register) 9-10, 9-14, 9-22 to 9-23
- CLOCK POLARITY bit (SPICCR register) 10-10, 10-15
- clock prescaler. *See* timer 1, prescaler
- CLR (clear) instruction 13-35
- CLRC (clear the carry bit) instruction 13-36
  - clearing the C bit of ST 3-6
- CMD/DEF AREA ENA bit (PACTSCR register) 12-37
- CMD/DEF AREA END BIT 2–6 bits (CDEND register) 12-40
- CMD/DEF AREA INT ENA bit (CDSTART register) 12-39
- CMD/DEF AREA START BIT 2–5 bits (CDSTART register) 12-39
- CMD/DEF INT 0–7 FLAG bits (CDFLAGS register) 12-45
- CMP (compare) instruction 13-37 to 13-38
  - setting the V bit of ST 3-5
  - status bit values 13-38
- CMPBIT (complement bit) instruction 13-39
- code conversion utility 15-2, 15-7
  - definition J-2
- COFF 15-4
  - definition J-3
- COLD START bit (SCCR0 register) 4-12
  - as source of reset 5-15
  - determining source of reset 4-4
- command/definition area
  - See also* PACT, command/definition area file format 12-7
- comment
  - definition J-3
- compact development tool. *See* CDT370
- compare instructions
  - CMP 13-37 to 13-38
  - CMPBIT 13-39
  - COMPL 13-40
- compare register
  - See also* timer 1, registers; timer 2, registers definition J-3
- compiler 15-2, 15-8
  - definition J-2
  - key features 15-8
  - optimizations 15-8
  - ordering information 17-19
- COMPL (2s-complement) instruction 13-40
- condition flags (C, N, Z, V) 3-6

- CPU (continued)
    - overview 3-2 to 3-3
    - registers 3-4 to 3-7
      - PC 3-7
      - SP 3-4
      - ST 3-5 to 3-6
    - value during low-power modes 4-7
  - CPU STEST bit (SCCR2 register) 4-14
  - crystal/clock connection 16-4
  - CSDIP 17-6, 17-12, 17-13
  - $\overline{\text{CSE1}}$  signal
    - activating 3-18
    - definition 4-19
  - $\overline{\text{CSE2}}$  signal
    - activating 3-18
    - definition 4-19
  - $\overline{\text{CSH1}}$  signal
    - activating 3-18
    - definition 4-19
  - $\overline{\text{CSH2}}$  signal
    - activating 3-18
    - definition 4-19
  - $\overline{\text{CSH3}}$  signal
    - activating 3-18
    - definition 4-20
  - $\overline{\text{CSPF}}$  signal
    - activating 3-19
    - definition 4-19
- D**
- DAC (decimal add with carry) instruction 13-41
  - data
    - EEPROM 3-12 to 3-13, 6-2 to 6-9
      - access time 3-12, 6-2
      - control registers 6-3 to 6-5
      - DEECTL 3-13, 6-2, 6-4 to 6-5, 6-6 to 6-9, B-2
      - WPR 6-2, 6-3 to 6-4, 6-6 to 6-9
    - definition 3-2
    - description 1-5
    - entering a low-power mode 6-7
    - finding voltage status 6-5, 6-9
    - loading a data byte 6-6
      - example 6-8
    - preventing data corruption 6-9
  - data, EEPROM (continued)
    - programming 3-13, 6-2, 6-6 to 6-9, 14-14 to 14-16
      - calculating loop delay 6-8
      - devices with multiple 256-byte arrays 6-5
      - devices with single 256-byte array 6-5
      - example 6-7 to 6-9
      - initiating 6-6, 6-8
    - reading 6-2
      - after voltage is stabilized 6-9
    - write protection 3-12, 6-2, 6-3 to 6-4
      - overriding 6-3
      - programming example 6-4
  - format. See SCI, data formats
  - memory
    - during microcomputer mode 3-18
  - DATA BUS signal, definition 4-20
  - DATA0–7 bits (ADDATA register) 11-12
  - debugger 15-2, 15-9 to 15-11
    - basic display 15-9
    - breakpoint, trace, and timing functions 15-12 to 15-14
      - definition J-2
      - key features 15-12 to 15-14
    - definition J-3
    - key features 15-10 to 15-12
    - symbolic debugging 15-4
      - definition J-9
  - DEC (decrement) instruction 13-42
    - setting the V bit of ST 3-5
  - decimal designator (P) 3-11
  - DEECTL (data EEPROM control) register 3-13, 6-2, 6-4 to 6-5, 6-6 to 6-9
    - AP bit 6-5
    - BUSY bit 6-5, 6-9
    - EXE bit 6-4, 6-6, 6-8
    - W1W0 bit 6-4, 6-6, 6-8
  - DEFTIM OVRFL INT ENA bit (PACTSCR register) 12-38
  - DEFTIM OVRFL INT FLAG bit (PACTSCR register) 12-38
  - design aids 14-1 to 14-44
    - sample routines 14-30 to 14-44
      - 32-bit multiplication 14-40
      - BCD string addition 14-36
      - BCD-to-binary conversion 14-35
      - binary-to-BCD conversion 14-34
      - bubble sort 14-38
      - clear RAM 14-31

**E**

- EPROM
  - See also* microcomputer interface example; reprogrammable EPROMs
  - definition J-4
  - description 1-6
  - program 3-14, 6-10 to 6-14
  - See also program, EPROM*
  - timing requirements 16-5
- EVEN/ODD PARITY bit (SCICCR register) 9-21
- event counter 7-15, 8-12
  - See also* PACT, event counter
- EVENT COUNTER SW RESET bit (CPPRE register) 12-52
- EXE bit (DEECTL register) 6-4, 6-6, 6-8
- EXE bit (EPCTL register) 6-11, 6-12
- expansion, memory. *See* microcomputer mode, with external expansion
- extended addressing mode 13-11 to 13-16
  - See also* addressing modes, extended definition J-4
- external interrupts. *See* interrupts

**F**

- family device. *See* TMS370 family
- FAST MODE SELECT bit (PACTSCR register) 12-37
  - calculating the divide rate 12-6
- FE bit (RXCTL register) 9-27
- frame. *See* peripheral file, frames
- frequency detector 4-4
  - See also* oscillator
- function A
  - expansion signals 3-20
  - external memory expansion 3-16, 3-18 to 3-20
  - signal definitions 4-19 to 4-20
- function B
  - accessing external memory chips 3-16, 3-20 to 3-21
  - during reset in microprocessor mode 3-22
  - expansion signals 3-21
  - signal definitions 4-19 to 4-20
- function mnemonic
  - definition 13-2

**G**

- general addressing mode 13-5 to 13-10
  - See also* addressing modes, general definition J-4

**H**

- halt mode. *See* low-power modes, halt
- HALT/STANDBY bit (SCCR2 register) 4-6 to 4-7, 4-15, 7-24
- hard watchdog. *See* watchdog timer, hard watchdog
- hardware interrupts. *See* interrupts, external
- hardware timer. *See* PACT, timers, hardware timer
- hexadecimal designator (P0) 3-11

**I**

- I/O configuration 4-16 to 4-23
  - See also* function A and function B
  - effect of memory operation mode 4-22 to 4-23
  - microcomputer interface example 14-2 to 14-13
    - See also microcomputer interface example bank switching examples 14-12 to 14-13*
    - design options 14-10 to 14-11*
    - slowing memory access 14-3*
    - timing 14-4*
      - read cycle 14-5 to 14-8
      - write cycle 14-9 to 14-10
  - pins 4-16 to 4-23
    - by device 4-16*
  - port control registers 4-17 to 4-18, B-3
    - set-up 4-19 to 4-20*
    - example 4-20 to 4-22
  - system interface example 4-23
- I/O ports. *See* ports
- IDLE (idle until interrupt) instruction 4-6, 10-9, 13-50
  - exiting halt mode 4-8
  - exiting standby mode 4-7
- idle line mode. *See* SCI, multiprocessor communications, idle line mode
- idle mode 4-6 to 4-10
  - control bits 4-6
  - definition J-4
  - entering 4-15
- IE1 bit (ST register) 3-5, 4-7, 5-2, 5-7, 7-21
  - while servicing an interrupt 5-14

## interrupts (continued)

- enabling 5-5
  - EINT instruction* 13-47
  - EINTH instruction* 13-48
  - EINTL instruction* 13-49
- exiting a low-power mode 5-8
- exiting halt mode 4-8
  - considerations* 4-8 to 4-10
  - correct method* 4-9
  - incorrect method* 4-10
- exiting standby mode 4-8
- external 5-6 to 5-9
  - configuring* 5-7
- functions of bits 5-8
- interrupt 1
  - setting as maskable or NMI* 4-14
- memory map 5-7
- nested routines 5-14
- NMI
  - caution while using* 5-7
  - definition* J-6
  - INT1 during hard watchdog* 4-7, 5-7, 7-21
  - programming INT1* 5-7
  - purpose* 5-7
- PACT 12-5 to 12-6, 12-28 to 12-29
- pins
  - INT1* 5-6 to 5-9
    - determining interrupt priority 5-10
    - determining transition direction 5-10
    - finding current condition 5-10
    - generating an interrupt 5-10
  - INT2* 5-6 to 5-9
    - determining interrupt priority 5-11
    - determining transition direction 5-11
    - finding current condition 5-11
    - generating an interrupt 5-11
  - INT3* 5-6 to 5-9
    - determining interrupt priority 5-12
    - determining transition direction 5-12
    - finding current condition 5-13
    - generating an interrupt 5-12
- priority levels 5-2 to 5-6
  - assigning* 5-3
  - by module* 5-3
  - level 1* 5-2 to 5-3
  - level 2* 5-2 to 5-3
  - servicing* 5-3
- routine description 5-2
- RTI instruction 13-74
- SCI 9-13
- servicing multiple 5-14

## interrupts (continued)

- SPI 10-9
- timer 1 7-16
  - overview* 7-3
- timer 2 8-13
  - overview* 8-2
- vector sources 5-2, 5-4 to 5-5
- INV (invert) instruction 13-53
- isochronous mode 1-8
  - See also* SCI, isochronous mode

## J

- JBIT0 (jump if bit = 0) instruction 13-54
- JBIT1 (jump if bit = 1) instruction 13-55
- Jcnd (jump on condition) instruction 13-56 to 13-57
- JMP (jump unconditional) instruction 13-58
- JMPL (jump long) instruction 13-59
- jump instructions
  - BTJO 13-31
  - BTJZ 13-32
  - DJNZ 13-45
  - JBIT0 13-54
  - JBIT1 13-55
  - Jcnd 13-56 to 13-57
  - JMP 13-58
  - JMPL 13-59

## L

- label
  - definition* J-5
- LDSP (load stack pointer) instruction 13-60
  - loading the SP 3-4
- LDST (load status register) instruction 13-61
  - modifying ST value 3-6
- level 1 interrupt enable bit (IE1). *See* IE1 bit (ST register)
- level 2 interrupt enable bit (IE2). *See* IE2 bit (ST register)
- linker 15-5 to 15-6
  - definition* J-5
  - directives 15-5
  - input files 15-5
  - ordering information 17-19
  - output 15-6
  - tasks 15-5

- memory map (continued)
    - timer 1 7-4
    - timer 2 8-4
    - watchdog timer 7-4
  - microcomputer interface example 14-2 to 14-13
    - bank switching examples 14-12 to 14-13
      - changing RAM banks* 14-13
      - changing to EPROM bank 2* 14-13
      - changing to EPROM bank 3 and RAM bank 2* 14-13
      - initializing to EPROM/RAM bank 1* 14-12
    - design options 14-10 to 14-11
      - faster speed* 14-11
      - lower cost* 14-11
    - slowing memory access 14-3
    - timing 14-4
      - read cycle* 14-5 to 14-8
        - address-to-data 14-5 to 14-6
        - chip-select low-to-data 14-6
        - chip-select high-to-next 14-7
        - read-data hold after chip-select high 14-8 to 14-9
      - write cycle* 14-9 to 14-10
        - data hold after chip-select 14-10
        - write-data set-up time 14-9 to 14-10
  - microcomputer mode 3-15
    - effect on I/O ports 4-22 to 4-23
    - function A expansion signals 3-20
    - function B expansion signals 3-21
    - programming port D 3-16, 3-18 to 3-21
    - selecting 4-11
    - single-chip 3-16 to 3-17
      - definition* J-5
      - memory map* 3-17
      - selecting* 3-16
    - with external expansion 3-18 to 3-21
      - definition* J-5
      - selecting* 3-21
  - microprocessor mode 3-15
    - effect on I/O ports 4-22
    - for ROM-less devices 3-14
    - programming port D 3-22
    - selecting 4-11
    - with internal program memory 3-23 to 3-25
      - definition* J-6
      - selecting* 3-24
    - without internal memory 3-22 to 3-23
      - definition* J-6
      - selecting* 3-23
  - mini-SCI. *See* PACT, mini-SCI
  - mnemonic
    - definition J-6
  - modes. *See* operating modes; privilege mode
  - MOV (move) instruction 13-62 to 13-63
  - move instructions
    - MOV 13-62 to 13-63
    - MOVW 13-64
  - MOVW (move word) instruction 13-64
  - mP/mC MODE bit (SCCR0 register) 4-11
  - MPY (multiply) instruction 13-65
  - MSB
    - definition J-6
  - MSbyte
    - definition J-6
  - multiple interrupts servicing. *See* interrupts, servicing multiple
  - multiprocessor modes
    - overview 9-2
- ## N
- N bit (ST register) 3-5
  - negative bit (N) 3-5
  - nested interrupts
    - See also* interrupts
    - definition J-6
  - NMI. *See* interrupts, NMI
  - nonmaskable interrupt (NMI). *See* interrupts, NMI
  - nonreturn to zero format 9-6
    - See also* SCI, data formats
    - definition J-6
  - nonwatchdog mode. *See* watchdog timer, standard watchdog, nonwatchdog mode
  - NOP (no operation) instruction 13-66
  - NRZ (nonreturn to zero) format 9-6
    - See also* SCI, data formats
    - definition J-6
  - numbering conventions 17-15 to 17-18
- ## O
- $\overline{\text{OCF}}$  signal
    - definition 4-20
  - OE bit (RXCTL register) 9-26
  - offset
    - calculation 13-9, 13-10
    - definition J-6

## PACT, definitions (continued)

- offset timer definition-time from last event* 12-18
- bit definitions 12-22
- block diagram 12-18
- uses 12-18
- required time slots* 12-20
- virtual timer definition* 12-16 to 12-17
  - bit definitions 12-21
  - block diagram 12-16
  - components 12-16
  - macro I-4
  - uses 12-17
- description 1-7
- dual-port RAM 12-5 to 12-6, 12-9 to 12-10
  - major areas* 12-9
  - memory map* 12-10
- event counter 12-10
- exiting standby mode 4-8
- hardware pins 12-5
  - input*
    - controlling functions 12-46 to 12-51
    - CP1–CP6 12-5, 12-11 to 12-13
  - output*
    - determining state 12-44
    - OP1–OP8 12-5, 12-14
  - outputs, defining* I-3
  - SCI receive/transmit*
    - RX 12-5
    - TX 12-5
- input captures 12-11 to 12-13
  - block diagram* 12-12
- interrupts 12-5 to 12-6, 12-28 to 12-29
  - enabling* 12-38, 12-39, 12-42
  - memory map* 12-28
  - priority level* 5-3
  - priority levels* 12-28
  - sources* 12-29
  - types* 12-28
  - vector sources* 5-4
- memory organization
  - memory map* 12-4, 12-6
  - overview* 12-5 to 12-6
- mini-SCI 12-31
  - calculating baud* 12-31
  - definition* J-6
- operating modes 12-10
  - mode A* 12-12
  - mode B* 12-12
- operation 12-5 to 12-8

## PACT (continued)

- outputs 12-14 to 12-19
  - block diagram* 12-14
- overview 12-2 to 12-4
- PACT.H macros I-1 to I-10
- physical description 12-2 to 12-3
- prescaler 12-6
  - calculating the divide rate* 12-6, 12-37
  - determining the number of time slots* 12-7 to 12-8
  - effect on controller* 12-14
- pulse width modulated (PWM) signal 12-32 to 12-34
  - copying the command/definition area to RAM* 12-33
  - defining the command/definition area* 12-32
  - example* 12-34
  - initializing the PACT peripheral frame* 12-33
- registers 12-35 to 12-54, B-6
  - BUFPTR (buffer pointer)* 12-13 to 12-14, 12-41
  - capture* 12-9
    - location in RAM 12-10
  - CDEND (command/definition area end)* 12-40
  - CDFLAGS (command/definition entry flags)* 12-45
  - CDSTART (command/definition area start)* 12-39
  - CPCTL1 (set-up CP control 1)* 12-46 to 12-47
  - CPCTL2 (set-up CP control 2)* 12-48 to 12-49
  - CPCTL3 (set-up CP control 3)* 12-50 to 12-51
  - CPPRE (CP input control)* 12-52 to 12-53
  - dedicated capture registers, definition* J-3
  - OPSTATE (output pins 1–8 state)* 12-44
  - overview* 12-4
  - PACTPRI (global function control)* 12-54
  - PACTSCR (set-up control)* 12-37 to 12-38
  - RXBUF (PACT-SCI RX data)* 12-43
  - SCICTLP (PACT-SCI control)* 12-42 to 12-43
  - TXBUF (PACT-SCI TX data)* 12-43
- sample routines 14-22 to 14-29
  - double event compare command example* 14-26 to 14-27
  - routine* 14-27

## peripheral file (continued)

frames 3-11 to 3-12

definition J-7

frame 1 B-2

See also system configuration; interrupts

memory map 5-7

overview 4-2

register descriptions 4-11 to 4-15, 5-10 to 5-13,  
6-4 to 6-5, 6-11

frame 2 4-17 to 4-18, B-3

See also I/O configuration

register descriptions 4-17 to 4-18

frame 3 B-4

See also SPI

memory map 10-4

register descriptions 10-13 to 10-20

frame 4 12-5 to 12-6, B-5, B-6

See also timer 1; watchdog timer; PACT

memory map 7-4, 12-4

register descriptions 7-25 to 7-38, 12-35 to  
12-54

frame 5 B-7

See also SCI

memory map 9-5

register descriptions 9-19 to 9-32

frame 6 B-8

See also timer 2

memory map 8-4

register descriptions 8-15 to 8-24

frame 7 B-9

See also A/D converter module

memory map 11-3

register descriptions 11-9 to 11-14

hexadecimal designator 3-11

memory map B-1 to B-9

peripheral addressing mode 13-7

PF (peripheral file). See peripheral file

PF AUTOWAIT bit (SCCR0 register) 4-4 to 4-7,  
4-12

slowing memory access 14-3

## pins and signals

See also individual pin and signal listings

A/D pins 11-4

AN0–AN7 11-4

CLKOUT, definition 4-20

CP1–CP6 12-5, 12-11 to 12-13

CSE1

activating 3-18

definition 4-19

CSE2

activating 3-18

definition 4-19

## pins and signals (continued)

CSH1

activating 3-18

definition 4-19

CSH2

activating 3-18

definition 4-19

CSH3

activating 3-18

definition 4-20

CSPF

activating 3-19

definition 4-19

DATA BUS

definition 4-20

descriptions

TMS370Cx1x 2-3

TMS370Cx2x 2-5

TMS370Cx3x 2-7

TMS370Cx4x 2-9

TMS370Cx5x 2-11 to 2-13

EDS 4-13

activating 3-20, 3-22, 3-23

definition 4-20

INT1 5-6 to 5-9

INT2 5-6 to 5-9

INT3 5-6 to 5-9

LOW ADDR/HI ADDR

definition 4-20

MC

definition J-5

finding voltage status 4-11

overriding write protection 4-11

selecting the memory operating mode 3-15

microcomputer single-chip 3-16

microcomputer with external expansion 3-21

microprocessor with internal program

memory 3-24

microprocessor without internal memory 3-23

OCF

definition 4-20

OP1–OP8 12-5, 12-14

pin descriptions 2-1 to 2-13

pinouts

TMS370Cx1x 2-2, G-2

TMS370Cx2x 2-4, G-2

TMS370Cx3x 2-6, G-3

TMS370Cx4x 2-8, G-3

TMS370Cx5x 2-10, G-4

PLCC to PGA sockets H-1 to H-9

- production lead time
    - definition 17-4
  - program
    - EPROM 3-14, 6-10 to 6-14
      - access time 3-14
      - control register
        - overview B-2
      - DEECTL register 6-11
      - description 1-6
      - entering reserved mode 6-13
      - finding voltage status 6-11
      - preparing for programming 6-10
      - programming 6-12 to 6-13
        - flowchart 6-13
        - initiating 6-12
        - preventing 6-13
        - voltage required 6-13
      - reading 6-10
      - write protecting 6-14
    - memory 3-13 to 3-14
      - disabling 4-13
      - enabling 4-13
      - for ROM-less devices 3-14
      - map 3-3
      - vector address map 3-13
    - ROM 3-14
  - program counter (PC) 3-7
    - definition 3-2, J-7
    - during context switch routine 5-6
    - during interrupt routine 5-2
    - during reset sequence 5-15
    - PCH (program counter high) 3-7
    - PCL (program counter low) 3-7
    - value during low-power modes 4-7
    - value during reset 3-7
  - program counter relative addressing mode 13-9
  - PROGRAM procedure 14-14 to 14-16
  - programmable acquisition and control timer. *See* PACT
  - programmer
    - gang 15-2, 15-24
      - definition J-4
      - features 15-24
      - ordering information 17-19
    - microcontroller 15-2, 15-22 to 15-23
      - definition J-6
      - features 15-23
      - ordering information 17-19
  - prototyping device
    - definition J-7
    - numbering conventions 17-18 to 17-20
    - production flow 17-2 to 17-5
  - prototyping lead time, definition 17-4
  - pulse accumulation 7-15, 8-13
    - definition J-7
  - PUSH (push on stack) instruction 13-69
  - PWM
    - See also* timer 1, PWM applications
    - definition J-7
  - PWRDWN/IDLE bit (SCCR2 register) 4-6 to 4-7, 4-15, 7-24, 8-14
- ## R
- R0 register. *See* A (R0) register
  - R1 register. *See* B (R1) register
  - RAM
    - See also* dual-port RAM; microcomputer interface example
    - clearing example 14-31
    - definition J-8
    - self test 14-32
  - ratiometric conversion. *See* A/D converter module, ratiometric conversion
  - RCVD0–7 bits (SPIBUF register) 10-17
  - receiver. *See* SCI, receiver
  - RECEIVER OVERRUN bit (SPICL register) 10-9, 10-16
  - REF VOLT SELECT0–2 bits (ADCTL register) 11-10
  - register addressing mode 13-6
  - register file 3-9 to 3-10
    - access time 3-10
    - accessing 3-10
    - address map 3-9
    - definition 3-2, J-8
    - description 1-5
    - memory map 3-3
    - partitioning 3-10
    - register addressing mode 13-6
  - registers
    - See also* individual register name listings
    - A (R0) 3-10
    - A/D control registers 11-9 to 11-14, B-9
    - ADATA, ADIR 4-17 to 4-18
    - ADCTL 11-10 to 11-11



- reprogrammable EPROMs 15-25
  - numbering conventions 17-18
- reset 5-15 to 5-18
  - See also* privilege mode circuit
  - simple* 5-16 to 5-17
  - with low-voltage detection* 5-17 to 5-18
  - exiting halt mode 4-8
  - exiting standby mode 4-8
  - finding cause 4-12
  - generating
    - by the hard watchdog* 7-20
    - by the standard watchdog* 7-18
  - initializing the SPI 10-11
  - PC value 3-7
  - register A value 3-10
  - register B value 3-10
  - resetting timer 1 counter 7-5, 7-10
  - resetting timer 2 counter 8-5
  - selecting the memory operating mode 3-15
  - sequence 5-15
    - number of cycles* 5-16
  - sources 5-15
  - SP value 3-4
  - value of control bits following reset 5-16
  - vectors 3-13, 5-2
- RESET** pin
  - as reset source 5-15
  - definition J-8
  - during oscillator fault 5-16
  - during watchdog overflow 5-16
  - entering reserved mode 6-13
  - reset circuit
    - simple* 5-16 to 5-17
    - with low-voltage detection* 5-17 to 5-18
  - selecting memory operation mode 3-15
    - microcomputer single-chip* 3-16
    - microcomputer with external expansion* 3-21
    - microprocessor with internal program memory* 3-24
    - microprocessor without internal memory* 3-23
- return from interrupt (RTI) instruction. *See* RTI instruction
- RF (register file). *See* register file
- RL (rotate left) instruction 13-70
- RLC (rotate left through carry) instruction 13-71
- ROM
  - See also* mask-ROM
  - checksum example 14-33 to 14-34
  - program 3-14
- rotate instructions
  - RL 13-70
  - RLC 13-71
  - RR 13-72
  - RRC 13-73
- RR (rotate right) instruction 13-72
- RRC (rotate right through carry) instruction 13-73
- RTI (return from interrupt) instruction 13-74
  - function at end of interrupt routine 5-2
  - restoring ST value 3-6
- RTS (return from subroutine) instruction 13-75
- R/W signal
  - definition 4-20
- RX ERROR bit (RXCTL register) 9-27
- RX pin 12-5
- RXBUF (SCI receiver data buffer) register 9-3, 9-13, 9-28
  - RXDT0–7 bits 9-28
- RXBUPF (PACT-SCI RX data) register 12-43
  - PACT RXDT0–7 bits 12-43
- RXCTL (SCI receiver interrupt control and status) register 9-26 to 9-27
  - BRKDT bit 9-13, 9-27
  - FE bit 9-27
  - OE bit 9-26
  - PE bit 9-26
  - RX ERROR bit 9-27
  - RXRDY bit 9-7, 9-13, 9-27
  - RXWAKE bit 9-26
  - SCI RX INT ENA bit 9-13, 9-26
- RXDT0–7 bits (RXBUF register) 9-28
- RXENA bit (SCICTL register) 9-22
- RXRDY bit (RXCTL register) 9-7, 9-13, 9-27
- RXSHF register 9-3
- RXWAKE bit (RXCTL register) 9-26

## S

- SAMPLE START bit (ADCTL register) 11-6, 11-11
- sample time. *See* A/D converter module, sampling time
- SBB (subtract with borrow) instruction 13-76
  - setting the V bit of ST 3-5

## SCI (continued)

- initialization examples 9-16 to 9-18
  - RS-232-C* 9-16
  - RS-232-C multiprocessor mode* 9-17 to 9-18
- interrupts 9-13
  - disabling* 9-25, 9-26
  - enabling* 9-25, 9-26
  - generating during multiprocessor communications* 9-7
  - priority level* 5-3
  - sequence of events* 9-7 to 9-8
  - vector sources* 5-4
- isosynchronous mode 9-11
  - baud formula* 9-14, 9-24
  - bit rates* 9-2
  - communication format* 9-11
  - definition* 9-4, J-5
  - selecting* 9-20
  - three-line communication* 9-11
  - two-line communication* 9-11
- key features 9-2 to 9-3
- memory map 9-5
- multiprocessor communications 9-7 to 9-9
  - address bit mode* 9-7, 9-9
  - data format* 9-6
  - format* 9-9
  - definition* J-6
  - generating an interrupt* 9-7
  - idle line mode* 9-7, 9-8 to 9-9
    - data format* 9-6
    - format* 9-8
  - sending a block start signal* 9-8, 9-9
  - interrupt sequence* 9-7 to 9-8
  - overview* 9-4 to 9-5
  - selecting mode* 9-7, 9-20
- overview 9-2 to 9-5
- physical description 9-3 to 9-4
- pins
  - SCICLK* 9-10, 9-11, 9-14 to 9-15
    - control register* 9-29
    - frequency formula* 9-24
    - selecting source* 9-22 to 9-23
  - SCIRXD* 9-3, 9-12
    - as general-purpose pin* 14-18
    - control register* 9-30
    - exiting halt mode* 4-8
    - exiting standby mode* 4-8
  - SCITXD* 9-3, 9-12
    - control register* 9-30 to 9-31
- port interfacing 14-18 to 14-20
  - example* 14-18

## SCI (continued)

- receiver
  - overview* 9-3
- receiving data
  - during low-power modes* 4-7
- reset 9-23
  - affected bits* 9-23
- supported devices 14-17
- timings 16-24 to 16-25
  - differences* A-6
- transmitter
  - overview* 9-3
- transmitting data
  - during low-power modes* 4-7
- WUT flag 9-8, 9-9
- SCI CHAR0–2 bits (SCICCR register) 9-20
- SCI ESPEN bit (SCIPRI register) 9-32
- SCI RX INT ENA bit (RXCTL register) 9-13, 9-26
- SCI RX PRIORITY bit (SCIPRI register) 9-13, 9-32
- SCI STEST bit (SCIPRI register) 9-32
- SCI SW RESET bit (SCICTL register) 9-23
- SCI TX INT ENA bit (TXCTL register) 9-13, 9-25
- SCI TX PRIORITY bit (SCIPRI register) 9-13, 9-32
- SCICCR (SCI communication control) register 9-20 to 9-21
  - ADDRESS/IDLE WUP bit* 9-7, 9-20
  - ASYNC/ISOSYNC bit* 9-10, 9-20
  - EVEN/ODD PARITY bit* 9-21
  - PARITY ENABLE bit* 9-21
  - programming the data format* 9-6
  - SCI CHAR0–2 bits* 9-20
  - STOP BITS bit* 9-21
- SCICLK DATA DIR bit (SCIPC1 register) 9-29
- SCICLK DATA IN bit (SCIPC1 register) 9-15, 9-29
- SCICLK DATA OUT bit (SCIPC1 register) 9-29
- SCICLK FUNCTION bit (SCIPC1 register) 9-10, 9-14, 9-29
- SCICLK pin. *See* SCI, pins, SCICLK
- SCICTL (SCI control) register 9-22 to 9-23
  - CLOCK bit* 9-10, 9-14, 9-22 to 9-23
  - RXENA bit* 9-22
  - SCI SW RESET bit* 9-23
  - SLEEP bit* 9-7, 9-8, 9-9, 9-22
  - TXENA bit* 9-22
  - TXWAKE bit* 9-7, 9-8, 9-9, 9-22
    - double-buffered WUT flag* 9-8

- SPI (continued)
  - operating modes 10-6 to 10-7
    - master* 10-6
      - clock source 10-10
      - communications 10-5
    - selecting* 10-16
    - slave* 10-6 to 10-7
      - clock source 10-10
      - communications 10-5
  - overview 10-2 to 10-4
  - physical description 10-2 to 10-3
  - pins
    - SPICLK* 10-2, 10-5, 10-6 to 10-7, 10-11
      - control register 10-18
      - selecting polarity 10-15
    - SPISIMO* 10-2, 10-6 to 10-7
      - control register 10-19 to 10-20
    - SPISOMI* 10-2, 10-6
      - control register 10-19
  - port interfacing 14-17
    - example* 14-17
  - receiving data
    - during low-power modes* 4-7
  - reset 10-15
  - supported devices 14-17
  - timings 16-26 to 16-27
    - differences* A-6
  - transmitting data
    - during low-power modes* 4-7
- SPI BIT RATE0–2 bits (SPICCR register) 10-6, 10-10, 10-14
- SPI CHAR0–2 bits (SPICCR register) 10-8, 10-14
- SPI ESPEN bit (SPIPRI register) 10-20
- SPI INT ENA bit (SPICTL register) 10-6, 10-9, 10-16
- SPI INT FLAG bit (SPICTL register) 10-6, 10-9, 10-16
- SPI PRIORITY bit (SPIPRI register) 10-9, 10-20
- SPI STEST bit (SPIPRI register) 10-20
- SPI SW RESET bit (SPICCR register) 10-9, 10-11, 10-15
- SPIBUF (serial input buffer) register 10-17
  - clearing interrupt flag 10-9
  - definition 10-2
  - during data transmission 10-6
  - formatting 10-8
  - RCVD0–7 bits 10-17
- SPICCR (SPI configuration control) register 10-14 to 10-15
  - CLOCK POLARITY bit 10-10, 10-15
  - SPI BIT RATE0–2 bits 10-6, 10-10, 10-14
  - SPI CHAR0–2 bits 10-8, 10-14
  - SPI SW RESET bit 10-9, 10-11, 10-15
- SPICLK DATA DIR bit (SPIPC1 register) 10-18
- SPICLK DATA IN bit (SPIPC1 register) 10-18
- SPICLK DATA OUT bit (SPIPC1 register) 10-18
- SPICLK FUNCTION bit (SPIPC1 register) 10-18
- SPICLK pin. *See* SPI, pins, SPICLK
- SPICTL (SPI operation control) register 10-16
  - MASTER/SLAVE bit 10-6, 10-11, 10-16
  - RECEIVER OVERRUN bit 10-9, 10-16
  - SPI INT ENA bit 10-6, 10-9, 10-16
  - SPI INT FLAG bit 10-6, 10-9, 10-16
  - TALK bit 10-7, 10-11, 10-16
- SPIDAT (serial data) register 10-17
  - definition 10-2
  - during data transmission 10-6 to 10-7
  - formatting 10-8 to 10-9
  - initialization 10-11 to 10-12
  - SDAT0–7 bits 10-17
- SPIPC1 (SPI port control 1) register 10-18
  - SPICLK DATA DIR bit 10-18
  - SPICLK DATA IN bit 10-18
  - SPICLK DATA OUT bit 10-18
  - SPICLK FUNCTION bit 10-18
- SPIPC2 (SPI port control 2) register 10-19 to 10-20
  - SPISIMO DATA DIR bit 10-19
  - SPISIMO DATA IN bit 10-20
  - SPISIMO DATA OUT bit 10-20
  - SPISIMO FUNCTION bit 10-19
  - SPISOMI DATA DIR bit 10-19
  - SPISOMI DATA IN bit 10-19
  - SPISOMI DATA OUT bit 10-19
  - SPISOMI FUNCTION bit 10-19
- SPIPRI (SPI interrupt priority control) register 10-20
  - SPI ESPEN bit 10-20
  - SPI PRIORITY bit 10-9, 10-20
  - SPI STEST bit 10-20
- SPISIMO DATA DIR bit (SPIPC2 register) 10-19
- SPISIMO DATA IN bit (SPIPC2 register) 10-20
- SPISIMO DATA OUT bit (SPIPC2 register) 10-20
- SPISIMO FUNCTION bit (SPIPC2 register) 10-19
- SPISIMO pin. *See* SPI, pins, SPISIMO
- SPISOMI DATA DIR bit (SPIPC2 register) 10-19

- T1C1 INT FLAG bit (T1CTL3 register) 7-6, 7-7, 7-8, 7-31
- T1C1 OUT ENA bit (T1CTL4 register) 7-6, 7-8, 7-34
- T1C1 RST ENA bit (T1CTL4 register) 7-6, 7-8, 7-34
- T1C2 INT ENA bit (T1CTL3 register) 7-7, 7-8, 7-16, 7-31
- T1C2 INT FLAG bit (T1CTL3 register) 7-32
- T1C2 OUT ENA bit (T1CTL4 register) 7-7, 7-8, 7-34
- T1CNTR counter. *See* timer 1, counter (T1CNTR)
- T1CR OUT ENA bit (T1CTL4 register) 7-34
- T1CR RST ENA bit (T1CTL4 register) 7-12, 7-33
- T1CTL1 (timer 1 control 1) register 7-27 to 7-28
  - T1 INPUT SELECT0–2 bits 7-27
  - WD INPUT SELECT0–2 bits 7-20, 7-22, 7-27 to 7-28
  - WD OVRFL TAP SEL bit 7-14, 7-18, 7-20, 7-22, 7-28
- T1CTL2 (timer 1 control 2) register 7-29 to 7-30
  - caution when modifying 7-30
  - differences in bits A-3
  - T1 OVRFL INT ENA bit 7-5, 7-10, 7-16, 7-29
  - T1 OVRFL INT FLAG bit 7-5, 7-10, 7-29
  - T1 SW RESET bit 7-5, 7-10, 7-29
  - WD OVRFL INT ENA bit 7-29
  - WD OVRFL INT FLAG bit 7-19, 7-20, 7-21, 7-29
    - as source of reset* 5-15
    - differences* A-3
  - WD OVRFL RST ENA bit 7-18, 7-20, 7-22, 7-30
    - differences* A-3
- T1CTL3 (timer 1 control 3) register 7-31 to 7-32
  - caution when modifying 7-32
  - T1C1 INT ENA bit 7-6, 7-8, 7-16, 7-31
  - T1C1 INT FLAG bit 7-6, 7-7, 7-8, 7-31
  - T1C2 INT ENA bit 7-7, 7-8, 7-16, 7-31
  - T1C2 INT FLAG bit 7-32
  - T1EDGE INT ENA bit 7-11, 7-16, 7-31
  - T1EDGE INT FLAG bit 7-7, 7-10, 7-11, 7-12, 7-32
- T1CTL4 (timer 1 control 4) register 7-33 to 7-34
  - T1 MODE bit 7-8, 7-34
  - T1C1 OUT ENA bit 7-6, 7-8, 7-34
  - T1C1 RST ENA bit 7-6, 7-8, 7-34
  - T1C2 OUT ENA bit 7-7, 7-8, 7-34
  - T1CR OUT ENA bit 7-34
  - T1CTL4 (timer 1 control 4) register (continued)
    - T1CR RST ENA bit 7-12, 7-33
    - T1EDGE DET ENA bit 7-10, 7-11, 7-12, 7-33
    - T1EDGE POLARITY bit 7-5, 7-10, 7-11, 7-12, 7-33
  - T1EDGE DET ENA bit (T1CTL4 register) 7-10, 7-11, 7-12, 7-33
  - T1EDGE INT ENA bit (T1CTL3 register) 7-11, 7-16, 7-31
  - T1EDGE INT FLAG bit (T1CTL3 register) 7-7, 7-10, 7-11, 7-12, 7-32
  - T1EDGE POLARITY bit (T1CTL4 register) 7-5, 7-10, 7-11, 7-12, 7-33
  - T1EVT DATA DIR bit (T1PC1 register) 7-35
  - T1EVT DATA IN bit (T1PC1 register) 7-35
  - T1EVT DATA OUT bit (T1PC1 register) 7-35
  - T1EVT FUNCTION bit (T1PC1 register) 7-35
  - T1EVT pin. *See* timer 1, pins, T1EVT
  - T1IC/CR DATA DIR bit (T1PC2 register) 7-36
  - T1IC/CR DATA FUNCTION bit (T1PC2 register) 7-36
  - T1IC/CR DATA IN bit (T1PC2 register) 7-36
  - T1IC/CR DATA OUT bit (T1PC2 register) 7-36
  - T1IC/CR pin. *See* timer 1, pins, T1IC/CR
  - T1PC1 (timer 1 port control 1) register 7-35
    - T1EVT DATA DIR bit 7-35
    - T1EVT DATA IN bit 7-35
    - T1EVT DATA OUT bit 7-35
    - T1EVT FUNCTION bit 7-35
  - T1PC2 (timer 1 port control 2) register 7-36 to 7-37
    - T1IC/CR DATA DIR bit 7-36
    - T1IC/CR DATA FUNCTION bit 7-36
    - T1IC/CR DATA IN bit 7-36
    - T1IC/CR DATA OUT bit 7-36
    - T1PWM DATA DIR bit 7-36
    - T1PWM DATA FUNCTION bit 7-36
    - T1PWM DATA IN bit 7-37
    - T1PWM DATA OUT bit 7-37
  - T1PRI (timer 1 interrupt priority control) register 7-38
    - T1 PRIORITY bit 7-38
    - T1 STEST bit 7-38
  - T1PWM DATA DIR bit (T1PC2 register) 7-36
  - T1PWM DATA FUNCTION bit (T1PC2 register) 7-36
  - T1PWM DATA IN bit (T1PC2 register) 7-37
  - T1PWM DATA OUT bit (T1PC2 register) 7-37

- T2PC2 (timer 2 port control 2) register (continued)
  - T2IC2/PWM DATA IN bit 8-24
  - T2IC2/PWM DATA OUT bit 8-24
  - T2IC2/PWM FUNCTION bit 8-23
- T2PRI (timer 2 interrupt priority control) register 8-24
  - T2 PRIORITY bit 8-24
  - T2 STEST bit 8-24
- TALK bit (SPICTL register) 10-7, 10-11, 10-16
- three-line communication 9-11
- time slots. *See* PACT, time slots
- timer
  - default
    - definition* J-3
  - PACT
    - See also* PACT
    - description* 1-7
  - timer 1
    - See also* timer 1
    - description* 1-6 to 1-7
  - timer 2
    - See also* timer 2
    - description* 1-6 to 1-7
  - watchdog
    - See also* watchdog timer
    - description* 1-7
- timer 1 7-1 to 7-38
  - block diagram 7-3
  - capabilities 7-2
  - clock 7-10, 7-13 to 7-15
    - counter duration* 7-14
    - counter resolution* 7-14
    - delays during compare equal* 7-6
    - event counter mode* 7-15
    - external clock input* 7-13
    - pulse accumulator mode* 7-15
    - selecting source* 7-27
    - sources* 7-13
  - counter (T1CNTR) 7-5
    - clock input* 7-10
    - generating a rollover interrupt* 7-5, 7-10
    - initialization* 7-5
    - resetting* 7-5, 7-10
      - during compare equal 7-5, 7-6, 7-8, 7-10
      - using the T1IC/CR pin 7-10
  - description* 1-6 to 1-7
- timer 1 (continued)
  - edge detection 7-10, 7-12
    - during capture/compare mode* 7-12
    - during dual compare mode* 7-12
    - generating an interrupt* 7-10
    - overview* 7-2
    - toggleing the T1PWM pin* 7-10
  - exiting standby mode 4-8
  - interrupts 7-16
    - clearing flags* 7-16
    - control register* 7-38
    - disabling* 7-29
      - compare register interrupts 7-31
    - enabling* 7-29
      - compare register interrupts 7-31
    - finding status* 7-29
    - generating*
      - by edge detection 7-10
      - during compare equal 7-6, 7-7, 7-8
      - with the T1IC/CR pin 7-10
    - overview* 7-3
    - priority level* 5-3
    - vector sources* 5-5
  - low-power modes 7-24
    - See also* low-power modes
    - halt* 7-24
    - standby* 7-24
  - memory map 7-4
  - operating modes 7-8 to 7-11
    - capture/compare* 7-11
      - block diagram 7-11, C-7
      - components 7-11
      - edge detection 7-12
      - function of capture/compare register 7-7
      - overview* 7-4
    - dual compare* 7-8 to 7-10
      - block diagram 7-9, C-6
      - components 7-8
      - edge detection 7-10, 7-12
      - function of capture/compare register 7-7
      - overview* 7-4
      - PWM applications 7-9 to 7-10
    - overview* 7-4
    - selecting* 7-8, 7-34
  - overview* 7-2 to 7-4
  - pins
    - definitions* 7-3
    - overview* 7-3
  - T1EVT 7-10, 7-13, 7-15
    - control register 7-35
    - overview* 7-3
    - use with watchdog timer 7-20

## timer 2 (continued)

## pins

*definitions* 8-3

*overview* 8-2 to 8-3

*T2EVT* 8-12 to 8-13

control register 8-22

overview 8-2 to 8-3

*T2IC1/CR* 8-7, 8-10, 8-11

control register 8-23 to 8-24

determining transition direction 8-21

generating an interrupt 8-18

overview 8-2 to 8-3

resetting the counter 8-5

*T2IC2/PWM* 8-10, 8-11

control register 8-23 to 8-24

determining transition direction 8-21

during compare equal 8-5, 8-7

generating an interrupt 8-18

overview 8-2 to 8-3

## registers 8-15 to 8-24

*capture* 8-6 to 8-7

definition J-2

*capture/compare* 8-7

during dual capture mode 8-7

during dual compare mode 8-7

*compare* 8-5 to 8-6

definition J-3

formula for registers values 8-6

resetting counter when equal 8-5, 8-11

sample registers values 8-6

*overview* 8-2, 8-4

*T2CTL1 (timer 2 control 1)* 8-17

*T2CTL2 (timer 2 control 2)* 8-18 to 8-19

*T2CTL3 (timer 2 control 3)* 8-20 to 8-21

*T2PC1 (timer 2 port control 1)* 8-22

*T2PC2 (timer 2 port control 2)* 8-23 to 8-24

*T2PRI (timer 2 interrupt priority control)* 8-24

## reset sources 8-5

## timings 16-1 to 16-52

for all devices 16-5

parameter measurements 16-2

parameter symbols 16-2

TMS370Cx1xA 16-8

TMS370Cx2xA 16-11

TMS370Cx3x 16-14

TMS370Cx4xA 16-17

TMS370Cx5xA 16-21 to 16-29

*differences* A-5

TMP prefix 17-15

TMS prefix 17-15

## TMS370 family

applications 1-3

architecture overview 1-5 to 1-8, 3-2 to 3-3

categories 1-3

design aids 14-1 to 14-44

*See also design aids*

development tools 15-1 to 15-25

*ordering information* 17-19 to 17-20

*software development flow* 15-3

differences A-1 to A-7

*summary* A-7

key features 1-4

numbering conventions 17-15 to 17-18

ordering information 17-1 to 17-20

*development tools* 17-19 to 17-20

*mechanical data* 17-6 to 17-14

*prototyping device production flow* 17-2 to

17-5

overview 1-1 to 1-15

packaging 17-6 to 17-14

pinouts and pin descriptions 2-1 to 2-13

*See also pins and signals, descriptions*

prototyping device

*See also prototyping device*

*definition* J-7

summary of components (by device) 1-8 to

1-10

## TMS370Cx1x

available pins 4-16

block diagram 1-11

electrical specifications 16-6 to 16-7

PGA pinout H-3

pin descriptions 2-3

pinouts 2-2, G-2

timings 16-8

## TMS370Cx2x

available pins 4-16

block diagram 1-12

electrical specifications 16-9 to 16-10

PGA pinout H-5

pin descriptions 2-5

pinouts 2-4, G-2

timings 16-11

## TMS370Cx3x

available pins 4-16

block diagram 1-13

electrical specifications 16-12 to 16-13

PGA pinout H-6

- watchdog timer (continued)
    - hard watchdog 7-17, 7-20 to 7-21
      - additional system integrity* 7-20
      - affect on INT1 pin* 4-7, 5-7, 7-21
      - block diagram* 7-20, C-5
      - generating a reset* 7-20
      - summary* 7-23, A-2
    - interrupts
      - disabling* 7-29
      - enabling* 7-29
      - finding status* 7-29
    - low-power modes 7-24
      - See also low-power modes*
      - during* 4-7
      - entering* 7-21
      - exiting* 7-21
      - halt* 7-24
      - standby* 7-24
    - memory map 7-4
    - overflow rates 7-14, 7-18
    - overview 7-2, 7-17
    - registers 7-25 to 7-30, B-5
      - T1CTL1 (timer 1 control 1)* 7-27 to 7-28
      - T1CTL2 (timer 1 control 2)* 7-29 to 7-30
    - reset key (WDRST) 7-21, 7-22, 12-30
      - determining source of reset* 4-4
      - reinitializing the watchdog counter* 7-18 to 7-19, 7-20 to 7-21
    - simple counter 7-17, 7-22
      - block diagram* 7-22, C-5
      - summary* 7-23, A-2
    - standard watchdog 7-17, 7-18 to 7-20
      - block diagram* 7-18, C-5
      - generating a reset* 7-18
      - initialization example* 7-19
      - nonwatchdog mode* 7-20 to 7-21
      - selecting* 7-18
      - summary* 7-23, A-2
      - watchdog mode* 7-18 to 7-19
    - timeout as reset source 5-15
  - WD INPUT SELECT0–2 bits (T1CTL1 register) 7-20, 7-22, 7-27 to 7-28
  - WD OVRFL INT ENA bit (T1CTL2 register) 7-29
  - WD OVRFL INT FLAG bit (T1CTL2 register) 7-19, 7-20, 7-21, 7-29
    - as source of reset 5-15
    - differences A-3
  - WD OVRFL RST ENA bit (T1CTL2 register) 7-18, 7-20, 7-22, 7-30
    - differences A-3
  - WD OVRFL TAP SEL bit (T1CTL1 register) 7-14, 7-18, 7-20, 7-22, 7-28
  - WDRST (watchdog reset) register
    - See also* watchdog timer, reset key (WDRST)
    - determining source of reset 4-4
  - WPO. *See* write protect override
  - WPR (write protection) register 3-12, 6-2, 6-3 to 6-4, 6-6 to 6-9
    - memory map 6-3
    - programming example 6-4
  - write protect override 3-14, 3-15
    - definition J-10
  - write protection override 6-3
  - WUT flag 9-8, 9-9
- X**
- XCHB (exchange with register B) instruction 13-85
  - XDS/22 15-2, 15-15 to 15-17
    - breakpoint, trace, and timing functions 15-12 to 15-14
      - key features* 15-12 to 15-14
    - breakpoints
      - A/D converter module actions* 11-14
      - SCI module actions* 9-32
      - SPI module actions* 10-20
    - configuration requirements 15-16
    - definition J-10
    - hardware 15-15
    - key features 15-15
    - operating considerations 15-17
    - ordering information 17-20
    - PACT 15-2, 15-15
  - XOR (exclusive OR) instruction 13-86
- Z**
- Z bit (ST register) 3-5
  - zero bit (Z) 3-5

