TEXAS INSTRUMENTS

TMS370 Family

Data Manual

1988

8-Bit Microcontroller Family

Microcontroller

Technical Hotline	Bulletin Board Service
713 274-2370	713 274-3700

TMS370 Family Data Manual

TEXAS INSTRUMENTS

IMPORTANT NOTICE

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

TI warrants performance of its semiconductor products 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. Unless mandated by government requirements, specific testing of all parameters of each device is not necessarily performed.

TI assumes no liability for TI applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that 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.

Contents

Illustrations

Figure *Page*

Tables

Table *Page*

1. Introduction

This manual describes the TMS370 family of microcontroller products. The objective of the manual is to provide the information needed to implement a microcontroller design using a TMS370 device.

This section gives a broad overview of the TMS370 family covering the following topics:

1.1 TMS370 Overview

The TMS370 family consists of VLSI, 8-bit, CMOS microcontrollers with on-chip EEPROM storage and peripheral support functions. This family of microcontrollers provides superior performance in complex real-time control applications in demanding environments. With devices available in mask-programmable read-only memory (ROM) and electrically-erasable programmable read-only memory (EEPROM), the designer has a significant range of options to chose from in deciding the most economical, efficient manner of getting a product to market.

The prototyping and production devices of the TMS370 family are totally interchangeable. This reduces development costs and cycle time, and facilitates rapid product modification and upgrade. The alterable non-volatile memory (EEPROM) allows a designer to customize his equipment for a specific application with quick turnaround.

The TMS370 family is fully supported by a host of TI development tools which provide simplified software development for quicker market introduction of new products. These development support tools include an Assembler, a Linker, an In-Circuit emulator (XDS - eXtended Development Support), and an EEPROM/UVEPROM programmer. All of these tools work together using an MS™-DOS-based Personal Computer (PC) as the host and central control element.[1] This allows selection of the host computer and the text management and editing tools based on user preference.

TMS370 FEATURES AND BENEFITS

FEATURES	BENEFITS
– Sub 2-Micron Technology	– Low power consumption over wide temperature range
– Series of compatible devices	– Supports software migration
– EEPROM Technology	– Alterable, non-volatile memory on-chip to support in-socket programming and form factor emulation
– Versatile memory configurations	– Many memory options to meet applications requirements
– Programmable Interrupt Handling	– Provides design flexibility
– 14 Addressing Modes	– Increases programmer's flexibility during software development phase

[1] MS is a trademark of Microsoft Corporation.

Table 1-1. TMS370 Family Features

FEATURE	370C010	370C810	370C050	370C850	COMMENTS/BENEFITS
Program Memory	4 Kbytes ROM	4 Kbytes EEPROM	4 Kbytes ROM	4 Kbytes EEPROM	EEPROM supports in-socket programming
Static RAM	128 bytes	128 bytes	256 bytes	256 bytes	Data retention in low-power modes
Data EEPROM	256 bytes	256 bytes	256 bytes	256 bytes	Data retention in power-off mode
Watchdog Timer	Y	Y	Y	Y	Helps ensure system integrity
Timer 1	Y	Y	Y	Y	16 bits with 200 ns resolution
Timer 2			Y	Y	16 bits with 200 ns resolution
A/D Converter			Y	Y	8 channel, 8-bit accuracy with selectable references; provides conversion of external analog inputs in 164 cycles
Serial Communications Interface			Y	Y	Async. transmission up to 156 kbits/s; Sync transmission up to 2.5 Mbits/s; software selectable baud rate and data format
Serial Peripheral Interface	Y	Y	Y	Y	Data transmission up to 2.5 Mbits/s.
External Interrupt Inputs (3)	Y	Y	Y	Y	Selectable edge detection
External Memory Bus Expansion			Y	Y	Non-multiplexed address bus and data bus. Eliminates requirements for glue chips and saves board space
Max. Digital I/O	22	22	55	55	Provides the designer with multi-purpose ports for increased flexibility
Pin Count	28	28	68	68	Provides alternatives to meet the requirements of the application
Packaging	DIP/ PLCC	DIP/ PLCC	PLCC	PLCC	Supports high density surface mount

TMS370 APPLICATIONS

INDUSTRIAL

- Motor Control
- Temperature controllers
- Process control
- Meter control
- Medical Instrumentation
- Security systems

AUTOMOTIVE

- Climate control systems
- Cruise control
- Entertainment systems
- Instrumentation
- Navigational systems
- Engine control
- Sub-systems diagnostics

TELECOMMUNICATIONS

- Modems
- Intelligent phones
- Intelligent line card control
- Telecopiers
- Debit cards

COMPUTER

- Keyboards
- Peripheral interface control
- Disk controllers
- Terminals

1.2 TMS370 Architecture Overview

Figure 1-1 is a block diagram of the TMS370 architecture showing the major functions. The unshaded blocks and paths in the figure are features common to TMS370Cx10 and TMS370Cx50 devices. Shaded blocks and paths are present only on TMS370C050 and TMS370C850 devices.

The TMS370 family is based on a register-to-register architecture which allows access to a 256-byte Register File in a single bus cycle. On-chip memory includes a 4-kilobyte Program Memory (EEPROM or mask ROM), a 128/256-byte Static RAM, and a 256-byte Data EEPROM.

The versatile on-chip peripheral functions include (depending on the specific member of the series) an Analog-to-Digital converter (A/D), a Serial Communications Interface function (SCI), a Serial Peripheral Interface (SPI), up to three timers, and up to 55 digital Input/Output pins.

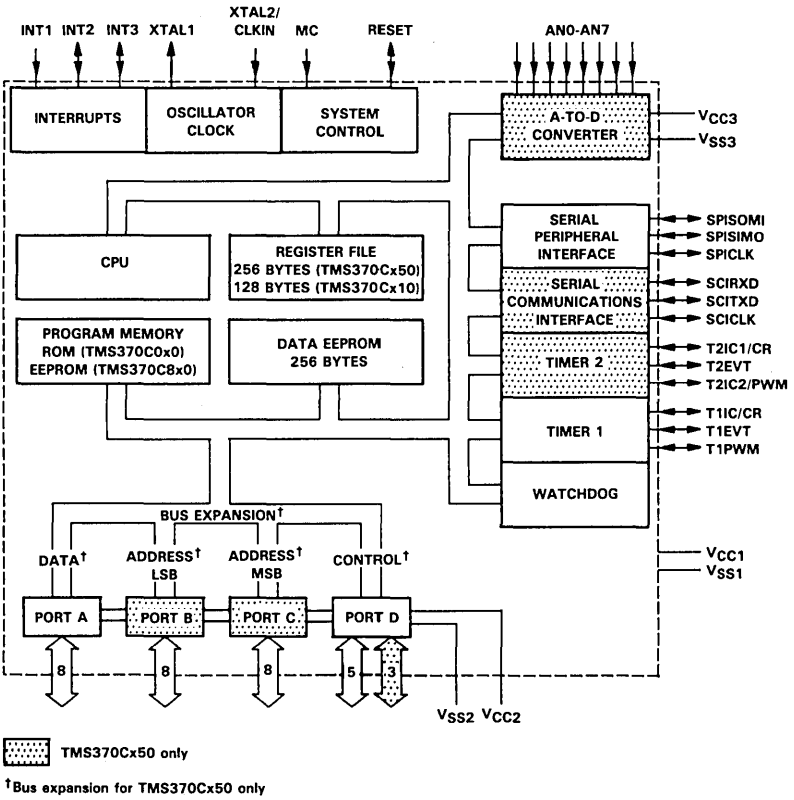


Figure 1-1. TMS370 Block Diagram

CPU

The TMS370 CPU is an 8-bit processor with Status register, Program Counter register, and Stack Pointer internal to the CPU module. The CPU uses the Register File as working registers, accessed on the internal bus in one bus cycle. The 8-bit internal bus also allows access to memory, and the peripheral interfaces. TMS370C050 and TMS370C850 devices allow external bus expansion through Ports A, B, C, and D.

REGISTER FILE

The Register File is located at the beginning of the TMS370 memory map. Register-access instructions in the TMS370 instruction set allow access to any of these registers in one bus cycle. This segment of the memory map is used as general purpose RAM and the Stack.

DATA EEPROM

The Data EEPROM module contains 256 bytes of Electrically Erasable Programmable Read Only Memory. This memory is useful for constants and infrequently changed variables required by the application program. The EEPROM can be programmed and erased using available Programmers or by the TMS370 itself under program control.

PROGRAM MEMORY

The Program Memory module contains four kilobytes of memory. In TMS370C810 and TMS370C850 devices the Program Memory is EEPROM and can be programmed, erased, and reprogrammed for prototype or small production runs. In TMS370C010 and TMS370C050 devices, the program memory is mask ROM, programmed at the factory.

INPUT/OUTPUT PORTS

TMS370C010 and TMS370C810 devices have two ports: Ports A and D. Both of these ports can be programmed, bit-by-bit, to function as either a digital input or a digital output.

TMS370C050 and TMS370C850 devices have four ports: Ports A, B, C, and D. These ports can be configured by software as the data, control, and address lines for an external bus. Any bits not needed for an external bus can be programmed to be either a digital input or a digital output.

WATCHDOG TIMER

The Watchdog Timer can be programmed to generate an interrupt when it times out. This function provides a hardware monitor over the software to prevent a "lost" program. If not needed as a watchdog, this timer can be used as a general purpose timer.

TIMER 1 and TIMER 2

These timers can be programmed to one of many configurations to count events, compare the counter contents to a preset value, or time-out after a preset interval. The results of these operations can generate an interrupt to the CPU, set flag bits, reset the timer counter, toggle an I/O line, or generate pulse-width-modulated (PWM) outputs.

SCI, SERIAL COMMUNICATIONS INTERFACE

The SCI module is a built-in serial interface which can be programmed to be asynchronous or isosynchronous. All timing, data format, and protocol factors are programmable and controlled by the SCI module in operation. The CPU takes no part in the serial communications except to write data to be transmitted to registers in the SCI and read received data from registers in the SCI when interrupted.

SPI, SERIAL PERIPHERAL INTERFACE

The SPI module is a built-in serial interface which facilitates communication between networked master and slave CPUs. As in the SCI, the SPI is setup by software and from then on, the CPU takes no part in timing, data format, or protocol. Also, as in 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.

A-TO-D CONVERTER

The A-to-D Converter module is an eight-channel, 8-bit, successive-approximation, analog-to-digital converter. The reference source and input channel are selectable. The conversion result can be programmed to be the ratio of the input voltage to the reference voltage or the ratio of one analog input to another. Input lines not required for A/D conversion can be programmed to be digital input lines.

1.3 Manual Organization

The following sections of this manual and their contents are summarized below.

Section 2: Family Devices

Presents the features of TMS370 family members including pinouts.

Sections 3 – 11

Describes the operation and programming of each major function in the TMS370 architecture.

Section 12: Instruction Set

Describes the TMS370 addressing modes and each of the 73 instructions including samples and examples.

Section 13: Design Aids

Gives sample interface circuits and programing examples.

Section 14: Development Support

Describes the hardware and software design-development tools available for the TMS370 series.

Section 15: Electrical Specifications

Gives timing diagrams and electrical specifications.

Section 16: Customer Information

Gives packaging, numbering, and ordering information.

Appendix A:

Gives reference tables and diagrams for TMS370 control bits.

Appendix B – D:

Give reference tables for the TMS370 character set, instruction set, and opcodes.

Appendix E: Glossary

Index

1.4 Symbols and Conventions

The following symbols and conventions are used in this manual.

SYMBOL	EXAMPLE	DESCRIPTION
(xxxxxx.n)	SPICTL.4	Bit location convention used in text, where 'xxxxxx' is the name of the register containing the bit and 'n' is the bit number (7=msb, 0=lsb).
(xx.n)	4A.0	Bit location convention used in figures, where 'xx' is the hexadecimal address of the peripheral register containing the bit and 'n' is the bit number (7 = msb, 0 = lsb).
h	1000h	Designates a number in the hexadecimal number system.
TMS370C0x0		Refers to TMS370C010 and TMS370C050 devices.
TMS370C8x0		Refers to TMS370C810 and TMS370C850 devices.
TMS370Cx10		Refers to TMS370C010 and TMS370C810 devices.
TMS370Cx50		Refers to TMS370C050 and TMS370C850 devices.
set		When used in reference to bits, means to write a logic 1 to the bit.
clear		When used in reference to bits, means to write a logic 0 to the bit.
P0n	P012	Hexadecimal Peripheral File (PF) address used in instructions accessing the PF.
Pn	P18	Decimal Peripheral File (PF) address used in instructions accessing the PF. (i.e., P18 = P012).
R0n	R010	Hexadecimal Register File (RF) address used in instructions accessing the RF.
Rn	R16	Decimal Register File (RF) address used in instructions accessing the RF. (i.e., R16 = R010)

1.5 Applicable Documents

1) *TMS370 Family Assembly Language Tools User's Guide,* SPNU010.
2) *TMS370/EEPROM Programmer's User's Guide,* SPNU011.
3) *TMS370 Family PC Debugger Interface User's Guide,* SPNU012
4) *TMS370 XDS/22 Quick Reference Card,* SPNU009
5) *TMS370C050/TMS370C850 8-Bit Microcontrollers Data Sheet,* SPNS010
6) *TMS370C010/TMS370C810 8-Bit Microcontrollers Data Sheet,* SPNS012

2. TMS370 Family Devices

This section discusses the features of the TMS370 family[1] of microcomputers. All family members are software compatible, allowing easy migration within the TMS370 family by maintaining a software base, development tools, and design expertise.

The TMS370 family devices are divided into two categories:

- **TMS370Cx10 devices** which include the TMS370C010 and TMS370C810
- **TMS370Cx50 devices** which include the TMS370C050 and TMS370C850

Both categories are supported by development tools that include the XDS, Assembler, and Linker.

This section begins with a summary and comparison of the TMS370 family devices, and then provides key features, pinouts, and pin descriptions for the individual categories.

[1] Throughout this manual, the term *TMS370* or *TMS370 family* refers to all members of the group.

2.1 Summary and Device Comparison.

The TMS370 family **CMOS** devices can be summarized as follows:

- The **TMS370C010** and **TMS370C810** are 8-bit, single-chip microcomputers, containing a CPU, a 16-bit timer, flexible I/O, a serial peripheral interface, 128 bytes of on-chip static RAM, and 256 bytes of data EEPROM. The **TMS370C010** also has 4K bytes of mask ROM program memory, while the **TMS370C810** has 4K bytes of EEPROM program memory.
- The **TMS370C050** and **TMS370C850** devices have the same basic features as the TMS370Cx10 with the addition of another 16-bit timer (timer 2), a serial communications interface, 128 bytes of on-chip static RAM (for a total of 256), memory expansion ports, and an eight channel A/D converter.
- Development tools include the **TMS370 XDS, Assembler, and Linker.**

Table 2-1. TMS370 Family Feature Summary

	TMS370C010	TMS370C810	TMS370C050	TMS370C850
Maximum Oscillator Freq.	20 MHz	20 MHz	20 MHz	20 MHz
Voltage	5 V ±10%	5 V ±10%	5 V ±10%	5 V ±10%
Operating temperature	-40°C to 85°C	-40°C to 85°C	-40°C to 85°C	-40°C to 85°C
Program Memory	4K ROM	4K EEPROM	4K ROM	4K EEPROM
Internal RAM	128 bytes	128 bytes	256 bytes	256 bytes
Data EEPROM	256 bytes	256 bytes	256 bytes	256 bytes
Modules				
SPI	Yes	Yes	Yes	Yes
Timer 1	Yes	Yes	Yes	Yes
Watchdog timer	Yes	Yes	Yes	Yes
Timer 2	No	No	Yes	Yes
SCI	No	No	Yes	Yes
A/D Port	No	No	Yes	Yes
I/O Lines:				
Bidirectional	22	22	46	46
Input only	1	1	9	9
Memory Expansion	No	No	Yes	Yes
Interrupts/Reset				
External	4	4	4	4
Vectors total	6	6	10	10
Sources total	13	13	23	23
Package Type	28-pin DIP 28-pin PLCC	28-pin DIP 28-pin PLCC	64-pin PLCC	64-pin PLCC

2.2 TMS370Cx10 Features

The key features of the TMS370Cx10 devices are as follows:

- CMOS EEPROM Technology
  - EEPROM programming with single 5-volt supply
- Flexible operating features
  - Power reduction STANDBY and HALT modes
  - -40C to 85C operating temperature
  - 2 MHz to 20 MHz input clock frequency
  - 5-volt supply (V_{CC})
  - Wake-up power-down mode
- Memory-mapped ports for easy addressing
- 14 addressing modes using eight formats, including:
  - Register-to-register arithmetic
  - Indirect addressing
  - Indexed and indirect branches and calls
- 16-bit general-purpose timer, software configurable as:
  - 16-bit event timer
  - 16-bit pulse accumulator
  - 16-bit input-capture function
  - Two compare registers
  - Self contained PWM output function
- On-chip 24-bit watchdog timer
- Serial peripheral interface (SPI)
  - Variable-length high-speed shift register
  - Synchronous master/slave operation
  - Error detection flags
- Flexible interrupt handling
  - Two software programmable interrupt levels
  - Programmable rising or falling edge detect
- System integrity features:
  - Oscillator fault detection
  - Privileged mode lockout
  - Watchdog timer (24-bit)

2.3 TMS370Cx50 Features.

Table 2-2. TMS370Cx50 Feature Summary

FEATURE	TMS370C010	TMS370C810	TMS370C050	TMS370C850
Maximum Oscillator Freq.	20 MHz	20 MHz	20 MHz	20 MHz
Voltage	5 V ±10%	5 V ±10%	5 V ±10%	5 V ±10%
Operating temperatures	-40°C to 85°C 0°C to 70°C	-40°C to 85°C 0°C to 70°C	-40°C to 85°C 0°C to 70°C	-40°C to 85°C 0°C to 70°C
Program Memory	4K ROM	4K EEPROM	4K ROM	4K EEPROM
Internal RAM	128 bytes	128 bytes	256 bytes	256 bytes
Data EEPROM	256 bytes	256 bytes	256 bytes	256 bytes
Modules SPI Timer 1 Watchdog timer Timer 2 SCI A/D Converter	 Yes Yes Yes No No No	 Yes Yes Yes No No No	 Yes Yes Yes Yes Yes Yes	 Yes Yes Yes Yes Yes Yes
I/O Lines: Bidirectional Input only	 22 1	 22 1	 46 9	 46 9
Memory Expansion	No	No	Yes	Yes
Interrupts/Reset External Vectors total Sources total	 4 6 13	 4 6 13	 4 10 23	 4 10 23
Package Type	28-pin DIP 28-pin PLCC	28-pin DIP 28-pin PLCC	64-pin PLCC	64-pin PLCC

The TMS370Cx50 devices contain all the features of the TMS370Cx10 devices plus additional capabilities. The following features are common to all TMS370Cx50 and TMS370Cx10 devices:

- CMOS EEPROM Technology
  - EEPROM programming with single 5-volt supply
- Flexible operating features
  - Power reduction STANDBY and HALT modes
  - -40C to 85C operating temperature
  - 2 MHz to 20 MHz input clock frequency
  - 5-volt supply (V_{CC})
  - Wake-up power-down mode
- Memory-mapped ports for easy addressing

- 14 addressing modes using eight formats, including:
  - Register-to-register arithmetic
  - Indirect addressing
  - Indexed and indirect branches and calls
- 16-bit general purpose timer - software configurable as:
  - 16-bit event timer
  - 16-bit pulse accumulator
  - 16-bit input-capture function
  - Two compare registers
  - Self contained PWM output function
- On-chip 24-bit watchdog timer
- Serial peripheral interface (SPI)
  - Variable-length high-speed shift register
  - Synchronous master/slave operation
  - Error detection flags
- Flexible interrupt handling
  - Two software programmable interrupt levels
  - Programmable rising or falling edge detect
- System integrity features:
  - Oscillator fault detection
  - Privileged mode lockout
  - Watchdog timer (24-bit)

The following features are unique to the TMS370Cx50 devices:

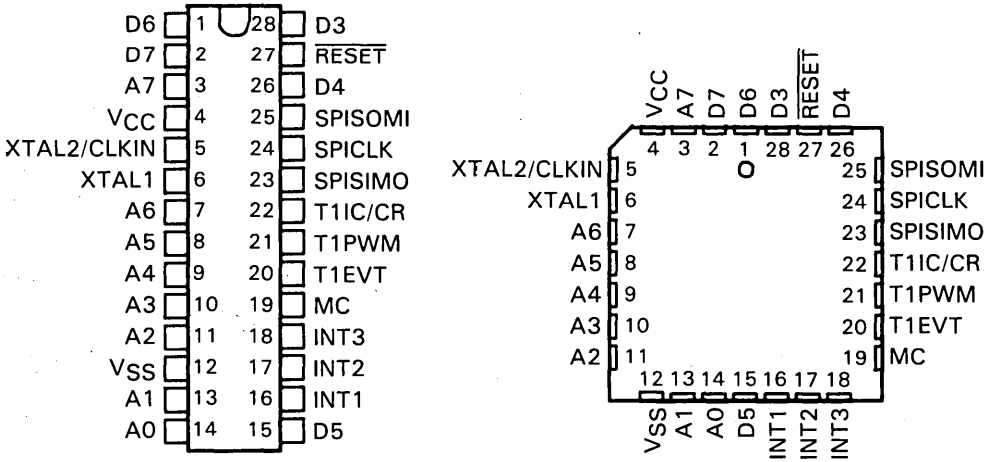
- Eight channel A/D converter
- 2nd 16-bit general purpose timer
- Serial communications interface (SCI)
  - Asynchronous and Isosynchronous modes
  - Full duplex, double buffered Rx and Tx
- Additional 128 bytes of on-chip RAM (256 bytes total)
- Flexible system memory configurations
  - Precoded external chip select outputs
  - Programmable external memory/peripheral WAIT states
  - Addressable memory expansion to over 112K bytes
  - No logic needed for external memory addressing
  - WAIT line to extend bus cycles

2

2.4 TMS370 Family Pinouts/Pin Descriptions

2.4.1 TMS370Cx10 Pinouts

The pinouts for the TMS370Cx10 devices are shown below.



A. 28-Pin DIP **B. 28-Pin PLCC**

Figure 2-1. Pinouts for TMS370C010, and TMS370C810

2.4.2 TMS370Cx10 Pin Descriptions

Table 2-3. TMS370Cx10 Pin Descriptions

Pin Name	Pin No.	I/O	Description
A0	14	I/O	Port A is a general purpose bidirectional I/O port.
A1	13	I/O	
A2	11	I/O	
A3	10	I/O	
A4	9	I/O	
A5	8	I/O	
A6	7	I/O	
A7	3	I/O	
D3	28	I/O	Port D is a general purpose bidirectional I/O port.
D4	26	I/O	
D5	15	I/O	
D6	1	I/O	
D7	2	I/O	
INT1	16	I	External non-maskable or maskable interrupt/General purpose input pin.
INT2	17	I/O	External maskable interrupt input/General purpose bidirectional pin.
INT3	18	I/O	External maskable interrupt input/General purpose bidirectional pin.
T1IC/CR	22	I/O	Timer 1 Input Capture/Counter Reset input pin/General purpose bidirectional pin.
T1PWM	21	I/O	Timer 1 PWM output pin/General purpose bidirectional pin.
T1EVT	20	I/O	Timer 1 external Event input pin/General purpose bidirectional pin.
SPISOMI	25	I/O	SPI Slave Output pin. Master Input pin/General purpose bidirectional pin.
SPISIMO	23	I/O	SPI Slave Input pin, Master Output pin/General purpose bidirectional pin.
SPICLK	24	I/O	SPI bidirectional Serial Clock pin/General purpose bidirectional pin.
RESET	27	I/O	System reset bidirectional pin. As an input it initializes microcontroller, as an open-drain output it indicates an internal failure was detected by the Watchdog or Oscillator Fault circuit.
MC	19	I	Mode control input pin; enables EEPROM Write Protection Override (WPO) mode. Normal operation = 0V, WPO = 12V.
XTAL2/CLKIN	5	I	Internal oscillator crystal input/External clock source input.
XTAL1	6	O	Internal oscillator output for crystal.
Vcc	4		Positive supply voltage
Vss	12		Ground reference

NOTE: Each pin associated with Interrupt 2, Interrupt 3, Timer 1, and SPI functional blocks may be individually programmed as a general purpose bidirectional pin if it is not used for it's primary block function.

2.4.3 TMS370Cx50 Pinouts

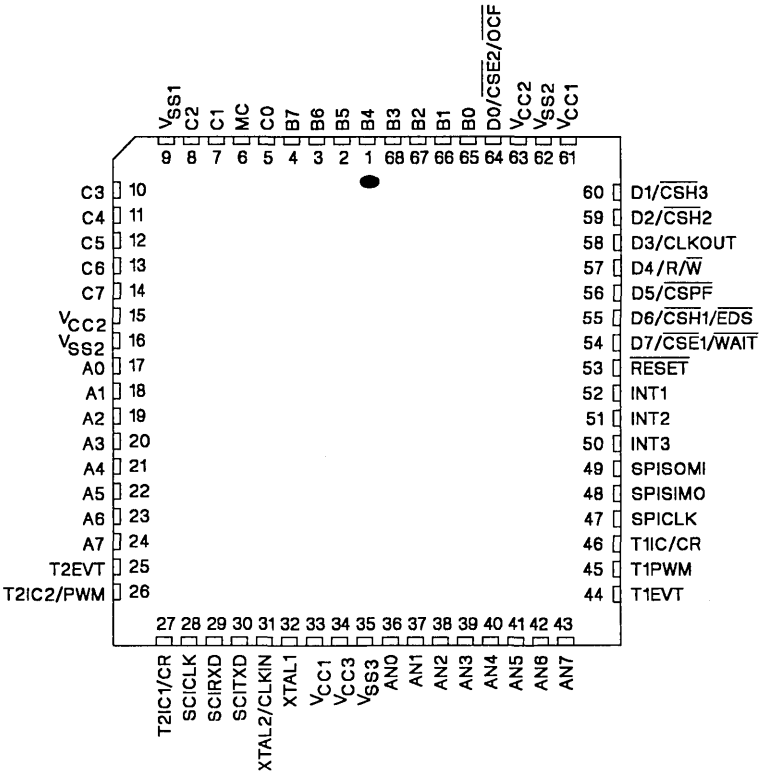


Figure 2-2. Pinouts for TMS370C050 and TMS370C850

2.4.4 TMS370Cx50 Pin Descriptions

Table 2-4. TMS370Cx50 Pin Descriptions

Pin				
Name	Alternate Function	No.	I/O	Description
A0	DATA0 (LSB)	17	I/O	Single-chip mode: Port A is a general purpose bidirectional port. Expansion mode: Port A may be individually programmed as the external bidirectional data bus (DATA0-DATA7).
A1	DATA1	18	I/O	
A2	DATA2	19	I/O	
A3	DATA3	20	I/O	
A4	DATA4	21	I/O	
A5	DATA5	22	I/O	
A6	DATA6	23	I/O	
A7	DATA7 (MSB)	24	I/O	
B0	ADD0	65	I/O	Single chip mode: Port B is a general purpose bidirectional I/O port. Expansion modes: Port B may be individually programmed as the low order address output bus (ADD0-ADD7).
B1	ADD1	66	I/O	
B2	ADD2	67	I/O	
B3	ADD3	68	I/O	
B4	ADD4	1	I/O	
B5	ADD5	2	I/O	
B6	ADD6	3	I/O	
B7	ADD7	4	I/O	
C0	ADD8	5	I/O	Single chip mode: Port C is a general purpose bidirectional I/O port. Expansion mode: Port C may be individually programmed as the high order address output bus (ADD8-ADD15).
C1	ADD9	7	I/O	
C2	ADD10	8	I/O	
C3	ADD11	10	I/O	
C4	ADD12	11	I/O	
C5	ADD13	12	I/O	
C6	ADD14	13	I/O	
C7	ADD15	14	I/O	
INT1	INTIN	52	I	External interrupt (non-maskable or maskable)/ General purpose input pin.
INT2	INTIO1	51	I/O	External maskable interrupt input/General purpose bidirectional pin.
INT3	INTIO2	50	I/O	External maskable interrupt input/General purpose bidirectional pin.

Table 2-4. TMS370Cx50 Pin Descriptions (Continued)

Pin Name	Pin Alternate Function		Pin No.	I/O	Description
	Function A	Function B			Single chip mode: Port D is a general purpose bidirectional I/O port. Each of the Port D pins can be individually configured as either a general purpose I/O pin, primary memory control signal (Function A), or secondary memory control signal (Function B). All chip selects are independent and can be used for memory bank switching.
D0	$\overline{CSE2}$	OCF	64	I/O	I/O pin/A: Chip Select Eighth output 2 goes low during memory accesses to 2000h-3FFFh /B: Opcode fetch goes low during the opcode fetch memory cycle.
D1	$\overline{CSH3}$		60	I/O	I/O pin/A: Chip Select Half output 3 goes low during memory accesses to 8000h-FFFFh.
D2	$\overline{CSH2}$		59	I/O	I/O pin/A: Chip Select Half output 2 goes low during memory accesses to 8000h-FFFFh.
D3	CLK-OUT	CLK-OUT	58	I/O	I/O pin/A, B: Internal clock signal is 1/4 XTAL2/CLKIN frequency.
D4	R/$\overline{W}$	R/$\overline{W}$	57	I/O	I/O pin/A, B: Read/Write output pin.
D5	$\overline{CSPF}$		56	I/O	I/O pin/A: Chip Select Peripheral output for peripheral file goes low during memory accesses to 10C0h-10FFh.
D6	$\overline{CSH1}$	$\overline{EDS}$	55	I/O	I/O pin/A: Chip Select Half output 1 goes low during memory accesses to 8000h-FFFFh /B: External Data Strobe output goes low during memory accesses from external memory and has the same timings as the five chip selects.
D7	$\overline{CSE1}$	$\overline{WAIT}$	54	I/O	I/O pin/A: Chip Select Eighth output goes low during memory accesses to 2000h-3FFFh /B: Wait input pin extends bus signals.
	Alternate Function				
T1IC/CR	T1IO1		46	I/O	Timer 1 Input Capture/Counter Reset input pin/General purpose bidirectional pin.
T1PWM	T1IO2		45	I/O	Timer 1 PWM output pin/General purpose bidirectional pin.
T1EVT	T2IO3		44	I/O	Timer 1 External Event input pin/General purpose bidirectional pin.
T2IC1/CR	T2IO1		27	I/O	Timer 2 Input Capture 1/Counter Reset input pin/General purpose bidirectional pin.
T2IC2/PWM	T2IO2		26	I/O	Timer 2 Input Capture 2/PWM output pin/ General purpose bidirectional pin.
T2EVT	T2IO3		25	I/O	Timer 2 External Event input pin/General purpose bidirectional pin.

Pin				
Name	Alternate Function	No.	I/O	Description
SPISOMI	SPII01	49	I/O	SPI Slave Output pin, Master Input pin/ General purpose bidirectional pin.
SPISIMO	SPII02	48	I/O	SPI Slave Input pin. Master Output pin/ General purpose bidirectional pin.
SPICLK	SPII03	47	I/O	SPI bidirectional Serial Clock pin/ General purpose bidirectional pin.
SCITXD	SCII01	30	I/O	SCI Transmit Data output pin/General purpose bidirectional pin.
SCIRXD	SCII02	29	I/O	SCI Receive Data Input pin/General purpose bidirectional pin.
SCICLK	SCII03	28	I/O	SCI bidirectional Serial Clock pin/ General purpose bidirectional pin.
AN0	E0	36	I	A/D analog input (AN0-AN7) or positive reference pins (AN1-AN7).
AN1	E1	37	I	
AN2	E2	38	I	Port E may be individually programmed as general purpose input pins if not used as A/D converter analog input or positive reference input.
AN3	E3	39	I	
AN4	E4	40	I	
AN5	E5	41	I	
AN6	E6	42	I	
AN7	E7	43	I	
Vcc3		34		A/D converter positive supply voltage and optional positive reference input pin.
Vss3		35		A/D converter ground supply and low reference input pin.
RESET		53	I/O	System reset bidirectional pin. As an input it initializes microcontroller, as open-drain output it indicates an internal failure was detected by the Watchdog or Oscillator Fault circuit.
MC		6	I	Microprocessor/Microcomputer mode control pin, also enables EEPROM Write Protection Override (WPO) mode.
XTAL2/ CLKIN		31	I	Internal oscillator crystal input/External clock source input.
XTAL1		32	O	Internal oscillator output for crystal.
Vcc1		33,61		Positive supply voltage for digital logic.
Vcc2		15,63		Positive supply voltage for digital I/O pins.
Vss1		9		Ground reference for digital logic.
Vss2		16,62		Ground reference for digital I/O pins.

NOTE. Each pin associated with the Interrupt, Timer 1, Timer 2, SPI, and SCI functional blocks may be individually programmed as a general purpose bidirectional pin if it is not used for its primary block function.

3. CPU and Memory Organization

This section describes the CPU registers and memory organization. In the TMS370 register-to-register architecture, the CPU and RAM act as a single unit along with the Program Counter, Stack Pointer, and Status Register.

The following conventions are used in this section when discussing specific members of the TMS370 family:

- TMS370C0x0 refers to TMS370C010 and TMS370C050 devices.
- TMS370C8x0 refers to TMS370C810 and TMS370C850 devices.
- TMS370Cx10 refers to TMS370C010 and TMS370C810 devices.
- TMS370Cx50 refers to TMS370C050 and TMS370C850 devices.

This section covers the following topics:

3.1 CPU/Register File Interaction

The first 256 address locations in the memory space, 0000h through 00FFh (0000h–007Fh for TMS370Cx10 devices), are called the *Register File*. Any location in this block can be accessed as: a general purpose register, data memory storage, program instructions, or part of the stack.

Figure 3-1 illustrates the multiple use of the Register File. For example, memory address 0004h can also be treated as register R4. Or, the stack pointer could be loaded with the address 0004h and the stack would start at the next location.

Registers R0 and R1 are also called 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.

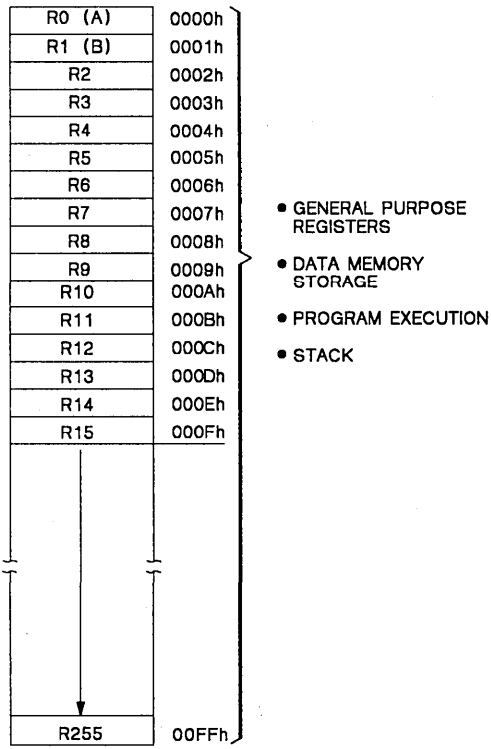


Figure 3-1. Register File

This multiple use of the Register File gives designers the flexibility to use the Register File however they wish. The partitioning of the Register File is determined by the value loaded into the stack pointer and the use of the Register File by the program.

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 paragraphs.

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) using the assembly language instruction LDSP. The LDSP instruction allows the stack to be located anywhere in the Register File space. The SP may be read into Register B using the STSP 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.
```

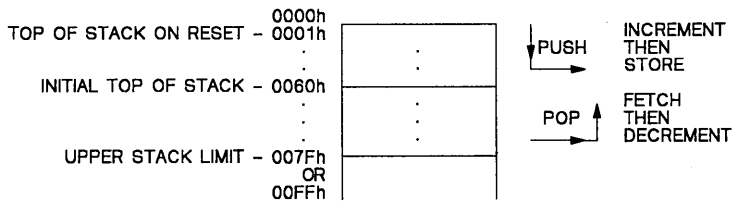


Figure 3-2. Stack Example

For TMS370Cx50 devices, if the stack is pushed beyond its limit of 00FFh, the SP register wraps around from 00FFh to 0000h without an error indication. The stack for TMS370Cx10 devices is not implemented beyond 7Fh; data pushed beyond this limit is lost. The application program must guard against stack overflow.

3.2.2 Status Register (ST)

The ST register includes four status bits and two interrupt enable bits. The four status bits indicate the outcome of the previous instruction. Conditional instructions (for example, the conditional jump instructions) use these status bits to determine program flow. The two interrupt bits control the two interrupt levels. The ST register, status bit notation, and status bit definitions are as follows:

Status Register (ST)

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

R=Read, W=Write, -n= Value after RESET

Bits 0-1 - Reserved. Read data is indeterminate.

Bit 2 - **IE1**. Level 1 Interrupt Enable.
This bit controls interrupt level 1 (highest priority).
0 = disable interrupt requests from priority level 1.
1 = enable interrupt requests from priority level 1.

Bit 3 - **IE2**. Interrupt Enable, Chain 2.
This bit controls interrupt level 2 (lowest priority).
0 = disable interrupt requests from priority level 2.
1 = enable interrupt requests from priority level 2.

Bit 4 - **V**. Overflow.
Set by the CPU if an arithmetic overflow condition was detected during the previous instruction. The value of this flag is significant at the completion of the following instructions: ADC, ADD, SUB, SBB, CMP, DIV.

Instruction	V
ADC, ADD, INC, INCW	(C XOR N) AND (Bit 7{s} XNOR Bit 7{d})
CMP, DEC, SUB, SBB	(C XOR N) AND (Bit 7{s} XOR Bit 7{d})
DIV	1 if Rn $\leq$ A, which means quotient > 255

Bit 5 - **Z**. Zero.
Set by the CPU if the result of the previous operation was 0; cleared otherwise.

Bit 6 - **N**. Negative.
CPU sets this bit to the value of the most significant bit (sign bit) of the result of the previous operation.

Bit 7 - **C**. Carry.
This status bit is set by arithmetic instructions as a carry bit or as a no-borrow bit. It is also effected by the rotate instructions. See each instruction in Section Section 12 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 on interrupt effects on the Status Register, see Section 5.1.1). The normal exit from an interrupt service routine is made with the RTI instruction. When the CPU executes the RTI instruction, it automatically restores the content 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 which manipulates or moves data. Thus, 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
- CALL
- CALLR
- BR
- DJNZ
- JMP
- Conditional Jump instructions
- IDLE,
- NOP
- PUSH ST
- RTS
- STSP
- JMPL
- 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 a "LDST #iop8" opcode by the assembler so that "#iop8" is the appropriate value to set or clear the specific interrupt (see section Section 12 for more information on the LDST instruction).

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

3.2.3 Program Counter

The contents of the Program Counter (PC) point to the memory location of the next instruction to be executed. The PC consists of two 8-bit registers in the CPU: the Program Counter High (PCH) and Program Counter Low (PCL). These registers contain the MSB and LSB of a 16-bit address.

During RESET, the PCH (MSB of the PC) is loaded with the contents of memory location 7FFEh and the PCL (LSB of the PC) is loaded with the contents of memory location 7FFFh. Figure 3-3 illustrates this operation using an example value of 7000h as the contents of memory locations 7FFEh and 7FFFh (Reset vector).

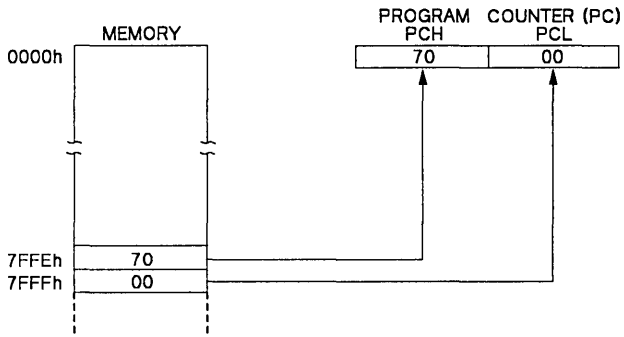


Figure 3-3. Program Counter After Reset

3.3 Memory Map

Figure 3-4 shows the memory maps of TMS370Cx50 and TMS370Cx10 devices. The partitioning of memory and the 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 TMS370Cx50 device can be programmed to use the 16 address bits to access up to 64 kilobytes of memory. In addition, memory expansion features allow up to 112 kilobytes of external memory. (The expansion features are described further in Section 3.4.2.)

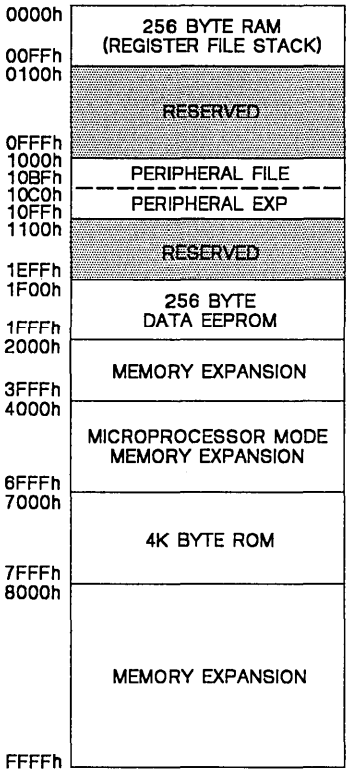


Figure 3-4. TMS370 Memory Map

The following paragraphs describe each block of the memory map.

3.3.1 Register File

The beginning addresses of the memory map (0000h–00FFh for TMS370Cx50 devices or 0000h–007Fh for TMS370Cx10 devices) are on-chip RAM called the Register File (RF). In TMS370Cx50 devices, the RF has 256 bytes of memory treated as registers R0 through R255. In TMS370Cx10 devices, the RF has 128 bytes of memory treated as registers R0 through R127.

The first two registers, R0 and R1, are also called Register A and Register B, respectively. The memory addresses of these registers are given in Figure 3-5.

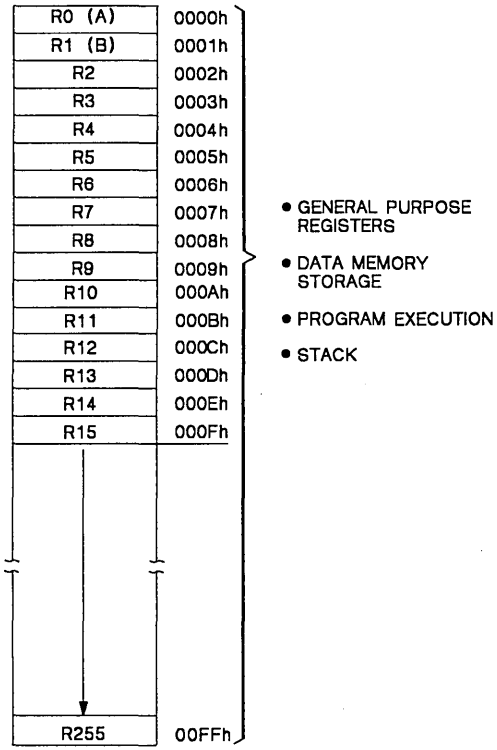


Figure 3-5. Register File Addresses

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

Therefore, any location in the register file can be accessed by one of three ways.

1) Normal memory access using a hexadecimal address. For example,

```
MOV   A,0006(B);Move the contents of Register A to
                ; memory location 0006h indexed by B.
```

2) Register access using the register number. For example,

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

3) 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.
```

Access time to the Register File, when used as a general purpose register, is a single system clock cycle. Any other access to the Register File takes two clock cycles.

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

The Halt, Idle, and Standby states have no effect on the contents of the Register File.

3.3.2 Peripheral File

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

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

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

Table 3-1 gives the address map for the Peripheral File.

Table 3-1. Peripheral File Address Map

FRAME NO.	ADDRESS	DESCRIPTION	TMS370Cx50	TMS370Cx10
0	1000h	Reserved for factory test	---	---
1	1010h	System and EEPROM control registers	Yes	Yes
2	1020h	Digital I/O port control registers	Yes	Yes
3	1030h	SPI registers	Yes	Yes
4	1040h	TIMER1 registers	Yes	Yes
5	1050h	SCI registers	Yes	NA
6	1060h	TIMER2 registers	Yes	NA
7	1070h	A-to-D registers	Yes	NA
8	1080h	Reserved	NA	NA
9	1090h	Reserved	NA	NA
10	10A0h	Reserved	NA	NA
11	10B0h	Reserved	NA	NA
12	10C0h	External Peripheral control	Yes	NA
13	10D0h	External Peripheral control	Yes	NA
14	10E0h	External Peripheral control	Yes	NA
15	10F0h	External Peripheral control	Yes	NA

NA - Not Available

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 programming. EEPROM module control registers are described in Section 6.

Frame 2 (1020h-102Fh) contains the Digital I/O Pin configuration/control registers. The individual functions controlled by these registers are described in Section 4.2, page 4-11.

Frames 3 through 7 are used by the internal peripherals. These peripherals and their control registers are described in the following sections.

- SPI registers - Section 10
- Timer 1 registers - Section 7
- SCI registers - Section 9
- Timer 2 registers - Section 8
- A-to-D registers - Section 11

Frames 8 through 11 are reserved.

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

3.3.3 Data EEPROM Module

The Data EEPROM module is a 256 byte array at memory locations 1F00h through 1FFFh. This 256 byte array is configured into 8 blocks of 32 bytes. Each block can be individually write protected. This module can be programmed on either a byte-wide or single-bit basis. Read-access time for the EEPROM module is two system clock cycles.

Programming of the Data EEPROM array is controlled by the Data EEPROM Control Register (DEECTL) at memory address 101Ah and a Write Protection Register (WPR) at memory address 1F00h. EEPROM programming commands are controlled through these registers. See Section 6.1.1.1 and Section 6.1.1.2 for more details on the WPR and DEECTL registers.

3.3.4 Program Memory

The Program Memory is arranged as individually-addressable bytes located at 7000h through 7FFFh in the memory map. Data may be read or code may be executed directly from these locations.

Memory addresses 7FECh through 7FFFh are reserved for interrupt and reset vectors. Trap vectors, used with TRAP0 through TRAP15 instructions, are at addresses 7FC0h through 7FDFh. Table 3-2 gives the memory map for the reserved vector locations.

The Program Memory may be either ROM or EEPROM depending the specific member of the TMS370 family. The differences are described in the paragraphs following Table 3-2.

Table 3-2. Vector Address Map

ADDRESS	DESCRIPTION	TMS370Cx50	TMS370Cx10	NO. OF BYTES
7FC0h	Trap 0-15	Yes	Yes	32
7FE0h	Reserved	NA	NA	12
7FECh	A-D Converter	Yes	NA	2
7FEEh	Timer 2	Yes	NA	2
7FF0h	Serial Communications Interface TX	Yes	NA	2
7FF2h	Serial Communications Interface RX	Yes	NA	2
7FF4h	Timer 1	Yes	Yes	2
7FF6h	Serial Peripheral Interface	Yes	Yes	2
7FF8h	Interrupt 3	Yes	Yes	2
7FFAh	Interrupt 2	Yes	Yes	2
7FFCh	Interrupt 1	Yes	Yes	2
7FFEh	Reset	Yes	Yes	2

NA - Not Available

3.3.4.1 Program ROM Module (TMS370C0x0 devices only)

The Program ROM module consists of read-only memory which is programmed at the time of device fabrication. All accesses to the ROM module requires 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 the customer's software algorithm, nor should it be used during mask ROM/firmware development.
> **The contents of the reserve locations are changed by TI.**

3.3.4.2 Program EEPROM Module (TMS370C8x0 devices only)

The Program EEPROM module replaces the Program ROM for systems in prototype or small production runs. The module consists of 4 kilobytes of EEPROM (7000h–7FFFh) and the necessary programming control logic.

The Program EEPROM Control Register (PEECTL) is located at memory location 101Ch in the Peripheral File.

Read access to the Program EEPROM is performed as normal memory read cycles. Write cycles require a special sequence of events. This sequence is the same as that for the Data EEPROM. See Section 6.2.2 for a detailed discussion of programming the EEPROM Modules.

The EEPROM can be written to only when the microcomputer is operating under Write Protect Override (WPO), which is set by applying 12 volts to the MC pin.

3.4 Memory Operating Modes

TMS370Cx50 devices have four memory operating modes.

- Microcomputer modes
  - microcomputer single-chip mode
  - microcomputer with external expansion
- Microprocessor modes
  - microprocessor without internal program memory
  - microprocessor with internal program memory

TMS370Cx10 devices have no memory expansion capability and operate only in the microcomputer, single-chip mode.

For TMS370Cx50 devices, the basic microcomputer and microprocessor operating modes are selected by the voltage level applied to the dedicated MC pin when the $\overline{\text{RESET}}$ pin goes inactive (high).

If the MC pin is low when the $\overline{\text{RESET}}$ signal goes high then the processor enters the microcomputer mode. If the MC pin is high when the RESET signal goes high, then it enters the microprocessor mode. Changing the MC pin alone will not change the memory mode. To change memory operating mode, change the MC pin and then reset the device.

Applying 12 volts to the MC pin *after* Reset forces the device to enter the Write Protect Override (WPO) mode.

> **Note:**
>
> If 12 volts is applied to the MC pin when the RESET pin goes from low to high, the results are unpredictable.

If the processor resets into a microcomputer mode, the software can change the internal system configuration registers to select the desired memory expansion configuration. Part of this configuration setup 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 may 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 these modes are described in the following paragraphs.

3.4.1 Microcomputer Single-Chip Mode (all TMS370 devices)

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

During reset the MC pin must remain at a low level in order to successfully enter the microcomputer mode. While operating in the single-chip mode, external circuitry may place 12 volts on the MC pin to enter the Write Protect Override (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 TMS370Cx10 devices even though they operate only in the Microcomputer, Single Chip Mode.

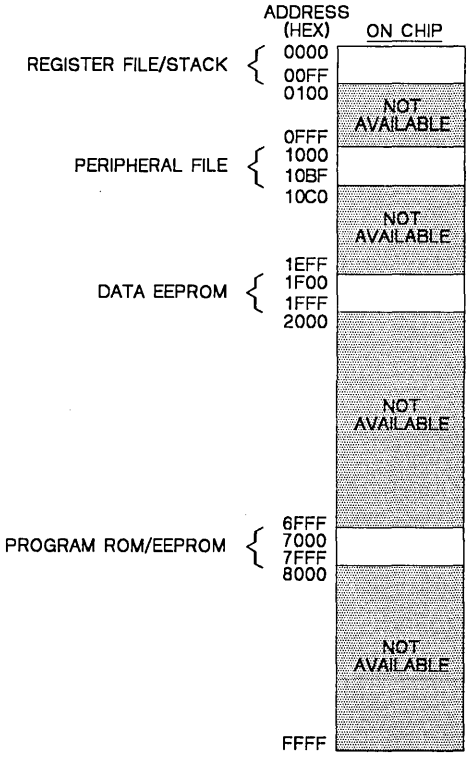


Figure 3-6. Microcomputer, Single Chip Mode

3.4.2 Microcomputer Mode w/External Expansion (TMS370Cx50)

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

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

If it is not necessary to use the entire address, data, or control bus, then each unused pin 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 4.2 for further information on programming I/O pins).

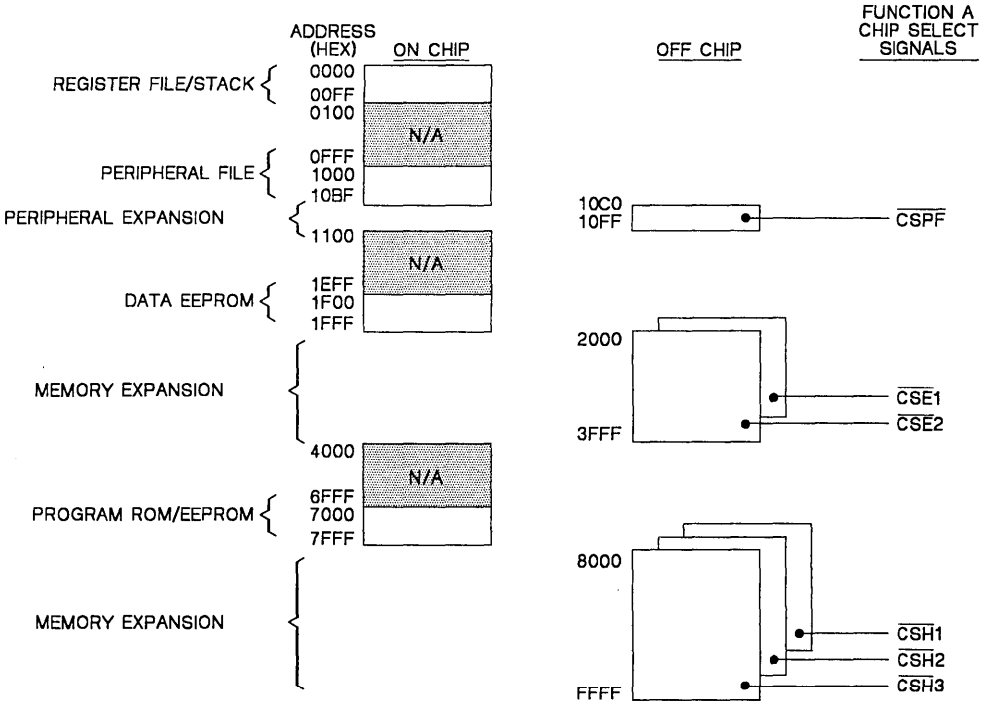
The address bus and data bus are non-multiplexed, eliminating the requirement for an external address/data latch, thereby lowering system cost. External interface decode logic can be reduced further by using the precoded chip select outputs. The Port D outputs can be programmed, on a pin-by-pin basis, to provide direct memory/peripheral chip selection or chip enable functions.

Each Port D pin can be individually set to Function A, Function B or general purpose I/O. When Port D is set up to drive the chip selection signals (Function A), a memory access to any location between 2000h and 3FFFh, activates pins $\overline{\text{CSE}}$1 and $\overline{\text{CSE}}$2. Typically, an application that uses both $\overline{\text{CSE}}$1 and $\overline{\text{CSE}}$2 sets one as the active chip-select function and sets the other as a general-purpose high-level output.

Similarly, a memory access to any location between 8000h and FFFFh activates $\overline{\text{CSH}}$1, $\overline{\text{CSH}}$2, and $\overline{\text{CSH}}$3 if enabled by the appropriate port control registers. The $\overline{\text{CSH}}$1, $\overline{\text{CSH}}$2, and $\overline{\text{CSH}}$3 signals can be used as memory bank select signals under software control. As a result, up to 96 kilobytes of external memory can be mapped into the 32-kilobyte logical address space of 8000h–FFFFh as shown in Figure 3-7.

The $\overline{\text{CSPF}}$ pin is activated, if enabled, during accesses to the upper 4 frames (memory addresses 10C0h–10FFh) of the peripheral file. This signal can be used as a chip select for external expansion of the Peripheral File.

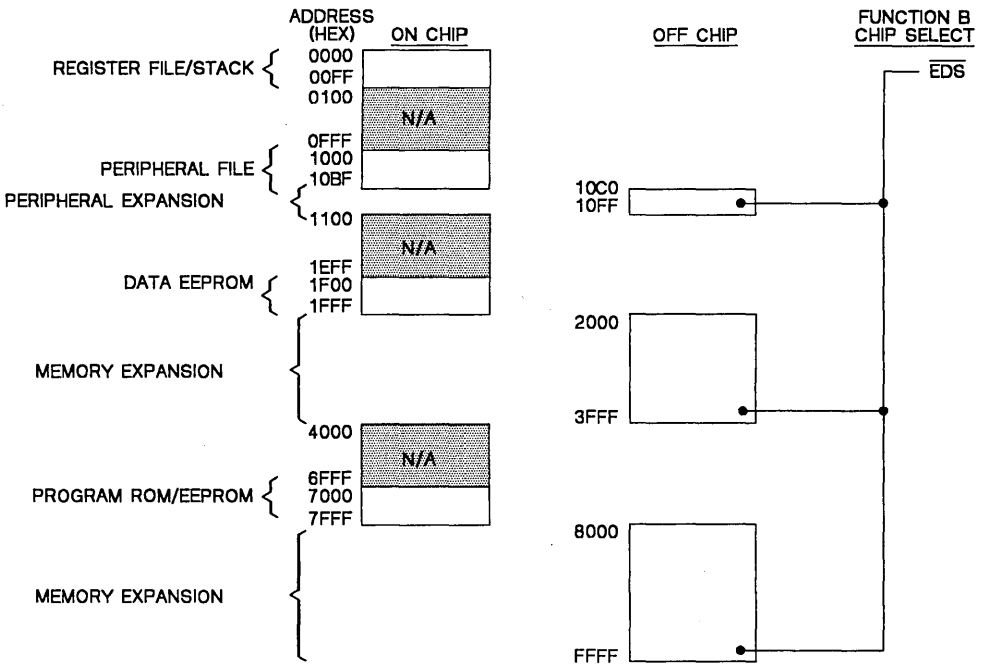
> **Note:**
>
> Applications that use more than one chip-select signal for the same address should set the unused chip-selects (i.e., chip-selects not currently used to select memory banks) to general-purpose high-level outputs.



N/A - NOT AVAILABLE

Figure 3-7. Microcomputer Mode with Function A Expansion

All predecoded chip selects have the same timing as the External Data Strobe ($\overline{EDS}$) signal (see Section 15, Electrical Specifications). $\overline{EDS}$ is a Function B (microprocessor mode) signal which 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.



N/A – NOT AVAILABLE

Figure 3-8. Microcomputer Mode with Function B Expansion

See Section 4.2 for a description of the Digital I/O port control registers and how the chip select signals are enabled.

To put a TMS370Cx50 device into the Microcomputer Mode with External Expansion:

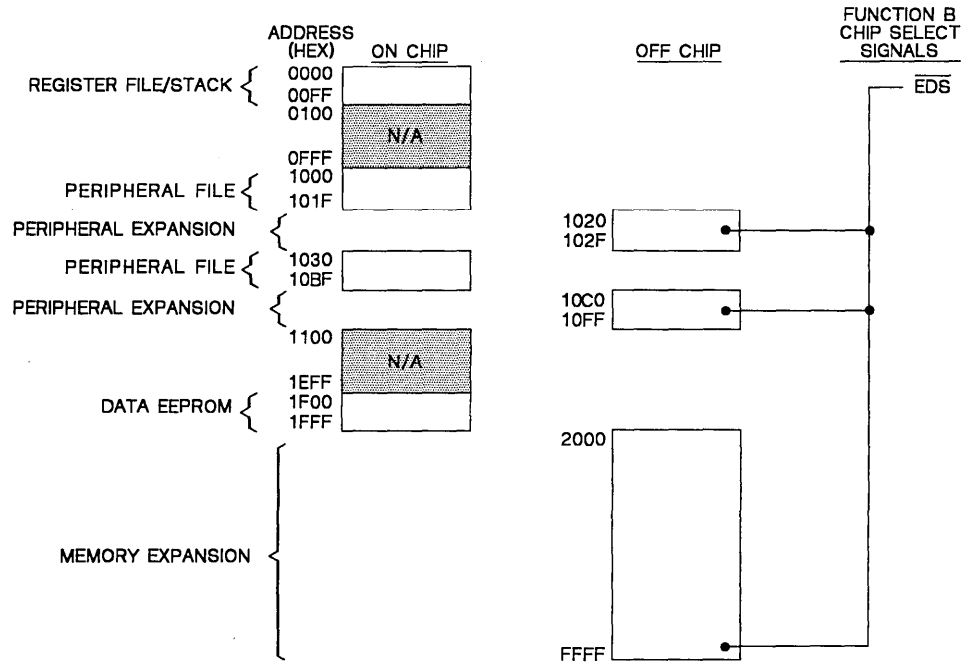
1) Place a low logic level on the MC pin.
2) Take the $\overline{\text{RESET}}$ pin active low, then return $\overline{\text{RESET}}$ to its inactive high state.
3) Program the Digital I/O registers to select the chip select or control signals needed (Function A or Function B).

3.4.3 Microprocessor Mode without Internal Memory
(TMS370Cx50 only)

When a TMS370Cx50 device is activated in the microprocessor mode, the Register File and data EEPROM remain active, but the on-chip Program ROM or EEPROM 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 TMS370Cx50 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.



N/A–NOT AVAILABLE

Figure 3-9. Microprocessor Mode without Internal Memory

To put a TMS370Cx50 device into the Microprocessor Mode without Internal Memory:

1) Place a high logic level on the MC pin.
2) Take the $\overline{\text{RESET}}$ pin active low, then return $\overline{\text{RESET}}$ to its inactive high state.

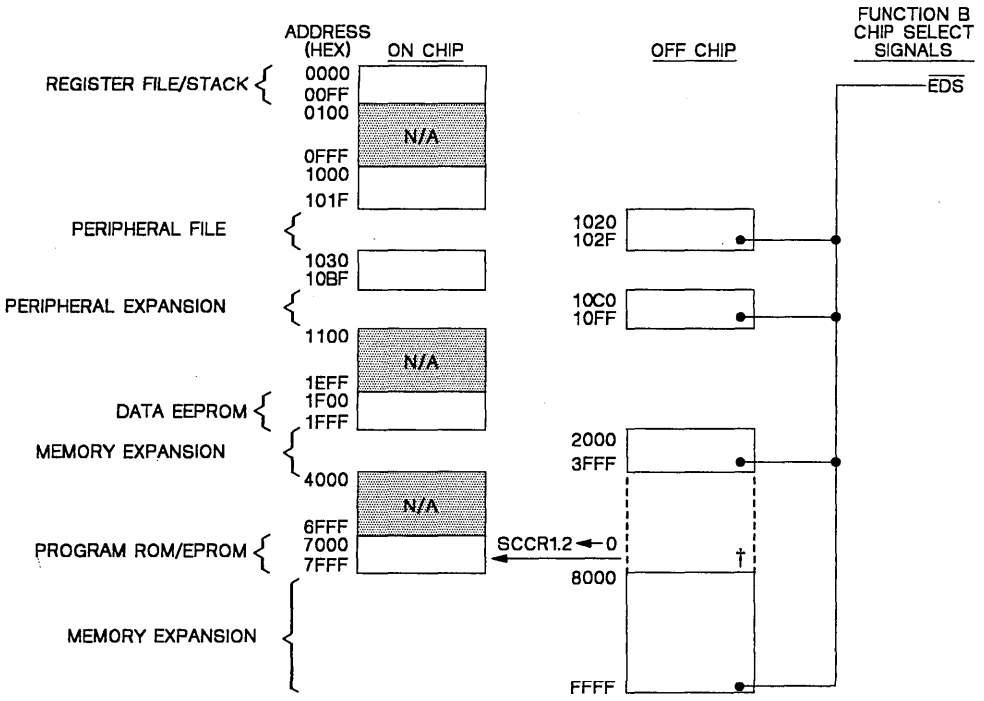
3.4.4 Microprocessor Mode with Internal Program Memory.

In the microprocessor modes, Ports A, B, C, and D become the address, data, and control buses for interface to external memory and peripherals. The on-chip Register File, data EEPROM, and internal Program ROM or EEPROM remain active. Any memory access to addresses 1020h–102F, 10C0h–10FFh, 2000h–3FFFh, and 8000h–FFFFh causes $\overline{\text{EDS}}$ to go low.

When a TMS370Cx50 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.

After RESET, the TMS370Cx50 device enters the microprocessor mode with no internal memory. External memory must contain the reset vector. Software must clear the MEMORY DISABLE bit (SCCR1.2) to enable the internal memory.

Figure 3-10 shows a memory map for the Microprocessor Mode, with Internal Program Memory.



N/A-NOT AVAILABLE
†-AFTER RESET UNTIL SCCR1.2 IS CLEARED BY THE PROGRAM.

Figure 3-10. Microprocessor Mode with Internal Program Memory

To put a TMS370Cx50 device into the Microprocessor Mode with Internal Program Memory:

1) Place a high logic level on the MC pin.
2) Take the $\overline{\text{RESET}}$ pin active low, then return $\overline{\text{RESET}}$ to its inactive high state.
3) 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 (7000h–7FFFh) is now available.

Note:

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

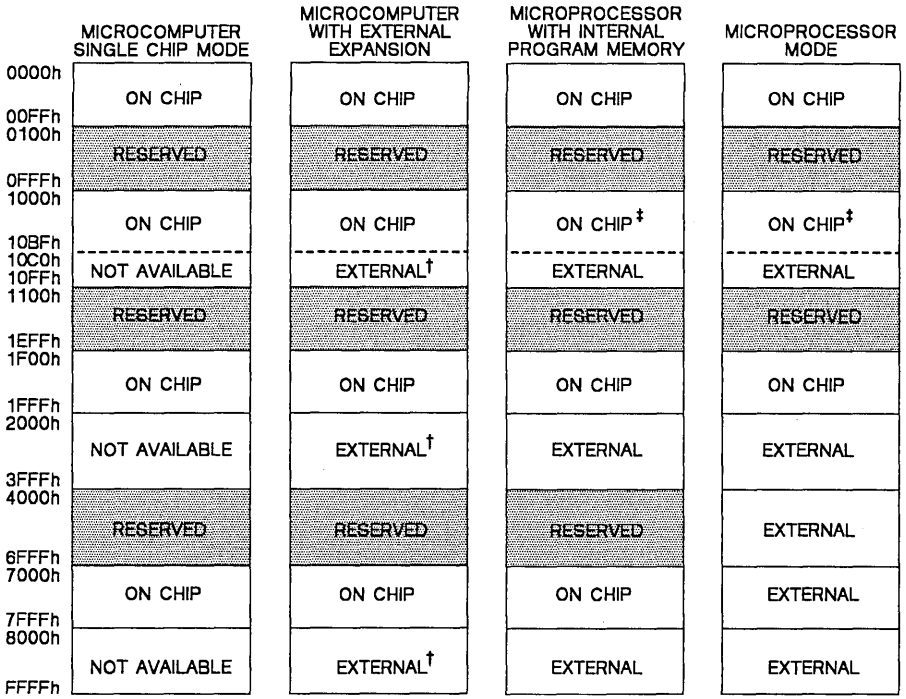
3.4.5 Memory Mode Summary

Table 3-3 summarizes the features of each Memory Mode and the procedure to activate the TMS370 device into each mode. Figure 3-11 gives the memory maps of the four modes.

Table 3-3. Operating Mode Summary

FEATURE	µCOMPUTER SINGLE CHIP	µCOMPUTER w/EXPANDED MEMORY	µPROCESSOR w/INTERNAL MEMORY	µPROCESSOR
Device	TMS370C050/850 TMS370C010/810	TMS370C050/850	TMS370C050/850	TMS370C050/850
Memory Address 7000h-7FFFh	Internal	Internal	Internal	External
Ports A,B,C,D	Digital I/O	Digital I/O Function A† Function B‡	Function B‡	Function B‡
Predecoded CS (Chip Selects)	No	Optional	No	No
Procedure to enter the mode	1. Place logic 0 on the MC pin 2. Take the $\overline{\text{RESET}}$ pin active low, then release $\overline{\text{RESET}}$	1. Place logic 0 on the MC pin 2. Take the $\overline{\text{RESET}}$ pin active low, then release $\overline{\text{RESET}}$ 3. Set Digital I/O registers to Function A†/B‡	1. Place logic 1 on the MC pin 2. Take the $\overline{\text{RESET}}$ pin active low, then release $\overline{\text{RESET}}$ 3. Enable internal memory (Clear SCCR1.2)	1. Place logic 1 on the MC pin 2. Take the $\overline{\text{RESET}}$ pin active low, then release $\overline{\text{RESET}}$

†Function A: Port D = chip select signals $\overline{\text{CSE1}}$, $\overline{\text{CSE2}}$, $\overline{\text{CSH1}}$, $\overline{\text{CSH2}}$, $\overline{\text{CSH3}}$, and $\overline{\text{CSPF}}$ (see Section 4.2).
‡Function B: Port D = expansion memory control signals $\overline{\text{OCF}}$, $\overline{\text{EDS}}$, and $\overline{\text{WAIT}}$ (see Section 4.2).



†PRECODED CHIP SELECT OUTPUTS AVAILABLE ON EXTERNAL EXPANSION BUS.
‡1020h–102Fh EXTERNAL

Figure 3-11. Memory Operating Modes

4. System and Digital I/O Configuration

This section discusses system and I/O configuration. First, the features and options are described; then, the register and bits that control the configuration are described. Lastly, examples of how to set configuration are given.

This section covers the following topics:

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, and Peripheral File register number (PF) are:

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

These registers are shown in Figure 4-1. The "PF" numbers are used by Peripheral File instructions, for example `MOV #00h,P010`.

PERIPHERAL FILE FRAME 1: SYSTEM CONFIGURATION AND CONTROL REGISTERS

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

Figure 4-1. System Configuration and Control Registers

The bits shown in Figure 4-1 in shaded boxes are Privilege Mode bits, that is, they can only be written to 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 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 may configure the system during the initialization sequence following reset. As the last step of a system initialization, set the PRIVILEGE DISABLE (SCCR2.0) to enter the nonprivilege mode and prevent changes to specific control bits within the peripheral file.

Table 4-1 shows the system configuration bits which are write-protected during the nonprivilege mode. These bits should be configured by software prior to exiting the Privilege Mode. The bits shown in the shaded part of Table 4-1 are discussed in later sections which cover the peripheral modules.

Table 4-1. Privilege-Mode Configuration Bits

REGISTER	BIT
SCCR1	MEMORY DISABLE AUTOWAIT DISABLE
SCCR2	PRIVILEGE DISABLE POWERDOWN/IDLE HALT/STANDBY INT NMI OSC FLT DISABLE OSC FLT RST ENA
T1CLT1	WD INPUT SELECT (2-0) WD OVERFL TAP SEL
SPIPRI	SPI PRIORITY
SCIPRI	SCI TX PRIORITY SCI RX PRIORITY
T1PRI	T1 PRIORITY
T2PRI	T2 PRIORITY
ADPRI	AD PRIORITY

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

4.1.2 Oscillator Fault

The processor contains circuitry to monitor the oscillator operation and to indicate whenever the oscillator runs outside the normal operating range. The oscillator fault detection circuit will always trigger below 20 kHz and never above 500 kHz.

Whenever this circuitry detects oscillator operation below the trigger level, the circuit stops the processor. The oscillator monitoring circuit can also pull the RESET pin low causing external devices to reset along with the processor. Three bits control and monitor the operation of the Oscillator Fault circuitry: OSC FLT FLAG, OSC FLT DISABLE, and OSC FLT RST ENA. These bits are described further in Section 4.1.5.

4.1.3 Automatic Wait States

If an application system uses peripherals or expansion memory with slower access time than the TMS370 processor, wait states are required. In some systems this involves complex additional circuitry, but the TMS370 series provides for the automatic addition of wait states which can slow the processor's access time to a compatible period.

In addition, the TMS370 series has a WAIT pin which can hold the processor in a wait state indefinitely. Two bits control the insertion of the Automatic wait state: the PF AUTO WAIT bit and the AUTOWAIT DISABLE bit. The PF AUTO WAIT bit controls the higher four frames (64 bytes) of the peripheral file so that these frames can be migrated off-chip. The AUTOWAIT DISABLE controls all other external memory.

When the AUTOWAIT DISABLE bit equals 1, any access to external memory (excluding the PF file) takes 2 system clock cycles to complete. When

AUTOWAIT DISABLE equals 0, the access takes 3 cycles. The reset value of this bit selects the slower 3-cycle access.

When the PF AUTO WAIT bit equals 1, memory access to the external peripheral files takes 4 system clock cycles. When the PF AUTO WAIT equals 0, the memory is treated like any external memory and the AUTOWAIT DISABLE bit selects the number of cycles per access as either 2 or 3 cycles. Table 13-1 summarizes the effects of the Wait State Control bits.

Table 4-2. Wait State Control Bits

Wait State Control Bits		No. of Clock Cycles per Access	
PF Auto Wait	Autowait Disable	PF File	External Memory
0	0	3	3
0	1	2	2
1	0	4	3
1	1	4	2

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

> **Note:**
>
> When constructing an application circuit with expansion memory, do not forget to connect an unneeded $\overline{\text{WAIT}}$ line to Vcc.

4.1.4 Powerdown and Idle Modes

Each TMS370 device has two low-power modes and an Idle mode. The powerdown modes reduce the operating power by reducing or stopping the activity of various modules whenever processing is not needed. The processor has two types of powerdown modes, the **Halt** mode and the **Standby** mode.

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.

The Halt mode stops the internal clock which stops processing in all the modules providing the lowest power consumption.

Bits 6 and 7 of SCCR2 select the Halt, Standby, or Idle modes. The Idle mode (which is not a low-power mode) is a state which waits for the next interrupt. Executing an IDLE instruction causes the processor to enter one of the two powerdown modes or the simple Idle mode depending on SCCR2.6 and SCCR2.7. The powerdown and Idle mode selection bits are summarized in Table 4-3

Table 4-3. 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

These modes and the methods of exiting the modes are discussed further in Section 4.1.4.1 and Section 4.1.4.2.

In the Standby and Halt mode, the following information is retained:

- The CPU registers:
  - PC
  - Status
  - Stack pointer
- The contents of the RAM
- The Digital output data registers
- The Digital output ports remain active
- Control and status registers of all the modules including the timer contents and the watchdog counter.

If the Serial Peripheral Interface (SPI) or Serial Communications Interface (SCI) is in the process of receiving or transmitting data, that data may be lost. The results of an A-to-D conversion in process will be invalid when a powerdown mode is entered.

The watchdog mode (described in Section 7) should be used with caution in the powerdown modes since the watchdog stops counting in both powerdown modes. If the program executes an IDLE instruction without the interrupts enabled (described in Section 4.1.4.1 and Section 4.1.4.2), then only a reset can start the processor running again.

4.1.4.1 Standby Mode

The Standby mode uses less power than the normal operating mode but more than the Halt mode. The Standby mode stops the clocks to every module except the Timer 1 module. The Timer 1 module can bring the processor out of this low power mode if the interrupts are enabled. To enter this mode set the PWRDWN/IDLE bit (SCCR2.6) and clear the HALT/STANDBY bit (SCCR2.7). The next execution of an IDLE instruction causes the processor to enter the Standby mode.

The processor can exit the Standby mode by one of the following four methods.

- Reset
- External Interrupt 1, 2, or 3 if enabled
- Low level on the RXD pin if, the SCI RX interrupt and receiver are enabled (described in Section 9)
- Timer 1 interrupt if enabled (described in Section 7)

For additional Standby Mode power savings, see Section 4.1.4.3.

4.1.4.2 Halt Mode

The Halt mode stops all internal operations (including Timer 1) and uses the least power of the low power modes. Timer 1 can not bring the processor out of this low-power mode. To select the Halt mode, set the PWRDWN/IDLE bit (SCCR2.6) and the HALT/STANDBY bit (SCCR2.7); then execute an IDLE instruction.

The processor can exit the Halt mode by the following three methods.

- Reset
- External Interrupt 1, 2, or 3 if enabled
- Low level on the RXD pin, if the SCI RX interrupt and receiver are enabled

4.1.4.3 Oscillator Power Bit

The OSC POWER bit (SCCR0.6) allows additional Stand-by 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. For power reduction specifications, see Tables 15-3 and 15-13.

4.1.5 System Control Registers

Each System Control register is summarized in the following charts with definitions

System Control and Configuration Register 0 (SCCR0)
[Memory address - 1010h]

Bit # -	7	6	5	4	3	2	1	0
P010	COLD START	OSC POWER	PF AUTO WAIT	OSC FLT FLAG	MODE PIN WPO	MC PIN DATA	---	µP/µC Mode
	RC	RP-0	RW-0	RW-†	R-†	R-†		R-†

R=Read, W=Write, P=Write only in Privilege mode, C=Clear only,
-n= Value after RESET, †= see bit description

Bit 0 - **µP/µC MODE**. Microprocessor/Microcomputer Mode
This bit indicates the current operating mode (as described in Section 3.4).

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

Bit 1 - Reserved. Read data is indeterminate.

Bit 2 - **MC PIN DATA**. Mode Control Pin Data.
This bit shows the current status of the MC pin.

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

Bit 3 - **MC PIN WPO**. Mode Control Pin Write Protect Override status.
This bit indicates whether or not the voltage on the MC pin is enough for WPO functions. (If this bit is set, then bit 2 is also set.)

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

Bit 4 - **OSC FLT FLAG**. Oscillator Fault Flag.
This flag is reset upon an initial power-up reset. A reset under power does not affect this flag. Therefore, this bit can be be polled to determine the source of a reset.

0 = No oscillator fault found.
1 = Oscillator Fault found. Oscillator period is now or was out of correct operating range. The Oscillator fault detect circuit triggers somewhere within the range of 20 kHz to 500 kHz.

Bit 5 - **PF AUTO WAIT**. Peripheral File Automatic Wait Cycle.

0 = Any access to the peripheral file will take 2 system clock cycles with no System Auto Wait (bit 4 of SCCR1=1), or 3 system clock cycles with the System Auto Wait on (bit 4 of SCCR1=0). (See Section 4.1.3, page 4-3.)
1 = Any access to the upper 4 frames of the peripheral file (address 10C0h to 10FFh) will take 4 system clock cycles to complete. This eases interface requirements for peripheral devices slower than the TMS370 processor. Normal full speed operation consists of 2 system clock cycles per access.

Bit 6 - **OSC POWER**.Oscillator Power.
This bit controls an oscillator power reduction feature. When this feature is in effect, the oscillator drive current is reduced and the oscillator fault detection circuitry is powered down. Current reduction is most useful in the Standby mode. For power reduction specifications, see Tables 15-3 and 15-13.

0 = no oscillator drive current reduction.
1 = oscillator drive current reduction.

Bit 7 - **COLD START.**
This bit does not change during a reset under power.

0 = No power-up reset occurred since last writing a 0 to this bit.
1 = Power-up reset has occurred since last writing a 0 to this bit, indicating one cause of a system reset. The Watchdog Overflow Flag and the Oscillator Fault Flag indicate two other causes of a system reset. A program may take different actions depending upon the source of the reset.

Only writing a 0 to this bit can clear the COLD START flag.

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

Bit # -	7	6	5	4	3	2	1	0
P011	---	---	---	AUTOWAIT DISABLE	---	MEMORY DISABLE	---	---
				RP-0		RP-†		

R=Read, P=write only in privilege state, -n= Value after RESET (†see bit description)

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

Bit 2 - **MEMORY DISABLE.**
This bit enables or disables the internal Program Memory (memory addresses 7000h–7FFFh). This bit does not affect Data EEPROM or internal RAM. RESET initializes this bit to the state of the MC pin. Changes to this bit can occur only in the privilege state.

0 = Enable internal Program Memory and access internal memory at these locations. The $\overline{EDS}$ memory signal will not appear during access to locations 4000h–7FFFh.
1 = Disable internal Program Memory and make all memory accesses to these locations access external memory. An operation on these locations generates an external memory bus cycle with the $\overline{EDS}$ memory signal validating the access. This bit disables the Program EEPROM control register, PEECTL (described in Section 6.2 on page 6-9), if applicable.

Bit 4 - **AUTOWAIT DISABLE**. Automatic Wait State Disable.
This bit, which is cleared at reset, causes an extra cycle to be added to all external bus accesses in order to accommodate slower memory.

0 = Enable the Autowait feature and make external bus access 3 system clock cycles long.
1 = Disable the Autowait feature and make external bus access 2 system clock cycles long.

Changes to this bit can occur only in the privilege state. If the Peripheral File Autowait bit in SCCR0 is set, external peripheral the AUTOWAIT DISABLE bit.

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

Bit # -	7	6	5	4	3	2	1	0
P012	HALT/ STANDBY	PWR- DWN/ IDLE	OSC FLT RST ENA	BUS STEST	CPU STEST	OSC FLT DISABLE	INT1 NMI	PRIV- ILEGE DISABLE
	RP-0	RP-0	RP-0	RP-0	RP-1	RP-0	RP-0	RS-0

R=Read, P=Write only in privilege state, S=Set only, -n= Value after RESET

Bit 0 - **PRIVILEGE DISABLE.** Privilege Mode Disable.
Many bits controlling the system configuration can only be changed 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 the Privilege Disable bit.

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

Bit 1 - **INT1 NMI.** Interrupt 1, Non-Maskable Interrupt.
This bit determines whether Interrupt 1 is maskable or non-maskable (NMI). When Interrupt 1 is non-maskable, it is the second highest priority interrupt (Reset is highest) and unaffected by the interrupt mask and level bits (described in Section 5.1.2) 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.

0 = Interrupt 1 is maskable.
1 = Interrupt 1 is non-maskable (NMI).

Bit 2 - **OSC FLT DISABLE.** Oscillator Fault Disable.
This bit controls circuitry which monitors the oscillator. If this circuitry is enabled and the oscillator falls outside of the correct voltage and frequency range, the processor enters an Oscillator Fault Halt. The oscillator fault circuitry will trigger below 20 kHz and may trigger anywhere between 20 kHz and 500 kHz 500 kHz. The only exit from this halt state is a Reset. If the Oscillator Fault Reset Enable bit (SCCR2.5) is 1, entry to the Fault Halt triggers a system reset. Changes to this bit can occur only in the privilege state.

0 = the oscillator fault circuitry is enabled.
1 = the oscillator fault circuitry is disabled; no attempt is made to halt or reset the processor if the oscillator falls out of range.

Bit 3 - **BUS STEST.**
This bit must be cleared (0) to ensure proper operation.

Bits 4 - **CPU STEST.**
This bit must be cleared (0) to ensure proper operation.

Bit 5 - **OSC FLT RST ENA.** Oscillator Fault Reset Enable.
This bit determines whether or not a system reset is generated when an oscillator fault is detected. Changes to this bit can occur only in the privilege mode.

0 = A Reset will not be generated if the oscillator falls below the correct operating range. (The monitor circuit also depends upon the Oscillator Fault Disable bit.)
1 = A Reset is generated whenever the oscillator frequency falls below the correct operating range if SCCR2.2 is cleared.

Bit 6 - **PWRDWN/IDLE.** Powerdown/Idle.
This bit determines the mode entered by the CPU when an Idle instruction is executed. Changes to this bit can occur only in the privilege mode.

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

Bit 7 - **HALT/STANDBY.**
The following descriptions apply only if the Powerdown/Idle bit is set, otherwise the Halt/Standby bit has no effect. See Section 4.1.4 for a description of the Halt and Standby modes. Changes to this bit can occur only in the privilege mode.

0 = When an IDLE instruction is executed, the processor will enter the Standby mode which stops program execution and disables the system clock to all "nonessential" peripherals. The system clock to the Timer 1 continues to run and the timer can generate an interrupt to bring the processor out of the Standby mode.
1 = When an IDLE instruction is executed, the processor will enter the Halt mode which stops the internal oscillator and suspends the system and peripheral operations. This mode provides the lowest power consumption.

4.2 Digital I/O Configuration

On TMS370 devices, the power, reset, MC, and crystal pins are dedicated to one function. Every other pin may 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.

On TMS370Cx50 devices, 32 of a possible 55 I/O pins are dedicated to Ports A, B, C and D; each port has 8 pins each.

On TMS370Cx10 devices, 13 of a possible 22 I/O pins are dedicated to Ports A and D. Port A contains 8 pins and Port D contains 5 pins.

Frame 2 of the peripheral file (memory addresses 1020h–102Fh) contain the control registers for reading, writing and configuring Ports A, B, C, and D. These registers are shown in Figure 4-2.

PERIPHERAL FILE FRAME 2: DIGITAL PORT CONTROL REGISTERS

ADDR	PF	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	
1020h	20	RESERVED								APORT1
1021h	21	PORT A CONTROL REGISTER 2								APORT2
1022h	22	PORT A DATA								ADATA
1023h	23	PORT A DIRECTION								ADIR
1024h	24	RESERVED								BPORT1
1025h	25	PORT B CONTROL REGISTER 2								BPORT2
1026h	26	PORT B DATA								BDATA
1027h	27	PORT B DIRECTION								BDIR
1028h	28	RESERVED								CPORT1
1029h	29	PORT C CONTROL REGISTER 2								CPORT2
102Ah	2A	PORT C DATA								CDATA
102Bh	2B	PORT C DIRECTION								CDIR
102Ch	2C	PORT D CONTROL REGISTER 1								DPORT1
102Dh	2D	PORT D CONTROL REGISTER 2								DPORT2
102Eh	2E	PORT D DATA								DDATA
102Fh	2F	PORT D DIRECTION								DDIR

Figure 4-2. Digital Port Control Registers

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

- Port X Control Register 1 (XPORT1)
- Port X Control Register 2 (XPORT2)
- Port X Data (XDATA)
- Port X Direction (XDIR)

The same bit position of each of these register affects the corresponding bit in the port. For example, Bit 0 of registers DPORT1, DPORT2, DDATA and DDIR control Port D, bit 0. This is illustrated in Figure 4-3.

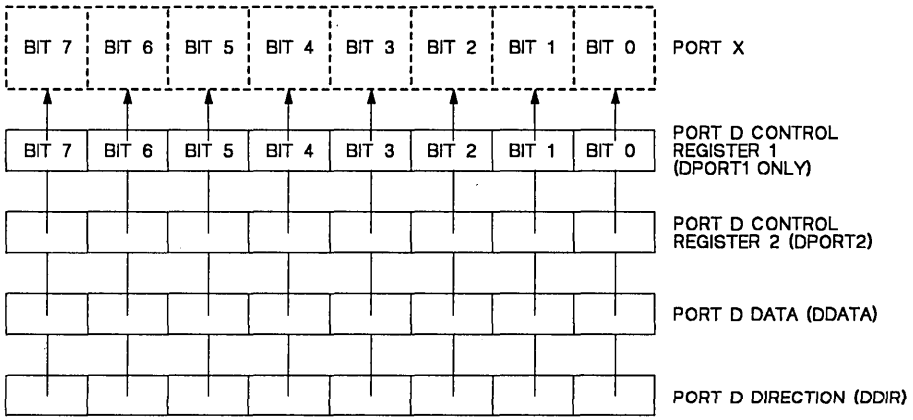


Figure 4-3. Port Control Register Operation

Bits from the XPORT1 and XPORT2 registers determine the function of the corresponding port pin, either a 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 a 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 a I/O pin.

Figure 4-4 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 figure.

PORT	PIN	INPUT	OUTPUT	FUNCTION A	FUNCTION B (μP MODE)
		XPORT1 = 0† XPORT2 = 0 XDATA = y XDIR = 0	XPORT1 = 0† XPORT2 = 0 XDATA = q XDIR = 1	XPORT1 = 0† XPORT2 = 1 XDATA = x XDIR = x	XPORT1 = 1† XPORT2 = 1 XDATA = x XDIR = x
A	0–7	DATA IN y	DATA OUT q	DATA BUS	RESERVED
B	0–7	DATA IN y	DATA OUT q	LOW ADDR	RESERVED
C	0–7	DATA IN y	DATA OUT q	HI ADDR	RESERVED
D	0	DATA IN y	DATA OUT q	$\overline{CSE2}$	$\overline{OCF}$
D	1	DATA IN y	DATA OUT q	$\overline{CSH3}$	
D	2	DATA IN y	DATA OUT q	$\overline{CSH2}$	
D	3	DATA IN y	DATA OUT q	CLKOUT	CLKOUT
D	4	DATA IN y	DATA OUT q	R/$\overline{W}$	R/$\overline{W}$
D	5	DATA IN y	DATA OUT q	$\overline{CSPF}$	
D	6	DATA IN y	DATA OUT q	$\overline{CSH1}$	$\overline{EDS}$
D	7	DATA IN y	DATA OUT q	$\overline{CSE1}$	$\overline{WAIT}$

XPORT1 = 1
XPORT2 = 0
XDATA = x
XDIR = x
} NOT DEFINED

†DPORT ONLY

Figure 4-4. Port Configuration Registers Set-Up

LOW ADDR/HI ADDR - External memory address bus. Output only.

DATA BUS - External data bus. Input and output.

$\overline{EDS}$ - External Data strobe: This signal goes low during external memory operations. The rising edge of $\overline{EDS}$ validates the read input data and the write data is available after the falling edge of $\overline{EDS}$.

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

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

$\overline{CSH3}$ - Chip Select Half 3: This signal has the same timing as $\overline{EDS}$ but it only goes active during access to the upper half of memory (locations 8000h–FFFFh). Used to select a third bank of memory. Setting this pin to a high-level general-purpose output disables the bank.

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

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

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

CLKOUT - Clock Output: Outputs one quarter of the crystal or external oscillator frequency. Used to synchronize external peripherals.

R/$\overline{W}$ - Read or Write operation: Goes high at the beginning of read operations and low during write operations.

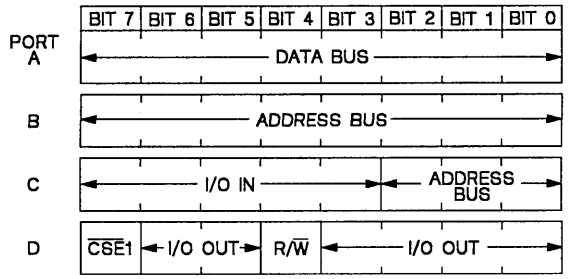
$\overline{WAIT}$ - WAIT input: An external, low signal applied to this pin, when sampled, causes the processor to hold the information on the expansion bus for 1 or more extra clock out cycles. This pin is sampled during the rising edge of CLKOUT after $\overline{EDS}$ goes active.

$\overline{OCF}$ - Opcode Fetch: Goes low at the beginning of a memory read operation that fetches the first byte of an instruction. It then resumes its high level at the end to the Opcode fetch cycle.

The Pre-decoded 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 pre-decoded chip selects and the non-multiplexed bus. Another advantage of the chip selects is the ability to do easy memory bank selection. Without bank selection, the $\overline{CSH1}$, $\overline{CSE1}$, and $\overline{CSPF}$ signals can easily access about 40 kilobytes of memory in the three different areas. With bank selection, the processor can access 112 kilobytes of memory.

Example 4-1. Digital Ports Set-up Example

To illustrate configuring the Digital Ports, assume that a TMS370C050 is to operate in the expanded microcomputer mode, and that 2 kilobytes of memory is needed at 2000h to 27FFh. The top half of the figure below shows the port configuration wanted. Port A is set as the external data bus. Port B is the low-order byte of the address bus. Bits 0 through 2 of Port C are the high-order address bits of the eleven bits necessary to access 2 kilobytes 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 used for external memory control signal R/$\overline{W}$. The remaining bits of Port D are used as I/O output.



	PORT A CONTROL REGISTERS								
1021h	1	1	1	1	1	1	1	1	MOV #0FFh, P021
1022h	X	X	X	X	X	X	X	X	
1023h	X	X	X	X	X	X	X	X	
	PORT B CONTROL REGISTERS								
1025h	1	1	1	1	1	1	1	1	MOV #0FFh, P025
1026h	X	X	X	X	X	X	X	X	
1027h	X	X	X	X	X	X	X	X	
	PORT C CONTROL REGISTERS								
1029h	0	0	0	0	0	1	1	1	MOV #007h, P029
102Ah	X	X	X	X	X	X	X	X	
102Bh	0	0	0	0	0	X	X	X	MOV #0, P02B
	PORT D CONTROL REGISTERS								
102Ch	0	0	0	0	0	0	0	0	MOV #0, P02C
102Dh	1	0	0	1	0	0	0	0	MOV #090h, P02D
102Eh	X	q	q	X	q	q	q	q	
102Fh	X	1	1	X	1	1	1	1	MOV #0FFh, P02E

The bottom half of the above figure shows the port control registers set up to establish the configuration shown in the top half of the figure. To determine the bits needed to set the registers, use Figure 4-4. For example, to set Port A as the data bus, find Port A in the left hand column of Figure 4-4. Look across the row to find "Data bus", then follow the column up to find

1
x
x

in the column heading. These are the bits needed to set *each* Port A bit as a data bus.

The assembly language instructions on the right of the preceding figure show one method of setting up the registers to the left. The "Pxxx" operand indicates peripheral file access (See Section 12 for more information on peripheral file instructions).

4.2.1 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 changing a pin from an general-purpose input pin to an output pin, write to the Data register first to set up the data and then set the data direction register. This prevents unknown data on the pin from interfering with the external circuitry.

4.2.2 Microprocessor Mode

Initializing the TMS370Cx50 to the microprocessor mode forces Ports A,B,C and D to Function B as shown in Figure 4-4. 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. The TMS370Cx10 is not defined for operation in the memory expansion modes so the device must be powered up in the Microcomputer mode.

When operating in the microprocessor mode, any access to the Port peripheral frame, 1020-102F, is decoded as external address. Memory accesses to this frame can control external hardware which emulates the digital I/O functions. Write operations to this frame still update the internal registers which is useful when operating in microcomputer mode with internal memory disabled.

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

Figure 4-5 shows an example of the TMS370C850 interfaced to 112 kilobytes of external memory. The Function-A chip-select signals are used to enable one of three banks of EPROM, an external peripheral device, and one of two banks of static RAM. In this example, all eight bits of port A are used as the data bus, all eight bits of port B are used as the address LSB, and seven port C bits are used to complete the 15 bit address bus.

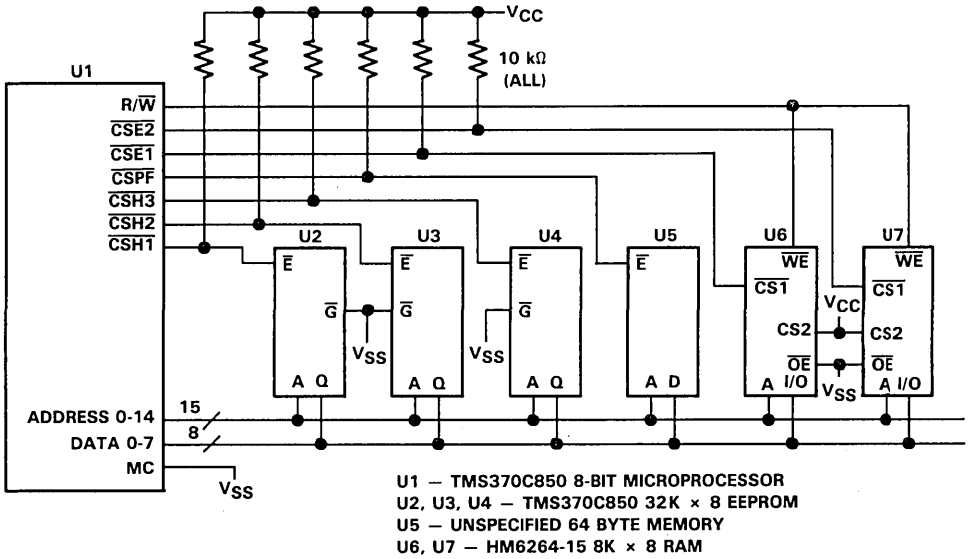


Figure 4-5. System Interface Example

5. Interrupts and System Reset

This section covers the following topics:

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 a 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.

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 Section 3.2.2 on page 3-4).

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

The processor services the pending interrupts upon completion of current instruction execution, depending on their interrupt mask and priority conditions. The processor services all enabled Level 1 interrupts before servicing any Level 2 interrupts. Within each level, the processor services the highest priority interrupts first. The hardware priorities are shown in Table 5-1.

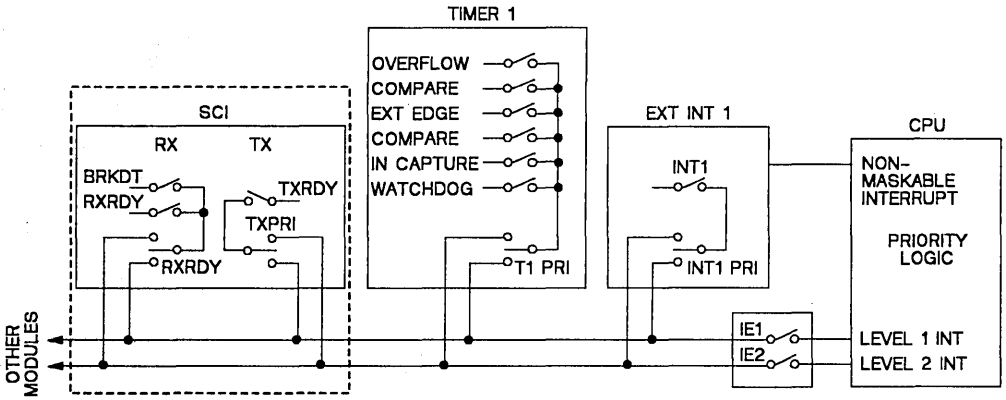


Figure 5-1. Interrupt Control

TMS370Cx50 devices have ten hardware system interrupts as shown in Table 5-1. TMS370Cx10 devices have the first six interrupts shown non-shaded in this table. Each system interrupt has a dedicated interrupt vector located near the end of program memory (locations 7FECh–7FFFh) which contains the address of the interrupt service routine. A system interrupt may have multiple interrupt sources (for example, SCI RX has two interrupt sources).

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

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

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

1) Increment the Stack Pointer (SP) and store the contents of the Status register (ST) at the location pointed to by the SP.
2) Set the ST to 00h (disables further interrupt recognition).
3) Obtain the identity of the interrupting peripheral.
4) Rewind the Program Counter to point to the aborted opcode.
5) Increment SP and store the original PC high byte at the location pointed to by the SP.

6) Get address (high byte) of interrupt service routine and store it in the PC high byte (PCH).
7) Increment SP and store the original PC (low byte) at the location pointed to by SP.
8) Get the interrupt-service-routine address (low byte) and store it in the PC low byte (PCL).
9) Resume instruction execution with the new PC contents.

It takes a minimum of 15 cycles from the time that an interrupt is triggered, to the reading of the first instruction of the interrupt service routine. The time depends on the instruction in progress when the interrupt is asserted and at what point during an instruction the interrupt is asserted. The worst case occurs 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 2 extra clock cycles to synchronize before the processor can detect it.

Table 5-1. Hardware System Interrupts

Interrupt Source	Interrupt Flag	System Interrupt	Vector Address	Priority§
External RESET Watchdog Overflow Oscillator Fault Detect	COLD START WD OVRFL INT FLAG OSC FLT FLAG	RESET†	7FFEh, 7FFFh	1
External INT1	INT1 FLAG	INT1†	7FFCh, 7FFDh	2
External INT2	INT2 FLAG	INT2†	7FFAh, 7FFBh	3
External INT3	INT3 FLAG	INT3†	7FF8h, 7FF9h	4
SPI RX/TX Complete	SPI INT FLAG	SPIINT	7FF6h, 7FF7h	5
Timer 1 Overflow Timer 1 Compare 1 Timer 1 Compare 2 Timer 1 External Edge Timer 1 Input Capture Watchdog Overflow	T1 OVRFL INT FLAG T1C1 INT FLAG T1C2 INT FLAG T1EDGE INT FLAG T1 IC INT FLAG WD OVRFL INT FLAG	T1INT‡	7FF4h, 7FF5h	6
SCI RX Data Register Full SCI RX Break Detect	RXRDY FLAG BRKDT FLAG	RXINT†	7FF2h, 7FF3h	7
SCI TX Data Register Empty	TXRDY FLAG	TXINT	7FF0h, 7FF1h	8
Timer 2 Overflow Timer 2 Compare 1 Timer 2 Compare 2 Timer 2 External Edge Timer 2 Input Capture 1 Timer 2 Input Capture 2	T2 OVRFL INT FLAG T2C1 INT FLAG T2C2 INT FLAG T2EDGE INT FLAG T2IC1 INT FLAG T2IC2 INT FLAG	T2INT	7FEEH, 7FEFh	9
A-D Conversion Complete	AD INT FLAG	ADINT	7FECh, 7FEDh	10

†Releases microcontroller from STANDBY and HALT low-power modes.
‡Releases microcontroller from STANDBY low-power mode.
§Relative priority within an interrupt level.

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.

PERIPHERAL FILE FRAME 1: EXTERNAL INTERRUPT CONTROL REGISTERS

ADDRESS	PF	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	
1017h	P017	INT1 FLAG	INT1 PIN DATA	---	---	---	INT1 POLARITY	INT1 PRIORITY	INT1 ENABLE	INT1
1018h	P018	INT2 FLAG	INT2 PIN DATA	---	INT2 DATA DIR	INT2 DATA OUT	INT2 POLARITY	INT2 PRIORITY	INT2 ENABLE	INT2
1019h	P019	INT3 FLAG	INT3 PIN DATA	---	INT3 DATA DIR	INT3 DATA OUT	INT3 POLARITY	INT3 PRIORITY	INT3 ENABLE	INT3

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

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

INT1 can be programmed to be a maskable or non-maskable interrupt. When INT1 is non-maskable, it cannot be masked by the individual or global mask bits. Remember that the INT1 NMI bit (SCCR2.1) is protected during non-privileged operation and should be configured during the initialization sequence following reset (see INT1 NMI bit description on page 4-9).

The application program must configure the following bits for each interrupt to function correctly. 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 which poll the interrupt flag instead of generating a system interrupt.

The INT PIN DATA bit shows the level presently on the interrupt pin. This also allows the use of this bit as a simple input pin if the interrupt function is not needed.

On interrupts 2 and 3, the INT DATA DIR determines if the pin functions as a general purpose output or as an input/interrupt pin. If you select the general purpose output function, then the value written by software to the INT DATA OUT bit determines the level 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.

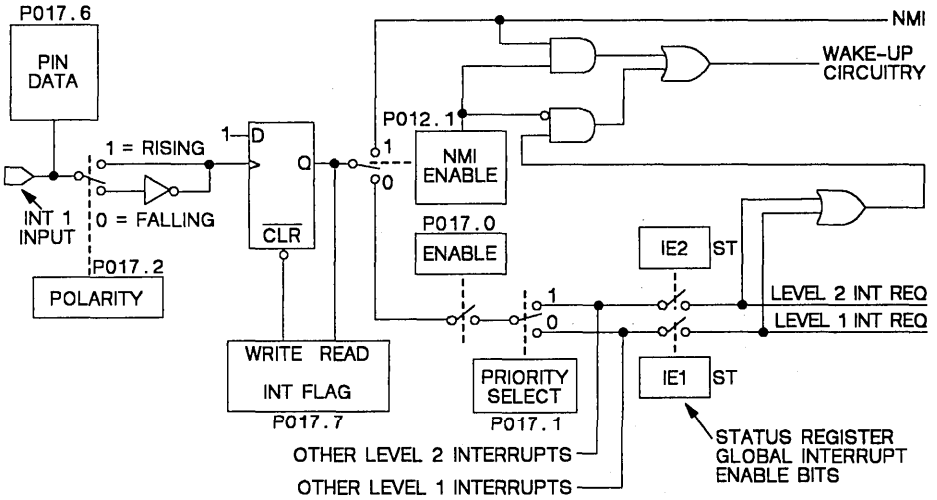


Figure 5-3. Interrupt 1 Block Diagram

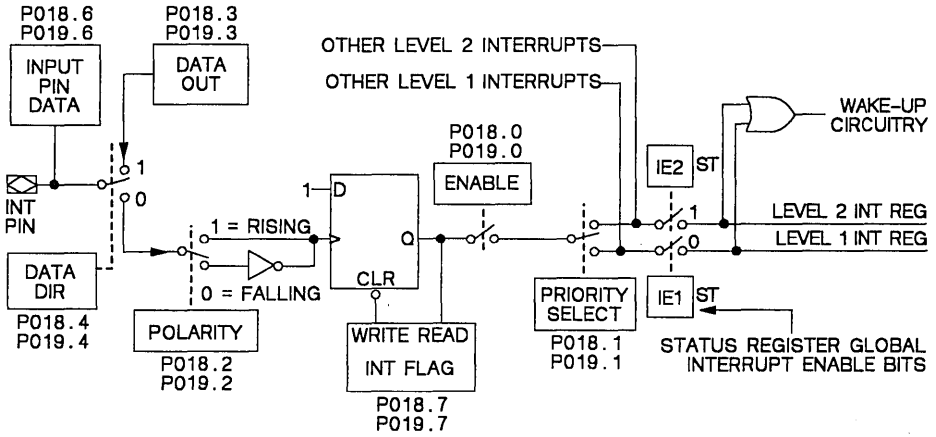


Figure 5-4. Interrupts 2 and 3 Block Diagram

5.1.3 Interrupt Control Registers

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 after RESET

Bit 0 - **INT1 ENABLE**. Interrupt 1 Enable.
This bit enables the interrupts for the INT1 pin.

1 = Enable INT1 interrupts.
0 = Disables INT1 interrupts.

Bit 1 - **INT1 PRIORITY**. Interrupt 1 priority.
This bit determines 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 - **INT1 POLARITY**. Interrupt 1 Polarity.
This bit determines whether INT1 triggers on a rising edge or 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.
This bit indicates that the selected transition on INT1 has occurred. An interrupt can occur as long as this bit remains set, thus the application program must clear this bit during the interrupt handling routine. This bit is not affected by the Interrupt 1 Enable bit.

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

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

R=Read, W=Write, C=Clear only, -n= Value after RESET

Bit 0 - **INT2 ENABLE**. Interrupt 2 Enable.
This bit enables the interrupts for the INT2 pin.

1 = Enable INT2 interrupts.
0 = Disables INT2 interrupts.

Bit 1 - **INT2 PRIORITY**. Interrupt 2 Priority.
This bit determines the interrupt level of the INT2 pin; either a high, level-1 interrupt or a low, level-2 interrupt.

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

Bit 2 - **INT2 POLARITY**. Interrupt 2 Polarity.
This bit determines whether INT2 triggers on a rising edge or a falling edge.

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

Bit 3 - **INT2 DATA OUT**. Interrupt 2 Data Out.
If software configures the INT2 pin as an output pin (INT2 DATA DIR=1), then the value that the software writes to the INT2 DATA OUT bit determines the value of that output pin.

Bit 4 - **INT2 DATA DIR**. Interrupt 2 Data Direction.
The INT2 pin can be configured as either an output pin or as an input/interrupt pin.

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

Bit 5 - Reserved. Read data is indeterminate.

Bit 6 - **INT2 PIN DATA**. Interrupt 2 Pin Data.
This bit displays the current value of the INT2 pin.

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

Bit 7 - **INT2 FLAG**. Interrupt 2 Flag.
This bit indicates that the selected transition on INT2 has occurred. An interrupt can occur as long as this bit remains set, thus the program must clear this bit during the interrupt handling routine. This bit is not affected by the INT2 ENABLE bit.

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

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

R=Read, W=Write, C=Clear only, -n= Value after RESET

Bit 0 - **INT3 ENABLE**. Interrupt 3 Enable.
This bit enables the interrupts for the INT3 pin.

1 = Enable 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 a falling edge.

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

Bit 3 - **INT3 DATA OUT**. Interrupt 3 Data Out.
If software configures the INT3 pin as an output pin (INT3 DATA DIR=1), then the value that the software writes to the INT3 DATA OUT bit determines the value of that output pin.

Bit 4 - **INT3 DATA DIR**. Interrupt 3 Data Direction.
The INT3 pin can be configured as either an output pin or as an input/interrupt pin.

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

Bit 5 - Reserved. Read data is indeterminate.

Bit 6 - **INT3 PIN DATA**. Interrupt 3 Pin Data.
This bit displays the current condition of the INT3 pin.

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

Bit 7 - **INT3 FLAG**. Interrupt 3 Flag.
This bit indicates that the selected transition on INT3 has occurred. An interrupt can occur as long as this bit remains set, thus the program must clear this bit during the interrupt handling routine. This bit is not affected by the Interrupt 3 Enable bit.

5.1.4 Multiple Interrupt Servicing

When servicing an interrupt, the processor automatically clears the global interrupt enable bits IE1 and IE2. 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. When a nested interrupt service routine completes, it returns to the previous interrupt service routine when the RTI instruction executes. Too many nested interrupts could overflow the stack causing program failure.

5.2 Resets

The TMS370 has three possible reset sources: a low input to the $\overline{\text{RESET}}$ pin, a programmable watchdog timer timeout (described in Section 7.3, 7-17), or a programmable oscillator fault failure (described in Section 4.1.2, 4-3). After the occurrence of a reset, the program can interrogate the status bits (shown in Table 5-2) to determine the source of the reset in order to take appropriate action. If none of the sources, indicated in Table 5-2, caused the interrupt then the $\overline{\text{RESET}}$ pin was pulled low by external hardware.

Table 5-2. Reset Sources

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

The $\overline{\text{RESET}}$ pin starts the hardware initialization and ensures an orderly software startup. The $\overline{\text{RESET}}$ pin is an input/output pin. A low level pulse initiates the reset sequence. The microcontroller is held in reset until the $\overline{\text{RESET}}$ pin goes inactive (high). If the reset input signal remains low for less than eight system clock cycles, the processor holds the external $\overline{\text{RESET}}$ pin low for eight system clock cycles to reset external system components.

> **Note:**
>
> TMS370 family members with on-chip EEPROM require external RESET control during power transitions. The external RESET pin must be active (low) while V_{CC} is below its minimum specified operating level, thereby ensuring the integrity of EEPROM contents. An active RESET prevents the EEPROM contents from being corrupted by improper instruction execution due to insufficient V_{CC} supply voltage and ensures that the EEPROM write control registers (DEECTL, PEECTL) power up in the correct state when V_{CC} returns to its specified operating range.

An application must activate the $\overline{\text{RESET}}$ pin at powerup, with an external input or a RC power-up reset circuit. Recall that the basic operating mode, microcomputer or microprocessor, is determined by the voltage level applied to the MC pin when the $\overline{\text{RESET}}$ pin goes inactive (high). The $\overline{\text{RESET}}$ pin can be pulled low at any time during operation to start the reset sequence immediately.

The sequence of events during reset is as follows:

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

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 set to 0 with the exception of the bits shown Table 5-3.

Table 5-3. Control-Bit States Following Reset

Register	Control Bit	Powerup	Warm Reset	
		Micro-computer	Micro-computer	Micro-processor
SCCRO	µP/µC Mode	0	0	1
	MC PIN DATA	0	0	1
	COLD START	1	†	†
	OSC FLT FLAG	0	†	†
PORT1‡	all 8 bits	0	0	1
PORT2‡	all 8 bits	0	0	1
T1CTL2	WD OVRFL FLAG	0	†	†
TXCTL	TX EMPTY	1	1	1
	TXRDY	1	1	1
ADSTAT	AD READY	1	1	1

†Status bit corresponding to the active reset source is set, else no effect.
‡Refers to Port Control Registers A, B, C, and D.

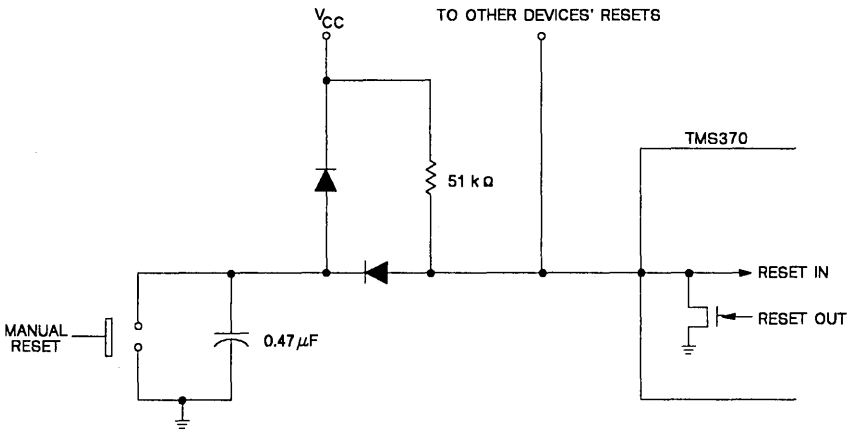


Figure 5-5. Typical Reset Circuit

6. EEPROM Modules

This section discusses the architecture and programming of the Data EEPROM modules on all TMS370 devices and the Program EEPROM module on TMS370C8x0 devices. Additional information about the EEPROM modules is included in Section 13, Design Aids and Section 15, Electrical Specifications.

This section covers the following topics:

6.1 Data EEPROM Module

The TMS370 Data EEPROM module is a 256-byte array configured into eight 32-byte blocks at addresses, 1F00h–1FFFh. This module also contains a voltage generator which 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} voltage source. Each EEPROM module contains a voltage generator so a routine can simultaneously program a byte of Data EEPROM and a byte of Program EEPROM without conflict.

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 either a byte or single-bit basis. The memory can also be protected from inadvertent writing with a write-protect feature.

The Data EEPROM is controlled by the DEECTL register and the Write Protect Register (WPR). The Data EEPROM control register (DEECTL) contains the bits needed to initiate and monitor EEPROM programming. The Write Protection Register (WPR) contains the write protection bits for each 32-byte block of Data EEPROM.

6.1.1 Control Registers

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

6.1.1.1 Write Protection Register (WPR)

The WPR register provides write protection for Data EEPROM contents. The WPR is located in BLK0 of the Data EEPROM at address 1F00h; therefore, the WPR itself is write protected whenever BLK0 is protected.

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

Once block 0 is protected, the write-protection configuration can not be altered unless write protection is overridden by placing the microcomputer into the Write Protection Override mode (12 volts on the MC pin). There is no write protection during a Write Protection Override, and the WPR is considered a normal data location within the Data EEPROM array during this time.

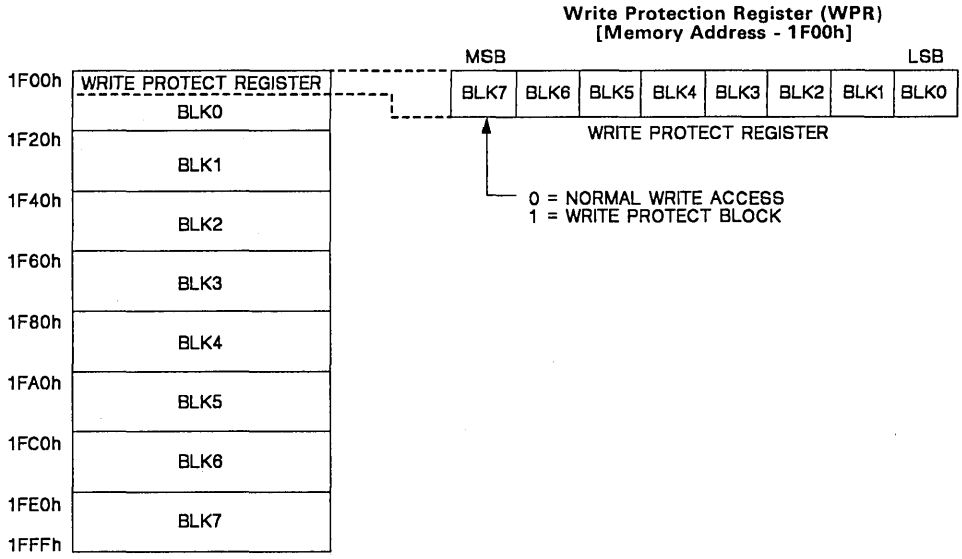


Figure 6-1. Write Protection Bits

6.1.1.2 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. The following figure illustrates the bit definitions for the DEECTL 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 after RESET (†-see Bit 7 description)

Bit 0- **EXE**. Execute.
Set this bit to initiate the write operation defined by the remaining control register bits. Clear this bit to terminate 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- **W1WO**. Write1/Write0.
This bit determines whether the Ones or Zeroes programming mode is to be used (see Section 6.1.2, page 6-4). This bit is write protected whenever the EXE bit is set.
0 = Write zeros.
1 = Write ones.

Bit 2- **AP**. Array Program.
Set this bit to program the entire array with the value specified by the W1W0 bit in a single programming cycle (refer to Section 15 for timing). Blocks protected in the WPR register are *not* programmed. This bit is write protected whenever the EXE bit is set.

If BLK0 is unprotected and W1W0 is zero, this function clears the WPR; any array locations previously protected will lose their protection, but their contents are not altered during the current programming cycle.
0 = Array programming disabled.
1 = Array programming enabled.

Bit 3-6 Reserved. Read data is indeterminate.

Bit 7 - **BUSY**.
This bit is set during Data EEPROM programming to indicate that an operation is in progress. Reading any location of the EEPROM during programming returns the data being programmed. In order to let the EEPROM voltages stabilize, the BUSY bit is set for 128 cycles:
1) after a reset,
2) after an exit from a power-down state, and
3) after programming the EEPROM.

If an attempt is made to access the EEPROM during this 128 cycle period, the Data EEPROM holds execution of the processor by asserting the WAIT signal until the 128 cycles is complete.
0 = EEPROM array is ready for access.
1 = EEPROM array is not ready for access.

6.1.2 Programming the Data EEPROM

The procedure for programming the Data EEPROM is controlled by the DEECTL (P01A) and the WPR (1F00h) registers. 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 set 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 (1F00h) register 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, 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).

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

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

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

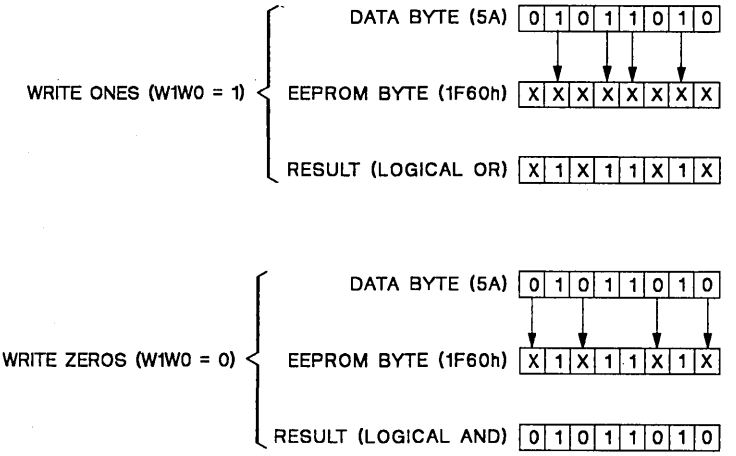


Figure 6-2. EEPROM Programming Example

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

Example 6-1. Data EEPROM Programming Example

The following subroutine loads the data byte 5A into the Data EEPROM location 1F60h. Figure 6-2 illustrates the result of this subroutine.

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

Load the value 5A into the Data EEPROM address 1F60h (a). Begin a Write Ones programming sequence by (b) setting the W1W0 and EXE bits in the DEECTL register to a 1. The programming delay parameter t_W(PGM)B (10 ms for this example – see Section 15 for required timing) is taken care of with a delay loop (d,e). The number of loops required is #2278 (c), and can be derived in the following manner:

1) Delay loop (d,e) requires 18 cycles to complete if a jump is taken.
2) An operating frequency of 20 MHz results in a system cycle time of 200 ns.
3) The number of loops required is calculated as follows:

   loop count = t_W(PGM)B / (system cycle time X delay loop cycle count)

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

Note: alternatively, a timer may be used for this delay.

After the delay, clear the EXE bit (f), and continue the Write Zeros routine (g through k). The value "5A" has now been programmed into location 1F60h of the Data EEPROM.

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

- after a reset,
- after exit from a power-down state, and
- after programming the EEPROM.

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

6.1.3 Write Protection Register Operation

The Write Protection Register (WPR) allows the application program to guard any or all of the eight blocks of EEPROM shown in Figure 6-1, page 6-3. Each block has a representative bit in the WPR. The blocks to be protected have their corresponding bit set in the WPR. Block zero contains the WPR. Therefore, once the bit for block zero (BLK0) is set, WPR can no longer be changed unless the Write Protect Override mode is entered.

The following example illustrates programming the WPR. In this example, the program protects blocks 0 and 2. See Section 13 for more examples of programming the EEPROM module.

```
        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
        MOVW   #2778,R011     ;10 ms delay loop
DELAY   INCW   #-1,R011
        JC     DELAY
        MOV    #0,P01A        ;Clear W1W0 and EXE bits
```

6.2 Program EEPROM Module

The following is a description of the Program EEPROM module used in the TMS370 family. This module serves in place of the 4-kilobyte program ROM within the TMS370C850 and TMS370C810 for systems in prototype or small production runs.

The module consists of a 4-kilobyte array of EEPROM at address locations 7000h through 7FFFh. The CPU can fetch data and execute instructions from this memory space. Programming control logic for the Program EEPROM is located at address 101Ch (P01C).

The CPU accesses the array with normal memory read cycles. Write cycles to the Program EEPROM require a special sequence of events. This sequence is similar as that for the Data EEPROM, except that the PEECTL register (P01C) is used for control and there is no Write Protection Register for the Program EEPROM.

The Program EEPROM module can only be written to when the TMS370 device is operating in the Write Protect Override mode (12 volts on the MC pin). A read access to the Program EEPROM during a WPO write operation, returns the data being programmed.

6.2.1 Program EEPROM Control Register (PEECTL)

The PEECTL register, at address 101Ch in the peripheral file, controls programming of the Program EEPROM. The following figure illustrates the bit definitions for the PEECTL.

Program EEPROM Control Register (PEECTL)
[Memory Address - 101Ch]

Bit # -	7	6	5	4	3	2	1	0
P01C	BUSY	---	---	---	---	AP	W1W0	EXE
	R-†					RW-0	RW-0	RW-0

R=Read, W=Write, -n= Value after RESET (†-see Bit 7 description)

Bit 0- **EXE**. Execute.
Set this bit to initiate the write operation defined by the other control register bits. Clear this bit to terminate the operation.

0 = Inactive.
1 = Active.

Bit 1- **W1WO**. Write1/Write0.
This bit determines whether the Ones or Zeroes programming mode is to be used. This bit is write protected when EXE=1.

0 = Write zeros.
1 = Write ones.

Bit 2- **AP**. Array Program.
Set this bit to program the entire array with the value specified by the W1W0 bit. With this function, large sections of EEPROM can be altered in a fraction of the time necessary to program byte by byte. This bit is write protected when EXE=1.

0 = Array programming disabled.
1 = Array programming enabled.

Bit 3-6 Reserved. Read data is indeterminate.

Bit 7 - **BUSY**.
This bit is set during EEPROM programming to indicate that an operation is in progress. Reading any location of the EEPROM during programming returns the data being programmed. In order to let the EEPROM voltages stabilize, the BUSY bit is set for 128 cycles:

1) after a reset,
2) after an exit from a power-down state, or
3) after programming the EEPROM.

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

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

6.2.2 Programming the Program EEPROM

The procedure to program this EEPROM module is similar to the procedure described in Section 6.1.2 with the following differences.

- The PEECTL register (address P01C in the peripheral file) controls the Program EEPROM.
- There is no write-protection register. The Program EEPROM is write protected at all times unless the TMS370 device is in the Write Protection Override mode.

The programming sequence is:

1) External hardware puts 12 volts on the MC pin to enter the WPO mode.
2) Write a data byte to the desired address in the Program EEPROM.
3) Write the command byte (03h for programming ones or 01h for programming zeros) to the PEECTL register at address 101Ch.
4) Wait for the programming delay time, $t_{W(PGM)B}$ (see Section 15, Device Specifications).
5) Write 00h to the PEECTL register to clear the EXE bit.
6) Repeat steps 3 through 6 as necessary to complete the programming.
7) External hardware returns the MC pin to its normal value (logic 1 for microprocessor mode or logic 0 for the microcomputer mode).

When the AP bit is set, the entire Program EEPROM is programmed to all ones or all zeros in a single write sequence. When W1W0=0, all zeros are programmed to the entire Program EEPROM. When W1W0=1, all ones are programmed to the entire Program EEPROM. This function is useful for block-erase operations. Issue this command by writing 05h to the PEECTL register to set the AP and EXE bits.

When the TMS370 is in the HALT or STANDBY low-power mode, all operations of the Program EEPROM are stopped. All programming operations should be completed before entering a HALT or STANDBY state. Any EEPROM programming operation, in progress at the time that the halt is entered, is aborted. The data at the address being programmed is indeterminate.

6.2.3 Write Protection of Program EEPROM

The Program EEPROM memory is always write protected unless the device is put into the Write Protect Override (WPO) mode by applying 12 volts to the MC pin while the $\overline{\text{RESET}}$ pin is a logic 1. When the device is in the WPO mode, all Program and Data EEPROM memory can be overwritten.

7. Timer 1 Module

This section discusses the architecture and programming of the Timer 1 module on all TMS370 devices.

This section covers the following topics:

7.1 Timer 1 Overview

The Timer 1 module of the TMS370 family provides enhanced timer resources to perform real-time system control. The Timer 1 Overview contains the following subsections:

7.1.1 Introduction

This module contains a general-purpose timer (T1) and a Watchdog timer (WD). Both T1 and WD allow program selection of input clock sources (real-time, external event, or pulse accumulate) with multiple 16-bit registers (input capture and compare) for special timer function control. These timers provide the capabilities for:

System Requirements	Timer Resource
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
Pulse-Width Modulated Output Control	PWM Output Function
System Integrity	Watchdog Function

FEATURES

- 16-bit General Purpose Counter
  - 16-bit Compare Register
  - 16-bit Capture/Compare Register
  - External Clock Source / Event Counter / Pulse Accumulator
  - Internal or External Counter Reset
  - Programmable Pulse Width Modulated (PWM) Output
- Selectable Edge Detection Input
- Programmable Interrupts
- Three I/O Pins
- Watchdog Timer

7.1.2 Major Components

The Timer 1 Module consists of three major blocks as shown in Figure 7-1:

Prescaler/Clock Source, which determines the independent clock sources for the general purpose timer and the watchdog timer.

16-bit General Purpose Counter which 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 Counter which software can reconfigure as a simple counter/timer, an event counter, or a pulse accumulator if the watchdog feature is not needed.

The Timer 1 Module contains additional blocks as follows:

Interrupts

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

- capture,
- compare equal,
- counter overflow, or
- external edge detect.

I/O Pins

The Timer 1 Module has three I/O pins which can be dedicated for timer functions or as general purpose I/O pins. They are:

- T1EVT
- T1IC/CR
- T1PWM

When these pins are dedicated to the timer module, T1EVT is an input to the event counter or the external clock source; T1IC/CR is an input to the input capture, counter reset, or PWM circuit; and T1PWM is the Pulse Width Modulation output.

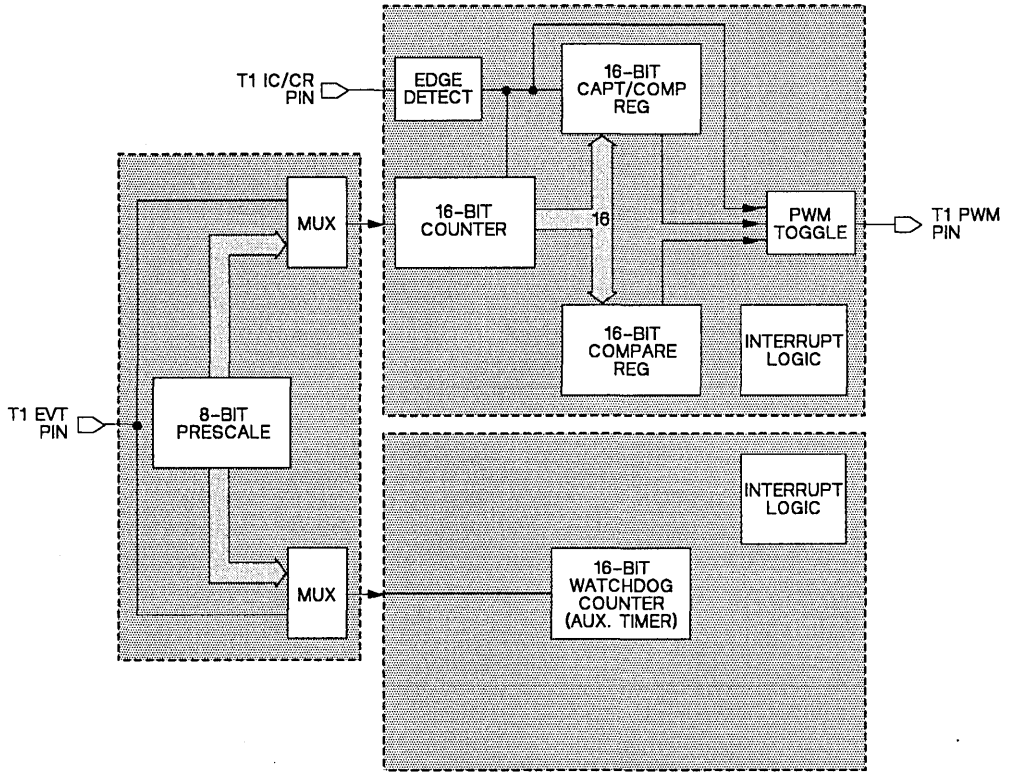


Figure 7-1. Timer 1 Block Diagram

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

PIN	DUAL COMPARE MODE	CAPTURE/COMPARE MODE
T1IC1/CR	COUNTER RESET INPUT	INPUT CAPTURE 1 INPUT
T1PWM	PWM OUTPUT	PWM OUTPUT
T1EVT	EXTERNAL EVENT INPUT OR PULSE ACCUMULATE INPUT	EXTERNAL EVENT INPUT OR PULSE ACCUMULATE INPUT

Note: pins may be used as general purpose I/O if not dedicated for timer functions.

Control Registers

Seven addressable control registers govern Timer 1. These registers:

- select the operating mode,
- enable interrupts,
- configure status flags,
- configure the I/O pins, and
- select the prescaler tap.

Timer 1 control registers (shown in Table 7-2 and described further in Section 7.5) are located at addresses P040 (1040h) to P04F (104Fh) in the Peripheral File. The location and name of each register is shown in Table 7-2.

Table 7-2. Timer 1 and Watchdog Counter Memory Map

Peripheral File Location	Symbol	Name
P040 P041	T1CNTR	Counter - MSB Counter - LSB
P042 P043	T1C	Compare Register - MSB Compare Register - LSB
P044 P045	T1CC	Capture/Compare Register - MSB Capture/Compare Register - LSB
P046 P047	WDCNTR	Watchdog Counter - MSB Watchdog Counter - LSB
P048	WDRST	Watchdog Reset Key
P049	T1CTL1	Timer 1 Control Register 1
P04A	T1CTL2	Timer 1 Control Register 2
P04B	T1CTL3	Timer 1 Control Register 3
P04C	T1CTL4	Timer 1 Control Register 4
P04D	T1PC1	Timer 1 Pin Control 1
P04E	T1PC2	Timer 1 Pin Control 2
P04F	T1PRI	Timer 1 Priority

7.1.3 Operating Modes Overview

The general-purpose Timer 1 module has two modes of operation: the Dual Compare Mode and the Capture/Compare Mode.

Dual Compare Mode

The counter is configured to provide two compare registers, external or software reset of the counter, internal or external clock source, and a programmable Pulse Width Modulated (PWM) output. The PWM output may be configured to toggle on selected events.

Capture/Compare Mode

The counter is configured to provide one input capture register and one compare register for use with the general purpose timer. The compare register may be used to provide periodic interrupts to the TMS370 CPU. The capture register may be configured to capture the current counter value upon either edge of an external input.

7.2 Timer 1 - 16-Bit, General Purpose Timer

This section describes the elements of the 16-bit General Purpose Timer (T1). The function of each block within T1 is discussed in general and for each mode of operation. Section 7.2 contains the following subsections:

7.2.1 General-Purpose Timer Operating Modes: Explains theory of operating modes.

7.2.2 Clock Prescaler/External Clock Source: Illustrates operation of the Prescaler and clock source selection circuitry.

7.2.3 Edge Detection: Explains operation of the External Edge Detection circuitry for both operating modes.

7.2.4 General Purpose Counter: Explains operation of the free running Timer 1 up counter.

7.2.5 Compare Register: Explains operation of the 16-bit Compare register.

7.2.6 Capture/Compare Register: Explains operation of the Capture/Compare register during both operating modes.

7.2.7 Interrupts: Explains interrupting capability for both operating modes.

7.2.1 General-Purpose-Timer Operating Modes

The General Purpose Timer operation mode 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

Dual Compare Mode

The Dual Compare Mode provides two compare registers, an external-resettable counter, and a timer output pin. These allow the timer to act as as interval timer, a PWM output, simple output toggle, or many other timer functions.

The Dual Compare mode as shown in Figure A-3 continuously compares the contents of the two compare registers to the current value of the 16-bit counter. If a timer compare register equals the counter, the circuit sets the associated interrupt flag to 1 and toggles the T1PWM output pin if enabled, and/or generates a Timer 1 interrupt.

A compare-equal condition from compare register 1 can also initiate a counter reset. A programmable-interval timer function (selected by using the compare equal condition to generate a system interrupt combined with the counter reset function) generates a periodic interrupt.

Either compare function may be used to toggle the T1PWM output pin when a compare-equal condition occurs, while the other compare function may be used for another system timing function. Using both compare functions to control the T1PWM pin allows direct PWM generation with minimal CPU software overhead.

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

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

1) reset the counter,
2) generate a timer interrupt, and
3) toggle the T1PWM pin to start the PWM output.

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

The input edge detect function is enabled under program control by the T1CR DET ENA bit, and upon the next occurrence of the selected edge transition:

1) the T1EDGE INT FLAG bit is set,
2) a timer interrupt is generated (if T1EDGE INT ENA =1), and
3) the T1PWM output pin is toggled (if T1CR OUT ENA = 1).

The T1EDGE POLARTIY bit selects the active input transition. In the Dual Compare mode, the edge detect function must be re-enabled after each valid edge detect.

The clock input to the counter 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 is free-running except when it receives a reset pulse from one of the following sources:

1) a 1 written to the T1 SW RESET (T1CTL2.0) bit,
2) a compare equal condition from the dedicated T1 compare function,
3) system RESET, or
4) an external pulse on the T1IC/CR pin (Dual Compare mode).

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

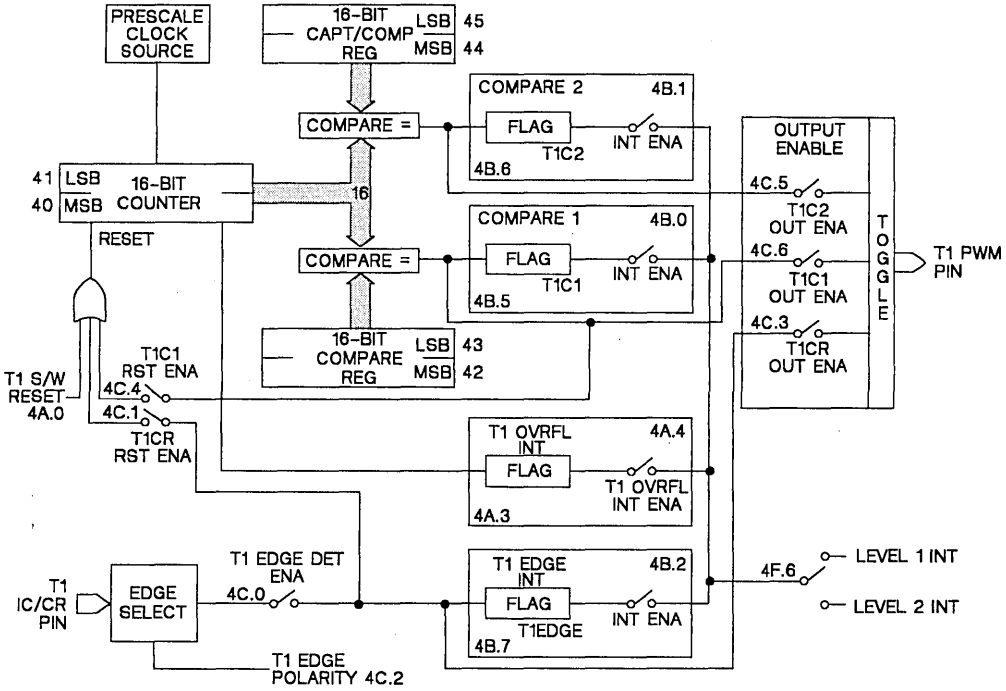


Figure 7-2. Dual Compare Mode

Capture/Compare Mode

In the Capture/Compare mode (T1 MODE = 1), Timer 1 provides one input capture register for external timing and pulse-width measurement, and one compare register for use as a programmable interval timer. In this mode, the compare register functions the same as in the Dual Compare mode described previously, including the ability to toggle the PWM pin. The capture/compare register functions in this mode as a 16-bit input capture register, as shown in Figure A-5. On the occurrence of a valid input on the T1IC/CR pin:

1) the current counter value is loaded into the 16-bit input capture register,
2) the T1EDGE INT FLAG is set, and
3) a timer interrupt is generated (if T1EDGE INT ENA = 1).

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.

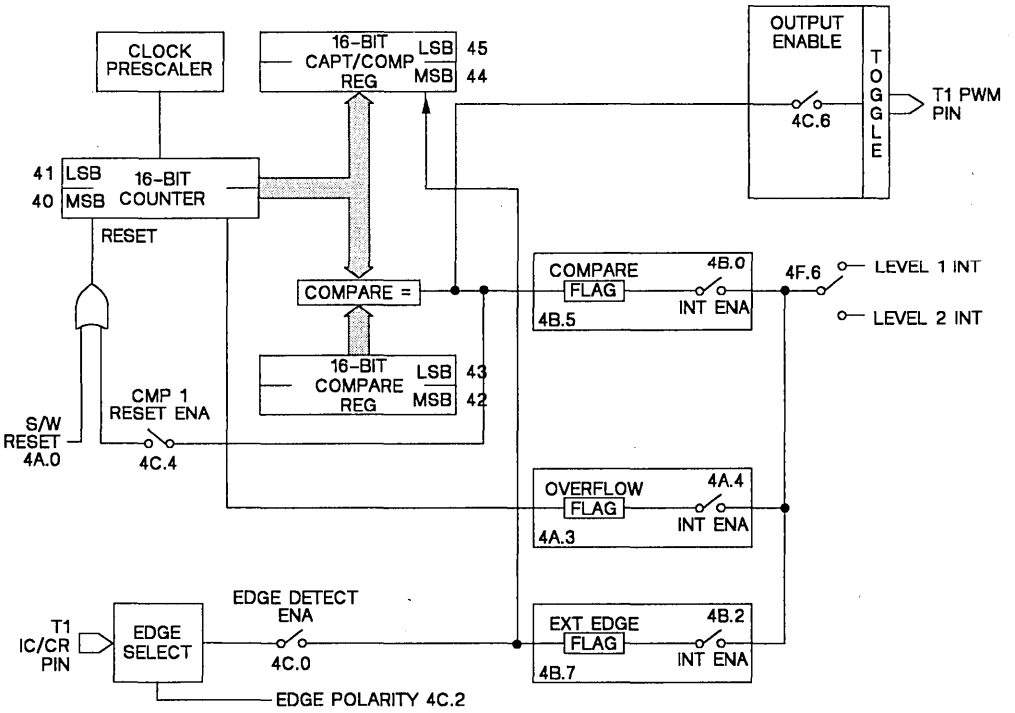


Figure 7-3. Capture/Compare Mode

7.2.2 Clock Prescaler/External Clock Source

This block, as illustrated in Figure A-4, allows selection of the clock inputs to the General Purpose Counter and the Watchdog Counter independently. Each counter has three bits in the T1CTL1 Register (see Section 7.5.1) which determine whether the counter is clocked by one of the prescaled system clock values or the external clock source (T1EVT).

The counter clock sources are as follows:

- system clock with no prescale.
- no clock, in which the counter is stopped.
- external source synchronized with the system clock (event counter operation).
- system clock while the external input is high (pulse accumulation).
- one of four taps from the prescaler which provide a system clock divided by 4, 16, 64, or 256.

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

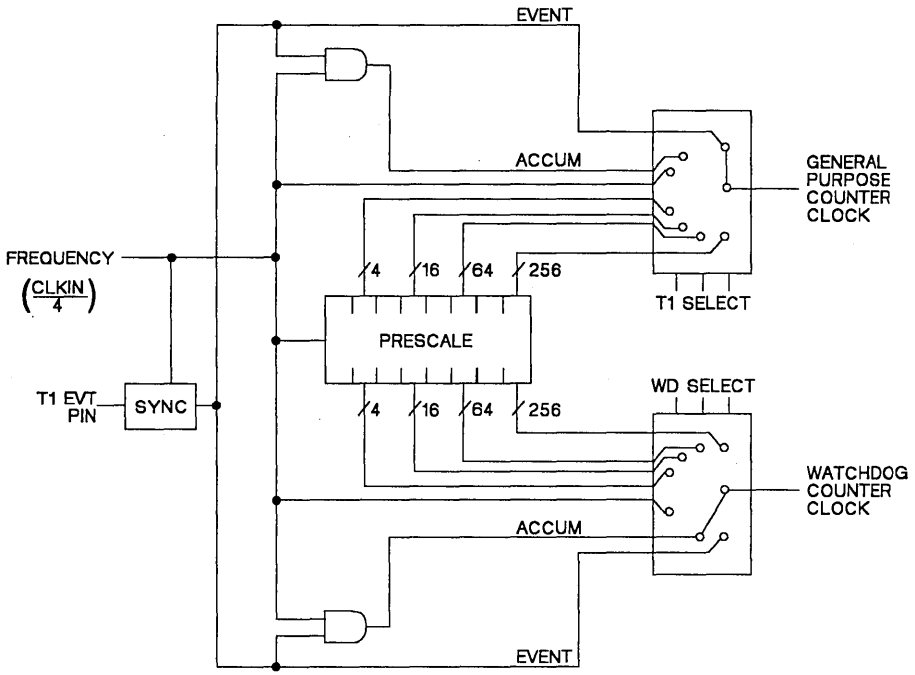


Figure 7-4. Timer 1 System Clock Prescaler

The event input is not routed through the prescaler; thus the Timer 1 module can use different taps of the prescaler for Timer 1 and the Watchdog Timer.

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

$$\text{Maximum Counter Duration (seconds)} = 2^{16} * PS * SYSCLK$$

$$\text{Counter Resolution} = PS * SYSCLK$$

where: SYSCLK = 4 / CLKIN
PS = 1 for no prescale
= 4 for divide by 4
= 16 for divide by 16
= 64 for divide by 64
= 256 for divide by 256

Table 7-3 gives the real-time counter overflow rates for various crystal and prescaler values.

Software can also configure the overflow rates for the Watchdog Counter as shown in Table 7-3 or the value shown divided by two if the WD OVRFL TAP SEL bit (T1CTL1.7) is set (see Section 7.3). This bit effectively sets the Watchdog Counter as either a 15-bit counter when set or a 16-bit counter when cleared.

Table 7-3. Counter Overflow Rates

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

†Time is given in seconds.
‡Not applicable.

The event counter input senses a low-to-high transition on the T1EVT pin while in the event-counter mode, and senses a high level (true) on the pin while in the pulse-accumulator mode.

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

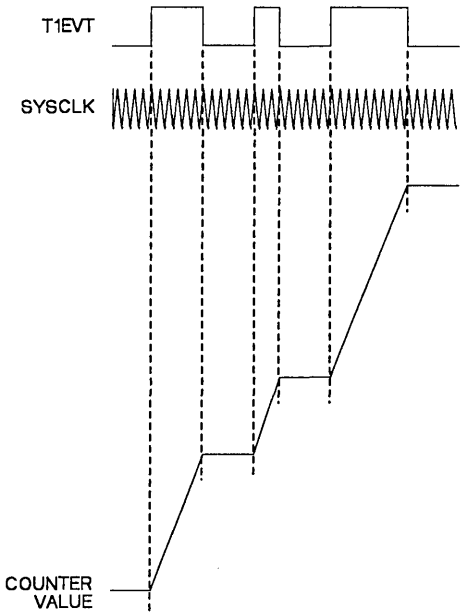


Figure 7-5. Pulse Accumulation

7.2.3 Edge Detection

The edge detection circuitry senses an active pulse transition on the Timer 1 Input-Capture/Counter-Reset pin (T1IC/CR), and provides appropriate output transitions to the rest of the module. 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 one 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 (T1CTL4.0) 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 T1 EDGE INT FLAG resets the counter. If the T1CR OUT ENA bit (T1CTL4.3) is set, the T1 EDGE INT FLAG toggles the T1PWM output latch.

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

Capture/Compare Mode

When the appropriate (rising or falling) transition is detected, the Edge Detection circuit signals the capture register to load the current counter value if the T1 EDGE DET ENA bit is set. The T1EDGE POLARITY bit (T1CTL4.2) determines which edge of the signal on the T1EVT pin to detect.

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

7.2.4 General Purpose Counter

The counter is a free-running, 16-bit up-counter, clocked by the output of the Prescaler/Clock source. During initialization, the counter is loaded with 0000h and begins its up-count. If the counter is not reset before reaching FFFFh, the counter rolls over to 0000h and continues counting. Upon counter roll-over, the T1 OVRFL INT FLAG (T1CTL2.3) is set, and a timer interrupt is generated if the T1 OVERFL INT ENA (T1CTL2.4) bit is set (see note, page 7-22).

The counter may be reset to 0000h during counting by either:

1) a 1 written to the T1 SW RESET (T1CTL2.0) bit,
2) a compare equal condition from the dedicated T1 compare function,
3) system RESET, or
4) an external pulse on the T1IC/CR pin (Dual Compare mode).

The designer may select through software (T1EDGE POLARITY bit) which external transition on the T1IC/CR pin, low-to-high or high-to-low, will reset the counter.

7.2.5 Compare Register

The Compare Register circuit consists of a 16-bit wide, read/write data register and logic to compare the counter's current value with the value stored in the compare register. The program can access the 16-bit compare register at P042 (Compare Register MSB) and P043 (Compare Register LSB) in the Peripheral File frame (see note, 7-22).

When the counter's value matches the compare register value, the following events occur:

1) the T1C1 INT FLAG bit (T1CTL3.5) is set,
2) the compare register generates a counter reset signal if the T1C1 RST ENA bit (T1CTL4.4) is set,
3) the output latch to T1PWM toggles if CMP 1 is enabled, and
4) an interrupt is generated if enabled (T1C1 INT ENA).

The Compare Register is initialized to 0000h following RESET.

Note:

If the counter is programmed to reset when its value equals the content 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 the compare equal value. Thus, there could be a delay of up to 256 system clock cycles (depending on the prescale tap in use) from the time 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 may not be reset on the following counter clock.

7.2.6 Capture/Compare Register

The Capture/Compare register for Timer 1 is a 16-bit wide register which can serve one of two functions depending on the operating mode. The Capture/Compare register is located at address P044 (Capture/Compare register MSB) and P045 (Capture/Compare register LSB) in the Peripheral File (see note, 7-22).

Dual Compare Mode

In the Dual Compare mode, the 16-bit Capture/Compare Register acts as a compare register. This compare register functions exactly as the one described in Section 7.2.5 except that it cannot reset the counter. When an output compare equal occurs, the T1C2 INT FLAG bit (T1CTL3.6) is set.

In the Dual Compare mode, the Capture/Compare register is a read/write register. Compare logic generates a pulse when the the counter value matches the Capture/Compare Register value. This pulse:

1) sets the T1C2 INT FLAG bit (T1CTL3.6),
2) clocks the output latch to T1PWM if the T1C2 OUT ENA bit (T1CTL4.5) is enabled, and
3) generates an interrupt (T1C2) if T1C2 INT ENA (T1CTL3.1) is enabled.

Capture/Compare Mode

In this mode, the edge detection signal captures the current counter content, loads it into the 16-bit Capture/Compare register, and sets the T1EDGE INT FLAG bit (T1CTL3.7).

7.2.7 Interrupts

Dual Compare

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

1) compare equal from Compare Register 1 if the T1C1 INT ENA bit (T1CTL3.0) is set,
2) compare equal from Compare Register 2 if the T1C2 INT ENA bit (T1CTL3.1) is set,
3) counter overflow if the T1 OVRFL INT ENA bit (T1CTL2.4) is set, or
4) edge detect is set if the T1EDGE INT ENA bit (T1CTL3.2) is set.

Capture/Compare

In the Capture/Compare mode, three separate events can generate an interrupt. These interrupts are:

1) output compare equal if the T1C1 INT ENA bit (T1CTL3.0) is set,
2) counter overflow if the T1 OVRFL INT ENA bit (T1CTL2.4) is set, and
3) input capture acknowledge if the T1EDGE INT ENA (T1CTL3.2) bit is set.

Note:

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

7.3 Watchdog Timer

The Watchdog Timer, shown in Figure A-6, consists of the following blocks:

- 16-bit, Watchdog/Event Counter which provides up to 2^{24} clock cycles between counter resets depending on the prescaler tap used. The program can read the contents of this counter at locations P046 (MSB) and P047 (LSB) in the Peripheral File.
- Prescaled clock input selection or external clock, the same as the General Purpose Timer.
- Watchdog Reset key which provides protection against illegal resets.
- An Overflow flag which the program may read following RESET to determine if the Watchdog caused the reset.
- Programmable interrupt and system RESET.

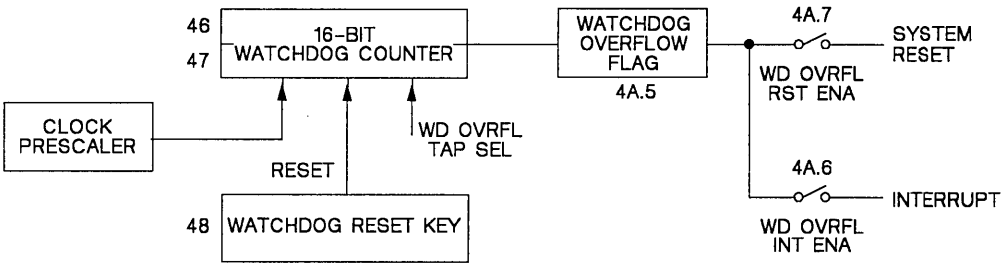


Figure 7-6. Watchdog Timer

7.3.1 Watchdog Counter

The watchdog timer is a free-running 16-bit resettable up-counter clocked by the output of the Prescaler/Clock source. The timer is software configured as either a watchdog timer to protect against system software failures and corruption, or as a simple counter/timer if the watchdog function is not desired. The 16-bit up-counter is programmable (with the WD OVRFL TAP SEL bit) to set the initial count at either 0000h or 8000h. The current value of the watchdog timer may be read at anytime during its operation (see note, page 7-22).

Watchdog Mode

In the Watchdog mode (WD OVRFL RST ENA = 1), the WD timer generates a system reset if the counter overflows or if the WD timer is reinitialized by an incorrect value. The required re-initialization frequency is determined by the system clock frequency, the prescaler/clock source selected, and whether the WD OVRFL TAP SEL bit is set for 15 or 16 bit counter rollover.

With a 20 MHz clock, the watchdog-timer overflow rates range from 6.55 ms to 3.35 seconds. These values are selected prior to entering 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 powerup reset, which enhances watchdog-timer system integrity.

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

A write of any value other than the correct predefined value to the watchdog reset key is interpreted as a lost program and a system reset is initiated. A watchdog-timer overflow or incorrect reset key sets the WD OVRFL INT FLAG bit to 1 and may be interrogated by the program following system reset to determine the source of the reset.

Non-Watchdog Mode

In the Non-Watchdog mode (WD OVRFL RST ENA = 0), the watchdog timer may be used as an event counter, pulse accumulator, or as an interval timer. In this mode, the system reset function is disabled. The watchdog timer may be re-initialized by writing 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 OVERFL TAP SEL bit, and continues counting. Upon counter overflow, the WD OVRFL INT FLAG is set and a timer interrupt is generated if the WD OVRFL INT ENA bit set. Alternatively, an external input on the T1EVT pin may be used with the watchdog timer to provide an additional 16-bit event counter or pulse accumulator.

7.3.2 Power-up RESET

After a system power-up RESET, the Watchdog Counter resets to the non-watchdog mode configured as a simple up-counter with the system clock (no prescale) as its input. Thus, if the watchdog mode is used, the program must explicitly enable it (by setting WD OVRFL RST ENA). The Watchdog Counter resets to 0000h when the WD OVRFL RST ENA bit (T1CTL2.7) is set.

Example 7-1. Watchdog Initialization Example

The following routine initializes the Watchdog Timer to generate a system reset when the counter overflows. The Watchdog counter is set to 16 bits in length and the full 8-bit prescale tap is used.

```
.
.
;Set up Watchdog Timer for a 24-bit countdown time.
;
OR    #70h,P049    ;Set the Watchdog Overflow Tap to 16 bits
                   ;and select the /256 prescale value
OR    #C0h,P049    ;Watchdog Timer Reset is enabled along
                   ; with clearing and enabling the
                   ; Watchdog Timer interrupt.
.
.
The Watchdog Timer has now been initialized to cause a
system RESET if the counter is not reset before reaching
FFFFh. To reset the counter, the code must write an
alternating 55h and AAh, starting with 55h, to the
Watchdog Timer Reset Key register (P048), e.g.:
.
.
MOV   #55h,P048    ;First write to WD RESET KEY
.
.
MOV   #0AAh,P048   ;Next write to WD RESET KEY
.
.
MOV   #55h,P048 ;Next write to WD RESET KEY
.
.
```

7.3.3 Reset Frequency

When the Watchdog timer overflows, it pulls the RESET line low to cause a system reset and sets the WD OVRFL INT FLAG bit (T1CTL2.5). The required reset frequency of the watchdog timer is determined by the value of the clock prescaler selected to clock the Watchdog Counter and by the choice of whether the overflow tap is set for a 15 or 16 bit counter. The program must set these choices before entering the watchdog mode.

The overflow tap is selected by the WD OVRFL TAP SEL bit (T1CTL1.7). When WD OVRFL TAP SEL is cleared, the Watchdog Counter is a full 16 bit counter. When WD OVRFL TAP SEL is set, the most-significant bit remains set, the Counter behaves as a 15-bit counter, and overflow occurs twice as often as in the 16-bit configuration.

The watchdog overflow rates are the same as given in Table 7-3, page 7-12 when configured as a 16-bit counter (WD OVRF TAP SEL = 0). Divide the rates in Table 7-3 in half when the timer is configured as a 15-bit counter (WD OVRFL TAP SEL = 1).

7.3.4 Overflow Flag

Watchdog Mode When the Watchdog Counter initiates a RESET, it sets the WD OVRFL INT FLAG bit (T1CTL2.5). The program may read this flag after a RESET to determine the source of the RESET. The program must clear this flag by writing a zero to the WD OVRFL INT FLAG bit.

Non-Watchdog Mode Upon overflow, the module sets the WD OVRFL INT FLAG bit (T1CTL2.5). This causes an interrupt if the WD OVRFL INT ENA bit (T1CTL2.6) is set.

7.4 Low-Power Modes

The Timer 1 module supports extended operating states which aid in reducing power consumption during periods of inactivity. These two states are the Halt and the Standby modes. For more information on Powerdown modes, see Section 4.1.4, page 4-4.

7.4.1 Halt

The Halt Mode is entered when the CPU executes an IDLE instruction while the Halt/Standby bit (SCCR2.7) and the Powerdown/IDLE bits (SCCR2.6) are set. During the Halt Mode, the Timer 1 Module clears the interrupt enable bits, but holds the pre-Halt status of all other storage elements.

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

When the Halt state terminates, the Timer 1 Module continues where it left off.

7.4.2 Standby

Standby Mode is entered by the CPU executing a IDLE instruction when the Powerdown/Idle (SCCR2.6) bit is set and the Halt/Standby bit (SCCR2.7) is cleared. During the Standby Mode, the Watchdog Counter clock input is halted while the rest of the Timer 1 Module remains fully functional.

7.5 Control Registers

Seven registers control the configuration of Timer 1 global functions, prescale values, watchdog timing, optional uses for the associated I/O pins, and other counter functions. The bits shown in shaded boxes in Figure 7-7 are Privilege Mode bits, that is, they can only be written to in the Privilege Mode.

Note:

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

Read: LSB then MSB
Write: MSB then LSB

PERIPHERAL FILE FRAME 4: TIMER 1 CONTROL REGISTERS

ADDR	PF	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	
1040h	040	BIT 15			COUNTER MSB				BIT 8	T1CNTR
1041h	041	BIT 7			COUNTER LSB				BIT 0	
1042h	042	BIT 15			COMPARE REGISTER MSB				BIT 8	T1C
1043h	043	BIT 7			COMPARE REGISTER LSB				BIT 0	
1044h	044	BIT 15			CAPTURE/COMPARE REGISTER MSB				BIT 8	T1CC
1045h	045	BIT 7			CAPTURE/COMPARE REGISTER LSB				BIT 0	
1046h	046	BIT 15			WATCHDOG COUNTER MSB				BIT 8	WDCNTR
1047h	047	BIT 7			WATCHDOG COUNTER LSB				BIT 0	
1048h	048	WATCHDOG RESET KEY								WDRST
1049h	049	WD OVRFL TAP SEL	WD INPUT SELECT 2	WD INPUT SELECT 1	WD INPUT SELECT 0	---	T1 INPUT SELECT 2	T1 INPUT SELECT 1	T1 INPUT SELECT 0	T1CTL1
104Ah	04A	WD OVRFL RST ENA	WD OVRFL INT ENA	WD OVRFL INT FLAG	T1 OVRFL INT ENA	T1 OVRFL INT FLAG	---	---	T1 SW RESET	T1CTL2

MODE: DUAL COMPARE REGISTERS WITH EVENT COUNTER

104Bh	04B	T1EDGE INT FLAG	T1C2 INT FLAG	T1C1 INT FLAG	---	---	T1EDGE INT ENA	T1C2 INT ENA	T1C1 INT ENA	T1CTL3
104Ch	04C	T1MODE = 0	T1C1 OUT ENA	T1C2 OUT ENA	T1C1 RST ENA	T1CR OUT ENA	T1EDGE POLARITY	T1CR RST ENA	T1EDGE DET ENA	T1CTL4

MODE: SINGLE CAPTURE AND COMPARE REGISTERS

104Bh	04B	T1EDGE INT FLAG	---	T1C1 INT FLAG	---	---	T1EDGE INT ENA	---	T1C1 INT ENA	T1CTL3
104Ch	04C	T1MODE = 1	T1C1 OUT ENA	---	T1C1 RST ENA	---	T1EDGE POLARITY	---	T1EDGE DET ENA	T1CTL4

104Dh	04D	---	---	---	---	T1EVT DATA IN	T1EVT DATA OUT	T1EVT FUNCTION	T1EVT DATA DIR	T1PC1
104Eh	04E	T1PWM DATA IN	T1PWM DATA OUT	T1PWM FUNCTION	T1PWM DATA DIR	T1IC/CR DATA IN	T1IC/CR DATA OUT	T1IC/CR FUNCTION	T1IC/CR DATA DIR	T1PC2
104Fh	04F	T1 STEST	T1 PRIORITY	---	---	---	---	---	---	T1PRI

Figure 7-7. Peripheral File Frame 4 - Timer 1 Control Registers

7.5.1 Timer 1 Counter Control Register 1

The T1CTL1 Register controls the prescaler inputs to the Watchdog counter and the general purpose counter. The bit assignments and definitions follow:

Timer 1 Control Register 1 (T1CTL1)
[Memory address - 1049h]

Bit # -	7	6	5	4	3	2	1	0
P049	WD OVRFL TAP SEL	WD INPUT SELECT 2	WD INPUT SELECT 1	WD INPUT SELECT 0	---	T1 INPUT SELECT 2	T1 INPUT SELECT 1	T1 INPUT SELECT 0
	RP-0	RP-0	RP-0	RP-0		RW-0	RW-0	RW-0

R=Read, W=Write, P=Write Protected when WD OVRFL RST ENA=1, -n=Value after RESET

Bits 0-2 - **T1 INPUT SELECT 0–2.** Timer 1 Input Select 0–2.
These three bits select one of eight possible clock sources for the Timer 1 general purpose counter. These sources are:

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

The combinations are shown below.

2	1	0	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	" " / 16
1	1	0	" " / 64
1	1	1	" " / 256

Bit 3 - Reserved. Read data is indeterminate.

Bits 4-6 - **WD INPUT SELECT 0–2.** Watchdog Input Select 0–2.
These three bits select one of eight possible clock sources for the Watchdog counter. These sources and the bit combinations to select the sources are the same as listed above for the General Purpose Counter. Once the WD OVRFL RST ENA bit is set, the values of these bits can only be changed after a Power-Up RESET.

Bit 7 - **WD OVRFL TAP SEL.** Watchdog Overflow Tap Select.
This bit determines whether the Watchdog Counter is to operate as a 15 bit or a 16 bit counter. The default is the full 16 bits of the counter. If a shorter Watchdog Counter overflow rate is desired, the most significant bit of the counter can be forced to remain at a 1. This, in effect, changes the Watchdog Counter to a 15-bit counter with an overflow period 1/2 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. This bit is cleared by a) a Power-Up RESET, or b) any RESET while WD OVRFL RST ENA=0 (Non-Watchdog Mode).

0 = 16-bit Watchdog Counter overflow.
1 = 15-bit Watchdog Counter overflow.

7.5.2 Timer 1 Counter Control Register 2

The T1CTL2 register controls the Timer 1 and Watchdog overflow interrupts, and contains the Timer 1 software reset bit. A summary of the bit assignments and definitions is shown below.

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

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

R=Read, S=Set Only, W=Write, C=Clear Only, -n=Value after RESET,
†-see bit 5 description

Bit 0 - **T1 SW RESET**. Timer 1 Software Reset.
This bit is always read as a zero; however, when a one is written to this bit, the counter resets to 0000h on the next system clock cycle.

Bit 1,2 - Reserved. Read values are indeterminate.

Bit 3 - **T1 OVRFL INT FLAG**. Timer 1 Overflow Interrupt Flag
This bit indicates the status of the T1 Overflow Interrupt. This bit is cleared by RESET or by writing a zero to it.
0 = General Purpose Overflow interrupt inactive.
1 = General Purpose Overflow interrupt pending.

Bit 4 - **T1 OVRFL INT ENA**. Timer 1 Overflow Interrupt Enable.
This bit controls the Timer 1 Overflow interrupting capability.
0 = Disable Interrupt.
1 = Enable Interrupt.

Bit 5 - **WD OVRFL INT FLAG**. Watchdog Overflow Interrupt Flag.
This bit indicates the status of the Watchdog overflow interrupt. Clear this bit by writing a zero to it. This bit is NOT cleared following a Watchdog initiated RESET. Thus it may be read and cleared, to determine the cause of the RESET. This bit is cleared by power-up RESET or by any reset if WD OVRFL RST ENA = 0. Once the WD OVRFL RST ENA bit is set, the values of these bits can only be changed after a Power-Up RESET.
0 = Watchdog Interrupt Inactive
1 = Watchdog Counter has overflowed or the incorrect value is written to the Watchdog Reset Key register while in Watchdog mode.

Bit 6 - **WD OVRFL INT ENA**. Watchdog Overflow Interrupt Enable.
This bit controls the Watchdog Overflow interrupting capability. Once the WD OVRFL RST ENA bit is set, the values of these bits can only be changed after a Power-Up RESET.
0 = Watchdog Interrupt Disabled.
1 = Watchdog Interrupt Enabled.

Bit 7 - **WD OVRFL RST ENA** Watchdog Overflow Reset Enable.
This bit controls the ability of a Watchdog overflow to generate a RESET. When set, this bit determines the function of the Watchdog Counter; either as a the Watchdog Counter, or as a simple up counter or event counter/pulse accumulator. Once set, this bit can only be cleared by a Power-Up RESET, and locks the values of other WD bits so they can only be changed during Power-up RESET.
0 = Watchdog Counter does *not* initiate a RESET upon overflow.
1 = Watchdog Counter *does* initiate a RESET upon overflow.

7.5.3 Timer 1 Counter Control Register 3

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

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

Mode: Dual Compare

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

Mode: Capture/Compare

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

R=Read, W=Write, C=Clear Only, -n=Value after RESET

Bit 0 - **T1C1 INT ENA.** Timer 1 Compare 1 Interrupt Enable.
This bit determines whether or not the compare register flag can generate an interrupt.
0 = Disable interrupt.
1 = Enable interrupt.

Bit 1 - **T1C2 INT ENA.** Timer 1 Compare 2 Interrupt Enable.
Dual Compare mode only: This bit determines whether or not the Capture/Compare register flag can generate an interrupt.
0 = Disable interrupt.
1 = Enable interrupt.

Capture/Compare Mode: Read data is indeterminate.

Bit 2 - **T1EDGE INT ENA.** Timer 1 Edge Interrupt Enable.
This bit determines whether or not the active edge input to the T1IC/CR pin generates an interrupt. The T1EDGE DET ENA bit (T1CTL4.0) must be set before an edge can be detected.
0 = Disable interrupt.
1 = Enable interrupt.

Bits 3,4 - Reserved. Read data is indeterminate.

Bit 5 - **T1C1 INT FLAG.** Timer 1 Compare 1 Interrupt Flag.
This bit is set when the compare register first matches the counter value. It is cleared by writing a zero to this bit, or during RESET.

0 = Interrupt inactive.
1 = Interrupt pending.

Bit 6 - **T1C2 INT FLAG** Timer 1 Compare 2 Interrupt Flag.
Dual Compare Mode: This bit is set when the Capture/Compare register first matches the counter value. It is cleared by writing a zero to this bit or by RESET.
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. The T1EDGE INT FLAG is cleared by writing a zero to the bit, or during RESET.
0 = no transition
1 = transition detected

7.5.4 Timer 1 Counter Control Register 4

The T1CTL4 register controls the mode of operation, and various functions of the Timer 1 input and output pins. The bits in this register serve different functions depending on the mode, as shown below:

Timer 1 Counter Control Register 4 (T1CTL4)
[Memory Address - 104Ch]

Mode: Dual Compare

Bit # -	7	6	5	4	3	2	1	0
P04C	T1 MODE =0	T1C1 OUT ENA	T1C2 OUT ENA	T1C1 RST ENA	T1CR OUT ENA	T1 EDGE POLARITY	T1CR RST ENA	T1EDGE DET ENA
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Mode: Capture/Compare

Bit # -	7	6	5	4	3	2	1	0
P04C	T1 MODE =1	T1C1 OUT ENA	---	T1C1 RST ENA	---	T1 EDGE POLARITY	---	T1EDGE DET ENA
	RW-0	RW-0		RW-0		RW-0		RW-0

R- Read, W- Write, -n = Value after RESET

The function of the bits are as follows.

Bit 0 - **T1EDGE DET ENA.** Timer 1 Edge Detect Enable.
Dual Compare Mode: This bit enables the edge detection circuit to sense the next level transition on the Timer 1 T1IC/CR pin. This bit is cleared after the selected transition is detected or during RESET.

0 = Edge detect disabled
1 = Edge detect enabled.

Capture/Compare Mode: This bit enables the input capture circuit to capture the current counter value upon the next level transition on the counter reset/input capture pin, as determined by the T1EDGE POLARITY bit. This bit remains unchanged after the selected transition is detected.

0 = Input capture disabled.
1 = Input capture enabled.

Bit 1 - **T1CR RST ENA.** Timer 1 External Reset Enable.
Dual Compare Mode: This bit determines whether or not an external signal can reset the counter.

0 = Disable external reset of the counter.
1 = Enable external reset of the counter on the next valid edge detect.

Capture/Compare Mode: Reserved. Read data is indeterminate.

Bit 2 - **T1EDGE POLARITY.** Timer 1 Edge Polarity.
This bit determines the transition direction on the Timer 1 T1IC/CR pin to trigger a capture or counter reset, depending on the counter mode selected.

0 = Trigger on a high-to-low transition.
1 = Trigger on a low-to-high transition.

Bit 3 - **T1CR OUT ENA.** Timer 1 External Edge Output Enable.
Dual Compare Mode: This bit determines whether or not the input signal on the T1IC/CR pin can toggle the output signal on the T1PWM pin.

0 = Disable pulse to toggle output.
1 = Enable 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 the Compare Register 1 is equal to the Counter, the Counter will reset on the next counter increment.

0 = Disable counter reset upon compare equal.
1 = Enable 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 the Compare Register 2 is equal to the Counter, the T1PWM pin toggles (when configured as a PWM pin).

0 = Disable pulse to toggle output.
1 = Enable 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 = Disable pulse to toggle output.
1 = Enable 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.

7.5.5 Timer 1 Port Control Registers

Port Control Registers T1PC1 and T1PC2 are organized to allow all functions for a pin to be programmed in one write cycle. Each module pin is controlled by a nibble in one of the PCRs. A summary of the Port Control Register functions and bit assignments is shown below.

7.5.5.1 Timer 1 Port Control Register 1

The T1PC1 register controls the I/O functions of the Timer 1 Module, T1EVT pin.

Timer 1 Port Control Register 1 (T1PC1)
[Memory Address - 104Dh]

Bit #-	7	6	5	4	3	2	1	0
P04D	---	---	---	---	T1EVT DATA IN	T1EVT DATA OUT	T1EVT FUNCTION	T1EVT DATA DIR
					R-0	RW-0	RW-0	RW-0

R=Read, W=Write, -n=Value after RESET

Bit 0 - **T1EVT DATA DIR**. Timer 1 Event-Pin Data Direction.
This bit selects the T1EVT pin as an input or output, if the T1EVT FUNCTION bit = 0.

0 = Enable T1EVT pin as data input.
1 = Enable T1EVT pin as data output.

Bit 1 - **T1EVT FUNCTION**. T1EVT Pin Function Select.
This bit determines the function of the T1EVT pin.

0 = T1EVT is a general-purpose digital I/O port.
1 = T1EVT is the event-input pin.

Bit 2 - **T1EVT DATA OUT**. T1EVT Pin Data Out.
This bit contains the data to be output on the T1EVT pin if the following conditions are met:
a. T1EVT DATA DIR = 1.
b. T1EVT FUNCTION = 0.

Bit 3 - **T1EVT DATA IN**. T1EVT Pin Data In.
This bit contains the data present on the T1EVT pin. A write operation to this bit has no effect.

Bits 4-7 Reserved. Read data is indeterminate.

7.5.5.2 Timer 1 Port Control Register 2

The T1PC2 register controls the I/O functions of the Timer 1 Module, T1IC/CR and T1PWM pins.

Timer 1 Port Control Register 2 (T1PC2)
[Memory Address - 104Eh]

Bit # -	7	6	5	4	3	2	1	0
P04E	T1PWM DATA IN	T1PWM DATA OUT	T1PWM FUNCTION	T1PWM DATA DIR	T1IC/CR DATA IN	T1IC/CR DATA OUT	T1IC/CR FUNCTION	T1IC/CR DATA DIR
	R-0	RW-0	RW-0	RW-0	R-0	RW-0	RW-0	RW-0

R=Read, W=Write, -n= Value after RESET

Bit 0 - **T1IC/CR DATA DIR.** T1IC/CR Pin Data Direction.
This bit selects the T1IC/CR pin as an input or output, if the T1IC/CR Function bit = 0.
0 = Enable T1IC/CR pin data input.
1 = Enable 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 port
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 = Enable T1PWM pin data input.
1 = Enable T1PWM pin data output.

Bit 5 - **T1PWM FUNCTION.** T1PMW Pin Function Select.
This bit determines the function of the T1PWM pin.
0 = the T1PWM pin is a general-purpose digital I/O port
1 = the T1PWM pin is the PWM output.

Bit 6 - **T1PWM DATA OUT.** T1PWM Pin Data Out.
This bit contains the data to be output on the T1PWM pin if the following conditions are met:
a. T1PWM DATA DIR = 1
b. T1PWM FUNCTION = 0
This bit may be used to preset the PWM output level.

Bit 7 - **T1PWM DATA IN.** T1PWM Pin Data In 1.
This bit contains the data input on pin T1PWM. A write operation to this bit has no effect.

> **Note:**
>
> See Section 13.5.1, page 13-22 for examples of PWM pin initialization.

7.5.6 Timer 1 Interrupt Priority Control Register

The T1PRI register controls the level of the Timer 1 interrupt. Software 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=Privileged Write, -n=Value after 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. This bit must be cleared (0) to ensure proper operation.

8. Timer 2 Module

This section discusses the architecture and programming of the Timer 2 module on TMS370C050 and TMS370C850 devices.

This section covers the following topics:

8.1 Timer 2 Overview

The Timer 2 module (T2), available on TMS370C050 and TMS370C850 devices, adds an additional timer for these devices that provides event count, input capture, and compare functions. Figure 8-1 shows a block diagram of the Timer 2 Module.

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
Pulse-Width Modulated Output Control	PWM Output Function

The Timer 2 Module has three I/O pins which may be reconfigured as general purpose I/O Pins for use by other parts of the microcomputer. They are:

- T2EVT
- T2IC1/CR
- T2IC2/PWM

When these pins are dedicated to the timer module, T2EVT is an input to the event counter or the external clock source, T2IC1/CR is an input to the counter reset, input capture, or PWM circuit, and T2IC2/PWM is the Pulse Width Modulation output or a second input capture.

The Timer 2 Module consists of the following blocks as shown in Figure 8-1.

- 16-bit resettable up counter,
- 16-bit Compare Register with associated compare logic,
- 16-bit Capture Register,
- 16-bit Capture/Compare Register.

The General Purpose Counter operates in one of two modes. The mode of operation determines whether the Capture/Compare Register functions as a compare register (in the Dual Compare mode) or as a capture register (the Dual Capture mode).

Timer 2 has maskable interrupts for two input captures, two output compares, counter overflow and external edge detect.

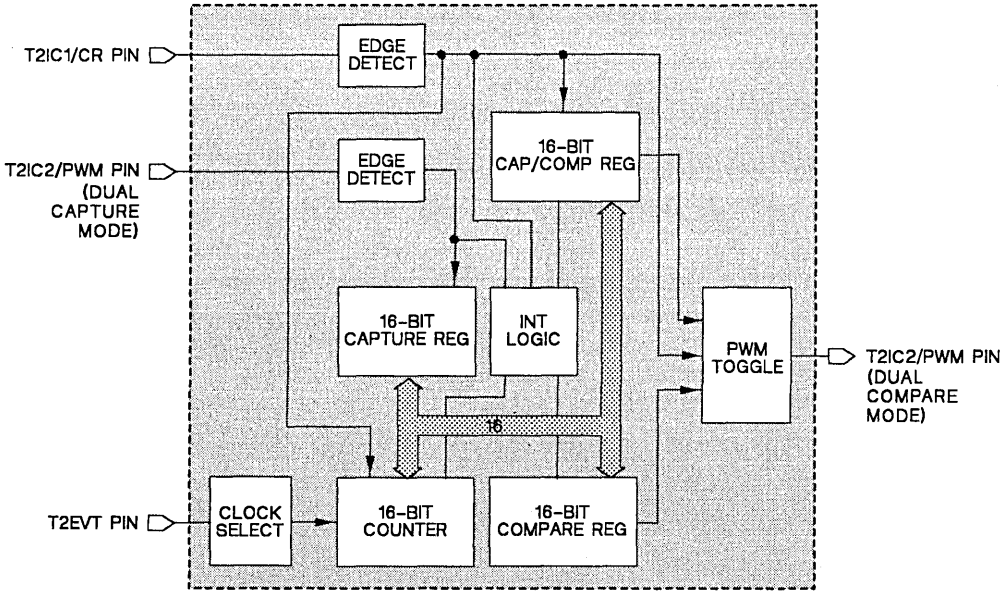


Figure 8-1. 16-Bit Programmable General Purpose Timer 2

FEATURES

- 16-BIT, General Purpose Counter
  - Compare Mode: Dual 16-Bit Compare Registers
  - Capture Mode: Dual Capture and one Compare Register
  - External Clock Source / Event Counter / Pulse Accumulator
  - Internal or External Counter Reset
  - Programmable Pulse Width Modulated Output
- Selectable Edge Detection Input
- Programmable Interrupts
- Three programmable I/O Pins

Timer 2 Operating Modes:

Dual Compare Mode: The counter is configured to provide dual compare registers, external or software reset of the counter, internal or external clock source, and a programmable Pulse Width Modulated (PWM) output. The T2IC2/PWM pin may also be configured to toggle upon an external input edge. The external clock source may be selected for use as an event counter or pulse accumulator.

Dual Capture Mode: The counter is configured to provide dual input capture registers and one compare register for use as a general purpose timer. The Compare Register may be used to provide periodic interrupts to the rest of the microcomputer. Each capture register may be configured to capture the current counter value upon either edge of an external input.

Timer 2 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 Figure 8-2.

Peripheral File Location	Symbol	Name
P060 P061	T2CNTR	T2 Counter - MSB T2 Counter - LSB
P062 P063	T2C	T2 Compare 1 Register - MSB T2 Compare 1 Register - LSB
P064 P065	T2CC	T2 Capture 1/Compare Register 2 - MSB T2 Capture 1/Compare Register 2 - LSB
P066 P067	T2IC	Capture Register 2 - MSB Capture Register 2 - LSB
P068		Reserved
P069		Reserved
P06A	T2CTL1	Timer 2 Control Register 1
P06B	T2CTL2	Timer 2 Control Register 2
P06C	T2CTL3	Timer 2 Control Register 3
P06D	T2PC1	Timer 2 Pin Control 1
P06E	T2PC2	Timer 2 Pin Control 2
P06F	T2PRI	Timer 2 Priority

Figure 8-2. Timer 2 Memory Map

8.2 Timer 2 Operation

The 16-bit general purpose timer, T2, 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 a compare register in the other mode. In the following paragraphs, the functions of each block within T2 is discussed in general and for each mode of operation.

8.2.1 Operation Modes

The Timer 2 Module mode of operation is determined by the T2 MODE bit (T2CTL3.7).

T2 MODE = 0 - Dual Compare Mode.
T2 MODE = 1 - Dual Capture Mode.

Dual Compare Mode
In this mode, as illustrated in Figure A-7, the timer has two compare registers, an external-resettable counter, and a timer output pin. These allow the timer to act as an interval timer, a PWM output, simple output toggle, or many other timer functions. In this mode, the Capture/Compare Register functions as a 16-bit read/write compare register. The operation of T2 is identical to T1 while operating in the Dual Compare mode with the exception of the clock sources.

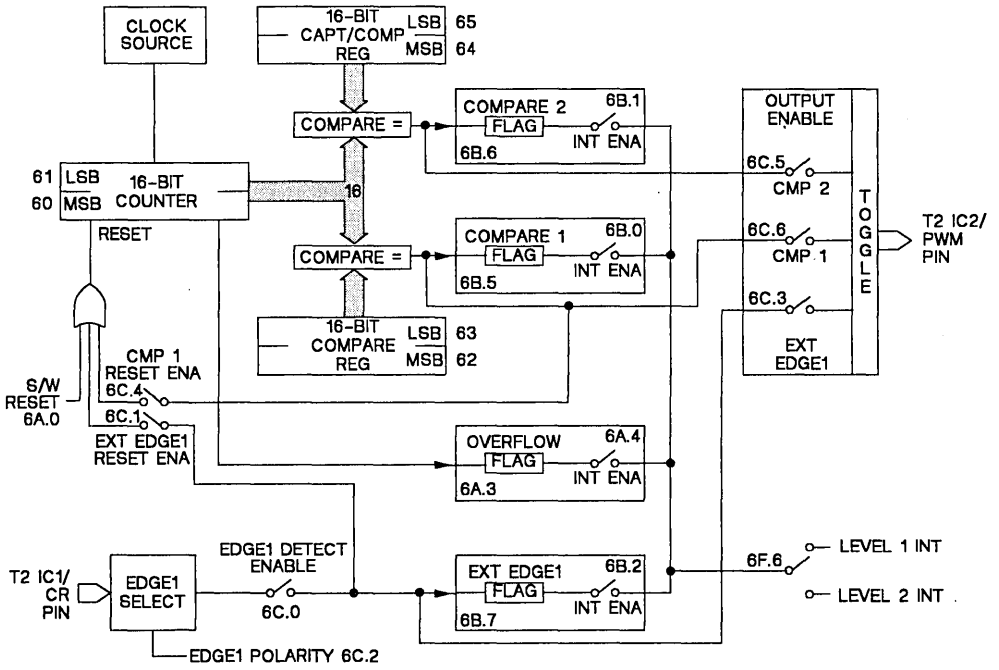


Figure 8-3. Dual Compare Mode

Dual Capture Mode

In the Dual Capture Mode, illustrated in Figure A-8, T2 is configured to provide one compare register for use as a programmable interval timer, and two input capture registers for external input timing and pulse width measurement. In this mode the Capture/Compare Register functions as 16-bit input capture register. 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 is set and a timer interrupt is generated if the respective INT ENA is set.

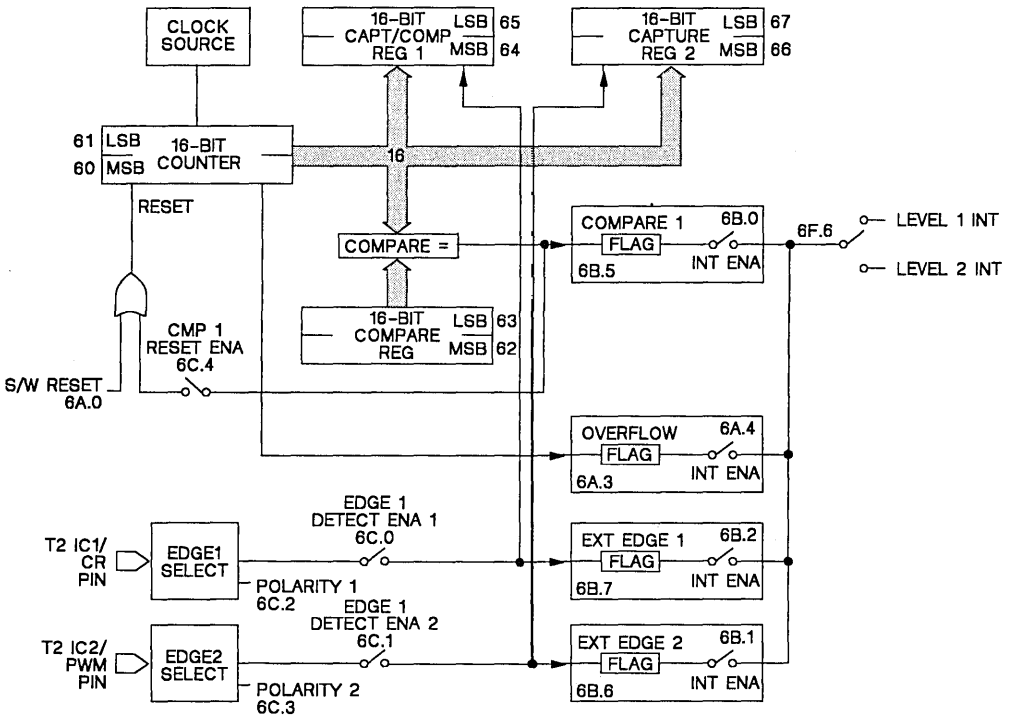


Figure 8-4. Dual Capture Mode

8.2.2 Clock Sources

Timer 2 clock sources are illustrated in Figure 8-5. The T2 INPUT SELECT 0 bit (T2CTL1.1) and the T2 INPUT SELECT 1 bit (T2CTL1.2) select one of four clock sources:

- system clock,
- no clock (in which the counter is stopped),
- external clock synchronized to the system clock (event counter), or
- system clock when external input is high (pulse accumulation).

The maximum counter duration with an internal clock is based on the internal system clock time (SYSCLK) as follows:

$$\text{Maximum Counter Duration} = 2^{16} * \text{SYSCLK}$$
$$\text{Counter Resolution} = \text{SYSCLK}$$

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

The external event frequency input to the module may not exceed CLKIN/8. All external event inputs are synchronized with the system clock.

8

When using the system clock input, the 16-bit timer generates an overflow rate of 13.1 ms with 200 ns resolution (CLKIN = 20 MHz).

Event Counter Mode

Using this clock source, the general purpose timer is programmable as a 16-bit event counter. An external low-to-high transition on the T2EVT pin is used to provide the clock for the internal timer. The T2EVT external clock frequency may not exceed the system clock frequency divided by 2.

Pulse Accumulator Mode

Using this clock source, the general purpose timer is programmable as a 16-bit pulse accumulator. An external input on the T2EVT pin is used to gate the internal system clock to the internal timers. While T2EVT input is logic one (high), the timer is clocked at the system clock rate and counts system clock pulses until the T2EVT pin returns to logic zero.

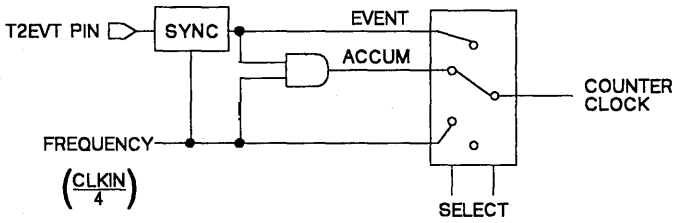


Figure 8-5. Timer 2 Clock Sources

8.2.3 Timer 2 Edge Detection Circuitry

This edge detection circuitry senses an active pulse transition on the input pins and provides appropriate output transitions to the rest of the module.

Dual Compare Mode

The edge detection circuitry is connected to the module's T2IC1/CR pin. In this mode, the program must re-enable the Timer 2 Module after each edge detection by setting the T2EDGE1 DET ENA bit (T2CTL3.0).

When the Timer 2 module detects an active transition (while enabled), the module performs the following actions:

1) clears the T2EDGE DET ENA bit,
2) sets the external edge flag, T2EDGE1 INTERRUPT FLAG (T2CTL2.7),
3) resets the counter if T2EDGE1 RST ENA bit (T2CTL3.1) is set, and
4) toggles the output flip-flop if the T2EDGE1 OUT ENA bit (T2CTL3.3) is set.

In the Dual Compare mode, the T2EDGE1 POLARITY bit (T2CTL3.2) determines whether the active transition is low-to-high or high-to-low.

Dual Capture Mode

Edge detection circuitry is connected to both the T2IC1/CR pin and the T2IC2/PWM pin.

When the Edge 1 Detect circuit detects an active edge transition on the T2IC1/CR pin:

1) the Capture/Compare Register is loaded with the current counter value, and
2) the T2EDGE1 INT FLAG bit is set.

When the Edge 2 Detect circuit detects an active edge transition on the T2IC2/PWM pin:

1) the Capture Register is loaded with the current counter value, and
2) the T2EDGE2 INT FLAG bit is set.

The T2EDGE1 POLARITY bit (T2CTL3.2) and the T2EDGE2 POLARITY bit (T2CTL3.3) determine the transition (rising or falling) to be detected.

8.2.4 16-Bit Resettable Up Counter.

The counter is a free-running, 16-bit, read-only, up-counter clocked by the system clock, external event, or system clock while an external event is active (pulse accumulate). During initialization, the counter is loaded with 0000h and begins its up-count. If the counter is not reset before reaching FFFFh, the counter rolls over to 0000h and continues counting. Upon counter roll-over, the T2 OVRFL INT FLAG (T2CTL1.3) is set, and a timer interrupt is generated if the T2OVRFL INT ENA bit (T2CTL1.4) is set.

The counter may be reset to 0000h during counting by either:

1) writing a 1 to the T2 SW RESET bit (T2CTL1.0),
2) a compare equal condition from the dedicated T2 compare function,
3) System RESET, or
4) an external pulse on the T2IC/CR pin (Dual Compare mode only).

The designer may select by software (T2CR POLARITY bit) which external transition, low-to-high or high-to-low, on the T2IC1/CR pin will cause the counter to be reset.

Special circuitry prevents the contents of the T2CNTR register from changing in the middle of a 16-bit read operation. See the note in Section 8.3. on page 8-14

8.2.5 Compare Register

The Compare Register circuit consists of a 16-bit wide, read/write data register (T2C) and logic to compare the counter's current value with the value stored in the Compare Register.

Special circuitry prevents the T2C register from changing in the middle of a 16-bit read or write operation. See the note in Section 8.3. on page 8-14

The compare logic sets T2C1 INT FLAG (T2CTL2.5) as soon as the value in the timer matches that in the Compare Register. Once T2C1 INT FLAG is set by a compare-equal condition, then cleared, it will not be set again if the same compare-equal condition still exists, i.e., the same compare-equal condition can only set the T2C1 INT FLAG once. This flag causes various events to occur depending on the mode of operation and which enable bits are set.

On a compare equal condition, the T2 module:

1) sets the T2C1 INT FLAG bit (T2CTL2.5),
2) generates an interrupt if the T2C1 INT ENA bit (T2CTL2.0) is set, and
3) resets the counter if T2C1 RST ENA bit (T2CTL3.4) is set.

In Dual Compare Mode only:

4) toggles the PWM output pin if the T2C1 OUT ENA bit (T2CTL3.6) is set.

8.2.6 Capture Register (Dual Capture Mode only)

The Capture Register is a 16-bit wide, read-only, data register (T2IC). 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 (MSB) and P067 (LSB) of the Peripheral File. Writes to this register are ignored. Thus, the Capture Register retains the last counter value captured until another input capture pulse loads a new value in the register.

On receipt of a capture pulse, the following events occur:

1) value of counter is loaded into the Capture Register,
2) the module sets the T2EDGE2 INT FLAG bit (T2CTL2.6) to indicate that the Capture Register has latched the current counter value, and
3) the module generates an interrupt if the T2C2 INT ENA bit (T2CTL2.1) is set.

Special circuitry prevents the T2IC register from changing in the middle of a 16-bit read or write operation. See the note in Section 8.3. on page 8-14

8.2.7 Capture/Compare Register.

The Capture/Compare Register (T2CC) for Timer 2 is a 16-bit wide register which can be programmed to serve one of two functions. In the Dual Capture Mode this register functions as a capture register and in the Dual Compare Registers Mode, it functions as a compare register. The Capture/Compare Register is located at addresses P064 (MSB) and P065 (LSB) of the Peripheral File.

Special circuitry prevents the T2CC register from changing in the middle of a 16-bit read or write operation. See the note in Section 8.3 on page 8-14.

Dual Compare Mode
In the Dual Compare mode, the Capture/Compare Register becomes a read/write compare register. This compare register's functions are similar to the dedicated Compare Register except that it can **not** reset the counter.

In this mode, the current counter value and the current Capture/Compare register value are directed to compare logic which generates a pulse when the two values match. This pulse is used to:

1) set the T2C2 INT FLAG bit (T2CTL2.6),
2) toggle the PWM output pin if the T2C2 OUT ENA bit (T2CTL3.5) is set, and
3) generate an interrupt if the T2C2 INT ENA bit (T2CTL2.1) is set.

Dual Capture Mode

In the Dual Capture Mode, the Capture/Compare Register becomes a read-only capture register. When an external pulse appears on pin T2IC1/CR, the following events occur if the T2EDGE1 DET ENA bit (T2CTL3.0) is set.

1) the current counter value is latched into the Capture/Compare register.
2) the T2EDGE1 INT FLAG bit (T2CTL2.7) is set.
3) an interrupt is generated if the T2EDGE1 INT ENA bit (T2CTL2.2) is set.

8.2.8 Timer-2 I/O Pin Functions

The Timer 2 module has three I/O pins which may be dedicated as timer functions or used as general purpose I/O pins. The definitions of these pins are contained in the two Port Control Registers located at addresses P06E and P06D of the Peripheral File.

Table 7-1 defines the functions of the three Timer-2 I/O pins for both operating modes.

Table 8-1. Timer 2 I/O Pin Definitions

PIN	DUAL COMPARE MODE	DUAL COMPARE MODE
T2IC1/CR	COUNTER RESET INPUT	INPUT CAPTURE 1 INPUT
T2IC2/PWM	PWM OUTPUT	INPUT CAPTURE 2 INPUT
T2EVT	EXTERNAL EVENT INPUT OR PULSE ACCUMULATE INPUT	EXTERNAL EVENT INPUT OR PULSE ACCUMULATE INPUT

8

8.2.9 Timer 2 Interrupts

Interrupts may be enabled to occur upon an input capture, output compare equal, counter overflow and/or upon an external edge detect.

Dual Compare Mode:
In this mode, interrupts are generated when any of the following events occur:

1) when a compare equal condition occurs for the dedicated Compare Register if the T2C1 INT ENA bit (T2CTL2.0) is set,
2) when a compare equal condition occurs for the Capture/Compare Register if the T2CC2 INT ENA bit (T2CTL2.1) is set,
3) when the counter overflows if the T2 OVERFL INT ENA bit (T2CTL1.4) is set, or
4) when an External Edge detect occurs if the T2EDGE1 DET ENA and T2EDGE1 INT ENA bits are set (T2CTL3.0 and T2CTL2.2 respectively).

Dual Capture Mode:
In this mode, interrupts are generated when any of the following events occur:

1) when a compare equal condition occurs for the dedicated Compare Register if the T2C1 INT ENA bit (T2CTL2.0) is set,
2) when the counter overflows if the T2 OVERFL INT ENA bit (T2CTL1.4) is set,
3) when an External Edge 1 detect occurs if the T2EDGE1 DET ENA and T2EDGE1 INT ENA bits are set (T2CTL3.0 and T2CTL2.2), or
4) when an External Edge 2 detect occurs if the T2EDGE2 DET ENA and T2EDGE2 INT ENA bits are set (T2CTL3.1 and T2CTL2.1).

Note:

All set and enabled interrupt flags must be cleared before exiting the T2 interrupt routine. If the flags are not reset, then the processor will enter the T2 interrupt routine again instead of continuing the mainstream program. If the flag bits are never reset then the program will lock up.

8.2.10 Power-Down Modes

This module supports the power-down modes which aid in reducing power consumption during periods of inactivity. In both the Halt and Standby modes, no clocks or external inputs are recognized.

The low-power modes are entered when an IDLE instruction is executed by the CPU if the POWERDOWN/IDLE bit (SCCR2.6) is set. During the low-power mode, the Timer 2 Module holds the pre-idle status of all storage elements. The module's external pins are held constant regardless of the pin function, i.e., 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 where it left off.

8.3 Timer 2 Control Registers

Peripheral File registers control the Timer 2 module operating mode selection, interrupt enable, status flags, and output configuration. These registers are shown in Table 8-2 The bits shown in shaded boxes in Table 8-2 are Privilege Mode bits, that is, they can only be written to in the Privilege Mode.

Note:

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

Read: LSB then MSB
Write: MSB then LSB

Table 8-2. Peripheral File Frame 6: Timer 2 Control Registers

PERIPHERAL FILE FRAME 6: TIMER 2 MODULE CONTROL REGISTERS

ADDR	PF	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	
1060h	060	BIT 15			T2 COUNTER MSB				BIT 8	T2CNTR
1061h	061	BIT 7			T2 COUNTER LSB				BIT 0	
1062h	062	BIT 15			T2 COMPARE 1 REGISTER MSB				BIT 8	T2C
1063h	063	BIT 7			T2 COMPARE 1 REGISTER LSB				BIT 0	
1064h	064	BIT 15			CAPTURE 1/COMPARE 2 REGISTER MSB				BIT 8	T2CC
1065h	065	BIT 7			CAPTURE 1/COMPARE 2 REGISTER LSB				BIT 0	
1066h	066	BIT 15			CAPTURE REGISTER 2 MSB				BIT 8	T2IC
1067h	067	BIT 7			CAPTURE REGISTER 2 LSB				BIT 0	
1068h 1069h	068 069	RESERVED								
106Ah	06A	---	---	---	T2 OVRFL INT ENA	T2 OVRFL INT FLAG	T2 INPUT SELECT 1	T2 INPUT SELECT 0	T2 SW RESET	T2CTL1

MODE: DUAL COMPARE REGISTERS

106Bh	06B	T2EDGE1 INT FLAG	T2C2 INT FLAG	T2C1 INT FLAG	---	---	T2EDGE1 INT ENA	T2C2 INT ENA	T2C1 INT ENA	T2CTL2
106Ch	06C	T2MODE = 0	T2C1 OUT ENA	T2C2 OUT ENA	T2C1 RST ENA	T2EDGE1 OUT ENA	T2EDGE1 POLARITY	T2EDGE1 RST ENA	T2EDGE1 DET ENA	T2CTL3

MODE: DUAL CAPTURE AND SINGLE COMPARE REGISTERS

106Bh	06B	T2EDGE1 INT FLAG	T2EGDE2 INT FLAG	T2C1 INT FLAG	---	---	T2EDGE1 INT ENA	T2EGDE2 INT ENA	T2C1 INT ENA	T2CTL2
106Ch	06C	T2MODE = 1	---	---	T2C1 RST ENA	T2EGDE2 POLARITY	T2EDGE1 POLARITY	T2EDGE2 DET ENA	T2EDGE1 DET ENA	T2CTL3

106Dh	06D	---	---	---	---	T2EVT DATA IN	T2EVT DATA OUT	T2EVT FUNCTION	T2EVT DATA DIR	T2PC1
106Eh	06E	T2IC2/ PWM DATA IN	T2IC2/ PWM DATA OUT	T2IC2/ PWM FUNCTION	T2IC2/ PWM DATA DIR	T2IC1/ CR DATA IN	T2IC1/ CR DATA OUT	T2IC1/ CR FUNCTION	T2IC1/ CR DATA DIR	T2PC2
106Fh	06F	T2 STEST	T2 PRIORITY	---	---	---	---	---	---	T2PRI

8.3.1 Timer 2 Control Register 1

The T2CTL1 register controls the clock input selection, counter overflow interrupts, and counter software reset.

Timer 2 Control Register 1 (T2CTL1)
[Memory Address - 106Ah]

Bit # -	7	6	5	4	3	2	1	0
P06A	---	---	---	T2 OVRFL INT ENA	T2 OVRFL INT FLAG	T2 INPUT SELECT 1	T2 INPUT SELECT 0	T2 SW RESET
				RW-0	RC-0	RW-0	RW-0	S-0

R=Read, S=Set only, W=Write, C=Clear only, -n= Value after RESET

Bit 0 - **T2 SW RESET.** Timer 2 Software Reset.
When a one is written to this bit, the counter will reset to 0000h on the next system clock cycle, however, this bit is always read as a zero.

Bits 1,2 - **T2 INPUT SELECT 0-1.** Timer 2 Input Select 0,1.
These two bits select one of four clock sources as an input to the counter. The four options are:
- system clock with no prescale,
- system clock when external input is high (pulse accumulation),
- external source synchronized with system clock (event input).
- no clock,

The combinations are shown below.

Bit 2	Bit 1	Counter Clock Source
0	0	system clock
0	1	pulse accumulation
1	0	event input
1	1	no clock input

Bit 3 - **T2 OVRFL INT FLAG.** Timer 2 Overflow Interrupt Flag.
This bit is the Timer 2 Counter Overflow Bit. It is cleared by writing a zero to this bit or during RESET.
0 = Overflow interrupt inactive.
1 = Overflow interrupt pending.

Bit 4 - **T2 OVRFL INT ENA.** Timer 2 Overflow Interrupt Enable.
This bit controls the Timer 2 overflow interrupting capability.
0 = Disable Interrupt.
1 = Enable Interrupt from overflow.

Bits 5,6,7 - Reserved. Read data is indeterminate.

8.3.2 Timer 2 Control Register 2

The T2CTL2 register contains interrupt flags and controls the capability of the module to issue interrupts.

Timer 2 Control Register 2 (T2CTL2)
[Memory Address - 106Bh]

Mode: Dual Compare

Bit # -	7	6	5	4	3	2	1	0
P06B	T2EDGE1 INT FLG	T2C2 INT FLG	T2C1 INT FLG	---	---	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 FLG	T2EDGE2 INT FLG	T2C1 INT FLG	---	---	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 after RESET

Bit 0 - **T2C1 INT ENA.** Timer 2 Compare 1 Interrupt Enable.
This bit controls the interrupting capability of the Compare 1 register.
0 = Disable interrupt.
1 = Enable 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 = Disable interrupt.
1 = Enable 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 = Disable interrupt.
1 = Enable 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 = Disable interrupt.
1 = Enable interrupt.

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

Bit 5 - **T2C1 INT FLAG.** Timer 2 Output Compare 1 Interrupt Flag.
This bit is set when the output Compare Register first matches the counter value. It is cleared by writing a zero to this bit, or during RESET.
0 = Interrupt inactive.
1 = Interrupt pending from Compare 1.

Bit 6 - *Dual Compare Mode:*
T2C2 INT FLAG. Timer 2 Output Compare 2 Interrupt Flag.
This bit is set when the Capture/Compare Register first matches the counter value. It is cleared by writing a zero to this register bit, or during RESET.
0 = Interrupt inactive.
1 = Interrupt pending from Compare 2.

Dual Capture Mode:
T2EDGE2 INT FLAG. Timer 2 Edge 2 Interrupt Flag.
This bit is set when the appropriate edge is detected on T2IC2/PWM and indicates that the Capture Register was loaded. It is cleared by writing a zero to this register bit, or during RESET.
0 = Interrupt inactive.
1 = Interrupt pending from Edge 2 Detect.

Bit 7 - **T2EDGE1 INT FLAG.**Timer 2 External Edge 1 Interrupt Flag.
This bit is set when the appropriate edge is detected on the T2IC1/CR pin. It is cleared by writing a zero to this register bit, or during RESET.
0 = Interrupt inactive.
1 = Interrupt pending from Edge 1 Detect circuitry.

8.3.3 Timer 2 Control Register 3

The T2CTL3 register controls the Timer 2 module mode of operation, outputs, active transition polarity, and counter reset.

Timer 2 Control Register 3 (T2CTL3)
[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 Compare

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 after RESET

Bit 0 - **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 or during RESET.
0 = Edge 1 detect disabled.
1 = Edge 1 detect enabled.

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 = Disable external reset of the counter.
1 = Enable 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 is cleared after the selected transition is detected or during RESET.
0 = Edge detect disabled.
1 = Edge detect enabled.

Bit 2 - **T2EDGE1 POLARITY.** Timer 2 Edge 1 Polarity Select. .
This bit controls which level transition on the T2IC1/CR pin is active.
0 = Trigger on high-to-low transition.
1 = Trigger on low-to-high transition.

Bit 3 - *Dual Compare Mode:*
T2EDGE1 OUT ENA. Timer 2 Edge 1 Detect Output Enable.
This bit controls whether or not the pulse indicating an external edge detect toggles the module's output pin.
0 = Disable pulse to toggle output.
1 = Enable pulse to toggle output.

Dual Capture Mode:
T2EDGE2 POLARITY. Timer 2 Edge 2 Polarity Select.
This bit controls which level transition on the T2IC2/PWM 1 pin, will trigger a counter reset, depending upon the counter mode selected.
0 = Trigger on high-to-low transition.
1 = Trigger on low-to-high transition.

Bit 4 - **T2C1 RST ENA.** Timer 2 Output Compare 1 Reset Enable.
This bit controls whether or not the compare equal pulse from the Compare Register resets the counter on the next counter increment.
0 = Disable reset upon compare equal.
1 = Enable reset upon compare equal.

Bit 5 - *Dual Compare Mode:*
T2C2 OUT ENA. Timer 2 Output Compare 2 Enable.
This bit controls whether or not the output compare equal pulse from the Capture/Compare Register toggles the T2IC2/PWM output pin.
0 = Disable pulse to toggle output.
1 = Enable pulse to toggle output.

Dual Capture Mode:
Reserved. Read data is indeterminate.

Bit 6 - *Dual Compare Mode:*
T2C1 OUT ENA. Timer 2 Output Compare 1 Enable.
This bit controls whether or not the compare equal pulse from the Compare Register toggles T2IC2/PWM pin.
0 = Disable pulse from toggling output.
1 = Enable pulse to toggle output.

Dual Capture Mode:
Reserved. Read data is indeterminate.

Bit 7 - **T2 MODE.** Timer 2 Mode Select.
This bit selects the operating mode for the counter.
0 = Dual Compare mode.
1 = Dual Capture mode.

8.3.4 Timer 2 Port Control Registers

The Port Control Registers (T2PC1, T2PC2) control the functions of the I/O pins. Each module pin is controlled by a nibble in one of the PCRs.

8.3.4.1 Timer 2 Port Control Register 1

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
					R-0	RW-0	RW-0	RW-0

R=Read, W=Write, -n= Value after 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 = T2EVT2 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 port.
1 = T2EVT is the event input pin.

Bit 2 - **T2EVT DATA OUT.** Timer 2 Event Pin Data Out.
This bit contains the data to be output on the T2EVT pin if the following conditions are met:
a. T2EVT DATA DIR = 1
b. T2EVT FUNCTION = 0

Bit 3 - **T2EVT DATA IN.** Timer 2 Event Pin Data In.
This bit contains the data to be input from the T2EVT pin. A write to this bit has no effect.

Bits 4,5,6,7 - Reserved. Read data is indeterminate.

8.3.4.2 Timer 2 Port Control Register 2

The T2PC2 register assigns the I/O functions of the T2IC1/CR and T2IC2/PWM pins as either general-purpose digital I/O pins or the input-capture/counter-reset and PWM output pins, respectively.

Timer 2 Port Control Register 2 (T2PC2)
[Memory Address - 106Eh]

Bit # -	7	6	5	4	3	2	1	0
P06E	T2IC2/ PWM DATA IN	T2IC2/ PWM DATA OUT	T2IC2/ PWM FUNCTION	T2IC2/ PWM DATA DIR	T2IC1/ CR DATA IN	T2IC1/ CR DATA OUT	T2IC1/ CR FUNCTION	T2IC1/ CR DATA DIR
	R-0	RW-0	RW-0	RW-0	R-0	RW-0	RW-0	RW-0

R=Read, W=Write, -n= Value after RESET

Bit 0 - **T2IC1/CR DATA DIR.** Timer 2 IC1/CR Data Direction.
This bit determines the direction of data on the T2IC1/CR pin if the T2IC1/CR FUNCTION bit = 0.

0 = T2IC1/CR is an input.
1 = T2IC1/CR is an output.

Bit 1 - **T2IC1/CR FUNCTION.** Timer 2 IC1/CR Function Select.
This bit determines the function of the T2IC1/CR pin.

0 = T2IC1/CR is a general-purpose digital I/O port.
1 = T2IC1/CR is the input capture/counter reset pin.

Bit 2 - **T2IC1/CR DATA OUT.** Timer 2 IC1/CR Data Out.
This bit contains the data output on the T2IC1/CR pin if the following conditions are true:
a. T2IC1/CR DATA DIR = 1
b. T2IC1/CR FUNCTION = 0

Bit 3 - **T2IC1/CR DATA IN.** Timer 2 IC1/CR Data In.
This bit contains the data input on the T2IC1/CR pin. A write to this bit has no effect.

Bit 4 - **T2IC2/PWM DATA DIR.** Timer 2 IC2/PWM Data Direction.
This bit determines the direction of data on the T2IC2/PWM pin if the T2IC2/PWM FUNCTION bit = 0.

0 = T2IC1/PWM is an input.
1 = T2IC2/PWM is an output.

Bit 5 - **T2IC2/PWM FUNCTION.** Timer 2 IC2/PWM Function Select.
This bit determines the function of the T2IC2/PWM pin.

0 = T2IC2/PWM is a general-purpose digital I/O port.
1 = T2IC2/PWM is the input capture/PWM output pin.

Bit 6 - **T2IC2/PWM DATA OUT.** Timer 2 IC2/PWM Data Out.
This bit contains the data output on the T2IC2/PWM pin if the following conditions are true:
a. T2IC2/PWM DATA DIR = 1
b. T2IC2/PWM FUNCTION = 0

Bit 7 - **T2IC2/PWM DATA IN.** Timer 2 IC2/PWM Data In.
This bit contains the data input on the T2IC2/PWM pin. A write to this bit has no effect.

8.3.5 Timer 2 Interrupt Priority Control Register

The T2PRI register assigns the priority level of interrupts generated by the Timer 2 module.

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=Privileged Write, -n= Value after 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.**
This bit must be cleared (0) to ensure proper operation.

9. Serial Communications Interface (SCI) Port

This section discusses the architecture and programming of the Serial Communications Interface module on TMS370C050 and TMS370C850 devices.

This section covers the following topics:

9.1 SCI Overview

The programmable Serial Communications Interface (SCI) allows digital communications between the TMS370 device and other asynchronous peripherals using the standard NRZ (Non Return to Zero) format. Both the SCI receiver and transmitter are double buffered and have their own separate enable and interrupt bits. Thus, they may be operated independently or simultaneously in the Full Duplex mode.

To ensure data integrity, the SCI checks received data for Break detection, Parity, Overrun, and Framing errors. The speed of operation, or Baud rate, is programmable to over 65,000 different speeds through a 16-bit baud-select register.

9.1.1 Physical Description

The major elements of the full-duplex SCI is shown in Figure A-9 and includes:

1) a transmitter (TX),

   a) TXBUF - Transmitter Buffer Register, contains data written by the CPU, to be transmitted.
   b) TXSHF - Transmitter Shift Register, loaded from TXBUF, shifts data onto SCITXD pin one bit at a time.

2) a receiver (RX),

   a) RXSHF - Receiver Shift Register, shifts data in from SCIRXD pin one bit at a time.
   b) RXBUF - Receiver Buffer Register, contains data to be read by the CPU, received from remote processor, loaded from RXSHF.

3) a programmable baud rate generator, and

4) memory mapped control and status registers.

The SCI receiver and transmitter can operate independently and simultaneously. A third port line (SCICLK) is available for the optional synchronizing clock line in the Isosynchronous mode.

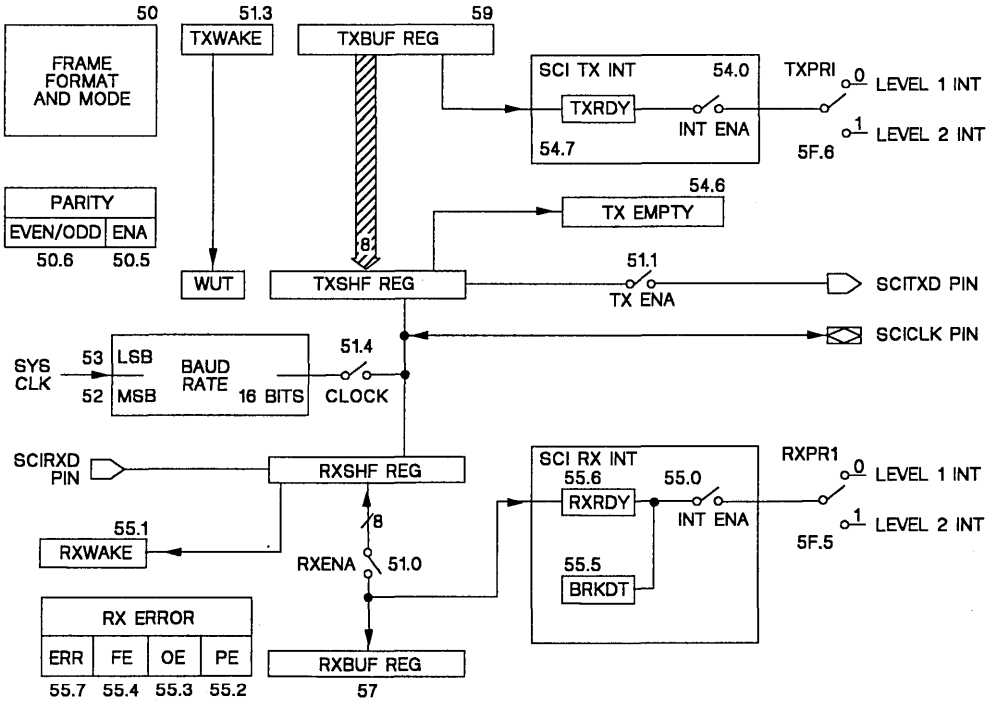


Figure 9-1. SCI Block Diagram

9

9.1.2 SCI Features

Features of the Serial Communications Interface (SCI) include the following:

- Two Communications Formats
  - Asynchronous
  - Isosynchronous
- Programmable Baud Rates
  - Asynchronous:
    - ▲ Range at 20 MHz - 3 Bps to 156K Bps
    - ▲ Number of Rates - 64K
  - Isosynchronous:
    - ▲ Range at 20 MHz - 39 Bps to 2.5M Bps
    - ▲ Number of Baud 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:
  - Parity Error
  - Overrun Error
  - Framing Error
  - Break Detect
- Two Wake-Up Multiprocessor Modes which may be used with either Communications Format.
  - Idle Line Wake-Up
  - Address Bit Wake-Up
- Full Duplex Operation
- Separate Transmitter and Receiver Interrupts For Polled or Interrupt Driven Operation
- Double Buffered Receive and Transmit Functions
- Separate Enable Bits for the Transmitter and Receiver.
- NRZ (Non-Return-To-Zero) Format.

9.1.3 SCI Formats and Operation Modes

The SCI may use one of two communication formats, Asynchronous or Isosynchronous. These formats may be programmed to contain:

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

The SCI provides the following Universal Asynchronous Receiver/Transmitter (UART) communications formats for interfacing with many popular peripherals:

- Asynchronous Mode (discussed in Section 9.2.4.1, page 9-9) requires two lines to interface with many standard devices such as terminals and printers using RS-232-C formats.
- Isosynchronous Mode (discussed in Section 9.2.4.2, page 9-10) permits high transmission rates and requires a synchronizing clock signal between the receiver and transmitter.

The SCI also has two multiprocessor protocols, the **Idle Line** Multiprocessor Mode (see Section 9.2.5.1) and the **Address Bit** Multiprocessor Mode (see Section 9.2.5.2). These protocols allow efficient data transfer between multiple processors, and may be used with either the Isosynchronous or standard Asynchronous formats.

The SCI transmits and receives serial data, one bit at a time at a programmable baud rate. If the TMS370 operates at 20 MHz, the Baud rate for the Asynchronous mode would range from 3 bits-per-second to 156 kilobits-per-second, and for the Isosynchronous mode would range from 39 bits-per-second to 2.5 megabits-per-second.

9.1.4 SCI Control Registers

The SCI Control registers are located at addresses 1050h to 105Fh. The function of each location is shown in Table 9-1.

Table 9-1. SCI Memory Map

Peripheral File Location	Symbol	Name
P050	SCICCR	SCI Communication Control Register
P051	SCICTL	SCI Control Register
P052	BAUD MSB	Baud Rate Select MSB
P053	BAUD LSB	Baud Rate Select LSB
P054	TXCTL	Transmitter Interrupt Control and Status Register
P055	RXCTL	Receiver Interrupt Control and Status Register
P056		Reserved
P057	RXBUF	Receiver Data Buffer
P058		Reserved
P059	TXBUF	Transmit Data Buffer
P05A P05B P05C		Reserved
P05D	SCIPC1	Port Control 1
P05E	SCIPC2	Port Control 2
P05F	SCIPRI	Priority Control

9.2 SCI Operation

The functions of the SCI are software configurable. A set of control words sent to the SCI initializes the desired communications format. These control words determine the:

1) operating mode and protocol,
2) baud rate,
3) character length,
4) even/odd parity or parity off,
5) number of stop bits, and
6) interrupt priorities and enables.

9.2.1 SCI Programmable Data Format

SCI data, both receive and transmit, is in NRZ (Non-Return to Zero) format. The data format consists of one Start bit, 1 to 8 data bits, an optional even/odd parity bit, and either 1 or 2 Stop bits, as illustrated in Figure 9-3

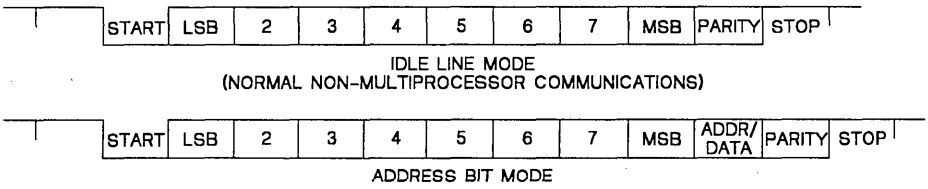


Figure 9-3. SCI Data Frame Formats

9.2.2 SCI Port Interrupts

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

The receiver interrupt is asserted when the RXRDY (RXCTL.6) or BRKDT (TXCTL.5) flags are set, assuming the SCI RX INT ENA bit (RXCTL.0) is set. The transmitter interrupt is asserted when the TXRDY flag (TXCTL.7) is set, assuming the SCI TX INT ENA bit (TXCTL.0) is set.

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

An SCI TX interrupt is asserted whenever TXBUF is transferred to TXSHF. This interrupt indicates that the CPU can write to the TXBUF.

An SCI RX interrupt is asserted whenever the SCI receives a complete frame (RXSHF transfers to RXBUF) or when a break detect condition occurs (SCIRXD is low for 10 bit periods following a stop bit).

9.2.3 SCI Clock Sources

The SCI port can be driven by an internal or external baud rate 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, then the SCICLK pin functions as the high impedance Serial Clock input pin.

If an internal clock source is selected (CLOCK = 1), the SCICLK pin may be used as a general purpose I/O pin or as the Serial Clock output pin. If the Serial Clock output is selected, a 50 percent duty cycle clock signal is output on the SCICLK pin, which becomes a Serial Clock output pin.

The internally generated serial clock is determined by the TMS370 CLKIN frequency and the Baud Rate Select Registers. The SCI uses the 16-bit value of the Baud Rate Select Registers to select one of 64K different serial clock rates for the communication modes in the following manner:

Asynchronous Baud Rate = CLKIN / [(BAUD RATE REG + 1) * 128]

and

Isosynchronous Baud Rate = CLKIN / [(BAUD RATE REG + 1) * 8]

and

SCICLK frequency = CLKIN / [(BAUD RATE REG + 1) * 8]

where,
BAUD RATE REG = The 16-bit value in the Baud Rate Select Registers.

> **Note:**
>
> When an external serial clock signal is used, the maximum SCICLK frequency is CLKIN/16.

The current logic level on the SCICLK pin can be determined by reading the SCICLK DATA IN bit (SCIPC1.3).

The SCI receives data on rising clock edges and transmits data on falling clock edges.

9.2.4 SCI 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.

9.2.4.1 Asynchronous Communications Mode

The SCI Asynchronous communication mode uses either single line (one way) or double 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 a negative edge followed by three samples, two of which MUST be zero. If two of the three samples are not zero, then the receiver continues to search for a start bit. These samples occur on the seventh, eighth, and ninth SCICLK periods after the negative edge. This sequence provides false start rejection and also locates the center of bits in the frame where the bits are read on a majority (two out of three) basis. Figure 9-4 illustrates the asynchronous communication format, with a start bit showing how edges are found and where a majority vote is taken.

Since the receiver synchronizes itself to frames, the external transmitting and receiving devices do not have to use synchronized serial clocks; it may be generated locally. If the CLOCK (SCICTL.4) and SCICLK FUNCTION (SCIPC1.1) bits are set, the serial clock function is output continuously on the SCICLK pin.

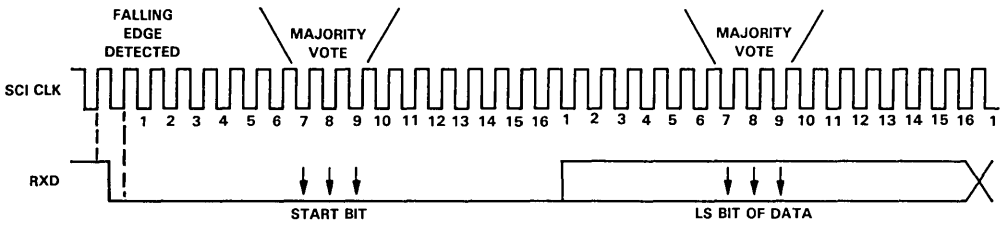


Figure 9-4. Asynchronous Communication Format

9.2.4.2 SCI Isosynchronous Communications Mode

The SCI Isosynchronous communication mode uses either two line (one way) or three line (two way) communications. The extra line (Serial Clock) is required for data synchronization. In the Isosynchronous mode, each bit of data requires only one serial clock pulse for transmission or reception. Thus, the data bit period equals the SCICLK period, and data bits are read on a single sample basis.

Since the receiver does not synchronize itself to data bits, the transmitter and receiver must be supplied with a common serial clock. If the internal serial clock is used it must be output continuously on the SCICLK pin. The arrival of a valid start bit, which consists of a low on the RXD line at the time of a rising SCICLK edge, initiates receiver operation.

Figure 9-5 illustrates the Isosynchronous communication format. A complete frame consists of a start bit, one to eight data bits, an optional even/odd parity bit, and one or two stop bits.

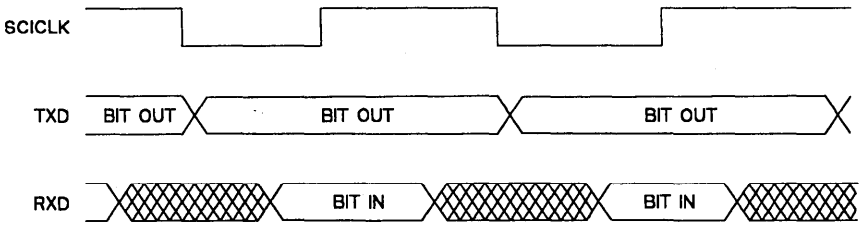


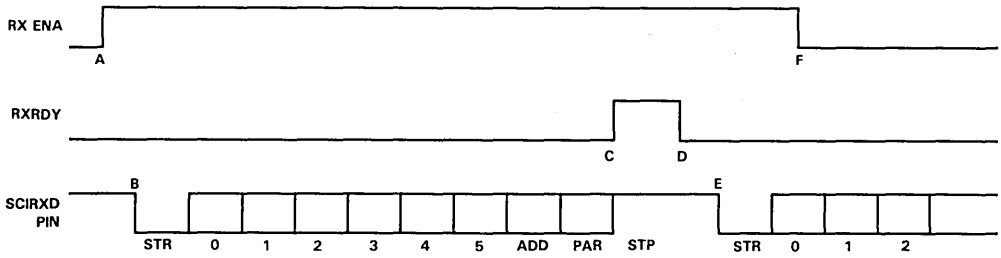
Figure 9-5. Isosynchronous Communication Format

9.2.4.3 Receiver Signals in Communications Modes

Figure 9-6 illustrates an example of receiver signal timing assuming the following:

1) Address bit wake-up mode
2) 6 bits per character

Lettered notes following the diagram are keyed to the letter labels in the diagram.



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. 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.

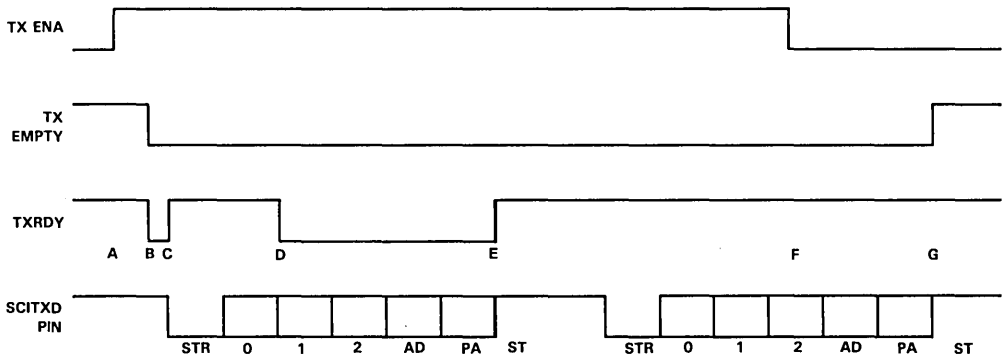
Figure 9-6. SCI RX Signals in Communication Modes

9.2.4.4 Transmitter Signals in Communications Modes

Figure 9-6 illustrates an example of transmitter signal timing assuming the following:

1) Address bit wake-up mode (address bit would not appear in Idle line mode)
2) 3 bits per character

Lettered notes following the diagram are keyed to the letter labels in the diagram.



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, 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.

Figure 9-7. SCI TX Signals in Communications Modes

9.2.5 SCI Multiprocessor Communications

The Multiprocessor Communication format allows one processor to efficiently send blocks of data to other processors on the same serial link. On one serial line there should be only one talker at a time. The first byte of a block of information contains an address byte which is read by all listeners. Only correctly addressed listeners can be interrupted by the following data bytes. The listeners not addressed remain uninterrupted until the next address byte.

The two different multiprocessor modes, supported by TMS370 devices, differ in how the processor recognizes an address byte. The **Idle Line** mode leaves a quiet space before the address byte. The **Address Bit** mode adds an extra bit into every byte to distinguish addresses from data.

The multiprocessor mode is software selectable via the ADDRESS/IDLE WUP bit (SCICCR.3). Both formats use the TXWAKE and SLEEP flags to control the SCITX and SCIRX features of these modes.

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

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

In both multiprocessor modes the sequence is:

1) The SCI port wakes up (requests an interrupt) at the start of a block and reads the first frame which contains the destination address.
2) A software routine is entered through the interrupt and checks the incoming byte against its device address byte stored in memory.
3) If the block is addressed to the microcomputer, the CPU clears the SLEEP bit and reads the rest of the block; if not, the software routine exits with the SLEEP bit still set and does not receive SCI interrupts until the next block start.

The Idle Line multiprocessor mode does not contain the extra address/data bit, and is more efficient than the Address Bit mode in handling blocks containing more than 10 bytes of data.

The Address Bit mode is more efficient in handling many small blocks of data because it does not have to wait between blocks of data as does the Idle Line mode. However, at high transmit speeds, the program may not be quick enough to avoid a 10-bit idle in the transmission stream.

9.2.5.1 Idle Line Multiprocessor Mode

In the Idle Line multiprocessor protocol, 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 9-8.

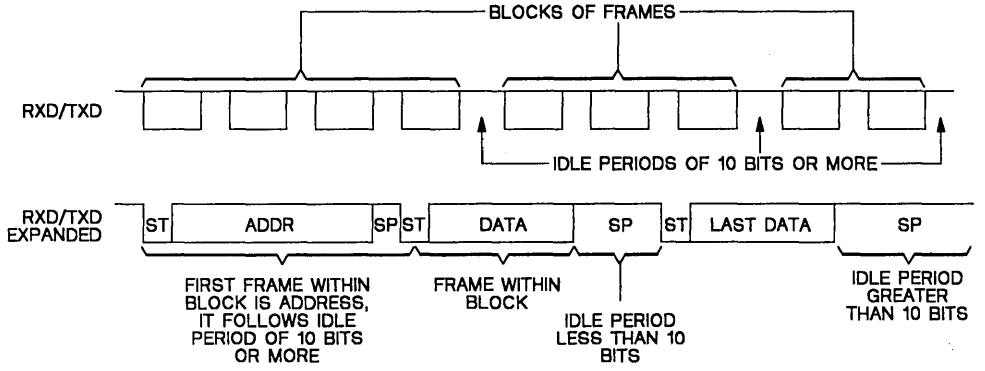


Figure 9-8. Idle Line Multiprocessor Communication Format

The SCI wakes up after the block start signal. The processor now recognizes the next SCI interrupt. The service routine then receives the address sent by a remote transmitter and compares this address to its own. If the CPU is addressed, the service routine clears the SLEEP bit, and receives the rest of the data block. If the CPU is not addressed, the SLEEP bit is left set. This lets the CPU continue to execute its main program without being interrupted by the SCI port.

There are two ways to send a block start signal.

1) 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 address frame of the new block.
2) In the second method, the SCI port uses the TXWAKE bit 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 (WUT) flag. WUT is an internal flag, double buffered with TXWAKE. When TXSHF is loaded from TXBUF, WUT is loaded from TXWAKE, and TXWAKE is reset to 0. This arrangement is shown in Figure 9-9.

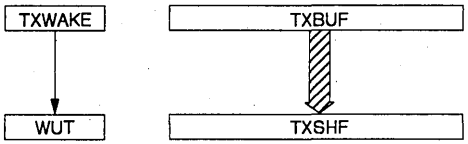


Figure 9-9. Double-Buffered WUT and TXSHF

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

1) Write a 1 to the TXWAKE bit.

2) Write a data word (don't care) to TXBUF. (The first data word written is suppressed while the block start signal is sent out, and ignored after that.)

   When TXSHF is free again, TXBUF's contents are shifted to TXSHF, the TXWAKE value is shifted to WUT, and then TXWAKE is cleared.

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

3) Write an address value to the TXBUF.

Writing the first "don't care" data word to the TXBUF is necessary so the TXWAKE bit value can be shifted to WUT. After the "don't-care" data word is shifted to the TXSHF, the TXBUF (and TXWAKE if necessary) may be written to again, since WUT and TXSHF are both double-buffered.

The receiver operates regardless of the SLEEP bit. The receiver does not set RXRDY, RXINT, or the error status bits until an address frame is detected.

9.2.5.2 Address Bit Multiprocessor Mode

In the Address Bit protocol, the frame has an extra bit called an address bit immediately after the last data bit. The first frame in the block has the address bit set to 1, and all other frames have the address bit set to 0. The idle period timing is irrelevant.

The TXWAKE bit sets the address bit. In SCITX, when the TXBUF and TXWAKE are loaded into TXSHF and WUT, TXWAKE is reset to 0 and WUT is the value of the address bit of the current frame. Thus, to send an address, set the TXWAKE bit to a 1, and write the appropriate address value to the TXBUF. When this address value is transferred to TXSHF and shifted out, its address bit is sent as a 1, which flags the other processors on the serial link to read the address. Since TXSHF and WUT are both double-buffered, TXBUF and TXWAKE may be written to immediately after TXSHF and WUT are loaded. To transmit non-address frames in the block, the TXWAKE bit is left at 0.

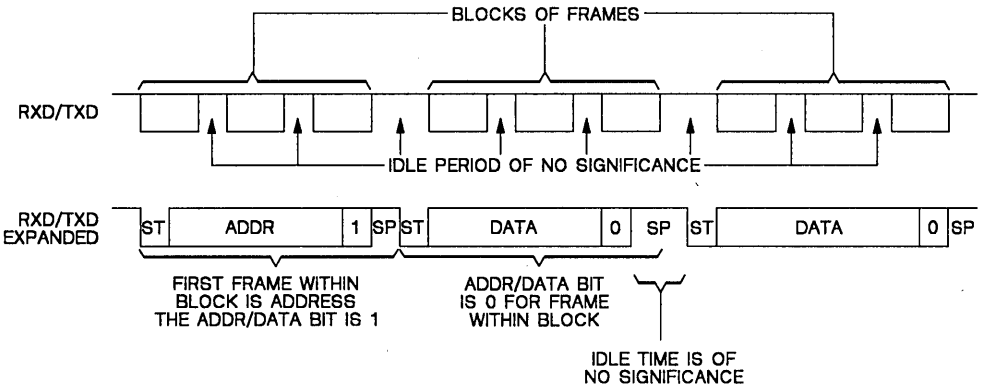


Figure 9-10. Address Bit Multiprocessor Communication Format

9.2.6 SCI Initialization Examples

This section contains four examples that initialize the serial port. In each case the data is moved to and from the buffers in the interrupt routines.

- The first example shows a typical RS-232 application that connects to a terminal.
- The second example illustrates the Address Bit mode in a multiprocessor application.

In all examples, assume the register mnemonics have been equated (EQU) with the corresponding Peripheral-File location.

9.2.6.1 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 rate) - 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 baud rate for 9600 (MSB)
        MOV  #Lo B9600,BAUDLSB  ;Set baud rate for 9600 (LSB)
        MOV  #077h,SCICCR       ;1 stop bit, even parity,
                                ;and 8 data bits/char
        MOV  #033h,SCICTL       ;Enable Rx, Tx, clock is internal
        MOV  #001h,TXCTL        ;Enable TX interrupt
        MOV  #001h,RXCTL        ;Enable RX interrupt
        EINT                    ;Let the interrupts begin
        MOV  #00,TXBUF          ;Start transmitter by sending null
                                ; character
```

9.2.6.2 RS-232-C Multiprocessor Mode Example

This example initializes the transmitter and receiver to accept data at 9600 baud with a format of 8 data bits, 1 stop bit, and even parity. It uses the address bit wake-up mode to implement the multiprocessor protocol.

```
B9600    .EQU 15                  ;Value for counter for 9600 baud
                                  ;value=(CLKIN/128/baud rate) - 1 =
                                  ;(20 MHz/128/9600) - 1 = 15.27 ~ 15
                                  ;1.8 percent error
         MOV  #000h,SCIPRI        ;Set TX and RX to high priority
         MOV  #005h,SCIPC1        ;Set SCLK for general purpose output
         MOV  #022h,SCIPC2        ;Set pins for RXD and TXD functions
         MOV  #Hi B9600,BAUDMSB   ;Set baud rate for 9600 (MSB)
         MOV  #Lo B9600,BAUDLSB   ;Set baud rate for 9600 (LSB)
         MOV  #077h,SCICCR        ;1 stop bit, even parity,
                                  ;and 8 data bits/char
         MOV  #037h,SCICTL        ;Enable Rx, Tx; RX to sleep,
                                  ;clock is internal
         MOV  #001h,TXCTL         ;Enable TX interrupt
         MOV  #001h,RXCTL         ;Enable RX interrupt
         EINT                     ;Let the interrupts begin
                                  ;
                                  ;MAIN ROUTINES
                                  ;
SENDADD  OR   #8,SCICTL           ;Main line routine; set TXWAKE
                                  ;wake bit
         MOV  ADDR,TXBUF          ;Transmit address stored in ADDR
         RTS
                                  ;
                                  ;INTERRUPT ROUTINES
                                  ;
SENDATA  PUSH A                   ;Address has already been sent by
                                  ;the SENDADD
         MOV  OUTDATA,TXBUF       ;Output character that is
                                  ;stored in DATA
         .                        ;
         .                        ;Other transmitter code
         .                        ;
         POP  A                   ;Restore and exit
         RTI
         .                        ;
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  #0F7h,SCICTL        ;If my address get out of sleep mode
         JMP  RXEXIT              ;Exit and wait for data
                                  ;
ISDATA   MOV  RXBUF,INDATA        ;Put incoming data in register
         .                        ;
         .                        ;Other receiver code
         .                        ;
RXEXIT   POP  A                   ;Restore and exit
         RTI
```

9.3 SCI Control Registers

The SCI is controlled and accessed through registers in the Peripheral File. These registers are listed in Table 9-1 and described in the following sections. The bits shown in shaded boxes in Table 9-1 are Privilege Mode bits, that is, they can only be written to in the Privilege Mode.

Table 9-1. SCI Control Registers

PERIPHERAL FILE FRAME 5: SERIAL COMMUNICATION INTERFACE (SCI) CONTROL REGISTERS

ADDR	PF	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	
1050h	050	STOP BITS	EVEN/ODD PARITY	PARITY ENABLE	ASYNC/ ISOSYNC	ADDRESS/ IDLE WUP	SCI CHAR2	SCI CHAR1	SCI CHAR0	SCICCR
1051h	051	---	---	SCI SW RESET	CLOCK	TXWAKE	SLEEP	TXENA	RXENA	SCICTL
1052h	052	BIT 15	BAUD RATE SELECT REGISTER MSB						BIT 8	BAUD MSB
1053h	053	BIT 7	BAUD RATE SELECT REGISTER LSB						BIT 0	BAUD LSB
1054h	054	TXRDY	TX EMPTY	---	---	---	---	---	SCI TX INT ENA	TXCTL
1055h	055	RX ERROR	RXRDY	BRKDT	FE	OE	PE	RX WAKE	SCI RX INT ENA	RXCTL
1056h	056	RESERVED								
1057h	057	RECEIVE DATA BUFFER REGISTER								RXBUF
1058h	058	RESERVED								
1059h	059	TRANSMIT DATA BUFFER REGISTER								TXBUF
105Ah 105Bh 105Ch	05A 05B 05C	RESERVED								
105Dh	05D	---	---	---	---	SCICLK DATA IN	SCICLK DATA OUT	SCICLK FUNCTION	SCICLK DATA DIR	SCIPC1
105Eh	05E	SCI TXD DATA IN	SCI TXD DATA OUT	SCI TXD FUNCTION	SCI TXD DATA DIR	SCI RXD DATA IN	SCI RXD DATA OUT	SCI RXD FUNCTION	SCI RXD DATA DIR	SCIPC2
105Fh	05F	SCI STEST	SCI TX PRIORITY	SCI RX PRIORITY	SCI ESPEN	---	---	---	---	SCIPRI

9.3.1 Communication Control Register (SCICCR)

The SCICC Register defines the character format, protocol, and communications mode 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 after RESET

Bits 0-2 **SCI CHAR0–2**. SCI Character Length Control Bits 0–2.
These bits select the SCI character 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 9-2. Transmitter 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 one of the multiprocessor protocols.

0 = Idle Line Mode protocol selected.
1 = Address Bit Mode protocol selected.

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 communication 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 (described in Section 9.2.4.2). In this mode, the bit period is equal to the SCICLK period; bits are read on a single sample basis.
1 = Selects Asynchronous mode (described in Section 9.2.4.1). In this mode the bit period is 16 times the SCICLK period; bits are read on a two out of three majority basis.

Bit 5 - **PARITY ENABLE**. SCI Parity Enable.
This bit enables or disables the parity function. When parity is enabled during the Address Bit multiprocessor mode, the address bit is included in the parity calculation.

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

Bit 6 - **EVEN/ODD PARITY**. SCI Parity Odd/Even.
If the PARITY ENABLE (SCICCR.5) is set, then this bit selects odd or even parity (odd or even number of 1 bits in both transmitted and received characters).

0 = Sets odd parity.
1 = Sets even parity.

Bit 7 - **STOP BITS**. SCI Number of Stop Bits.
This bit determines the number of stop bits transmitted. The receiver checks for one stop bit only.

0 = One stop bit.
1 = Two stop bits.

9.3.2 Control Register (SCICTL)

The SCICTL register controls the RX/TX enable, TXWAKE and SLEEP functions, internal clock enable, and the SCI software Reset.

SCI Control Register (SCICTL)
[Memory - 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 after 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. Thus, if RXENA is set during the reception of a character, the complete character is transferred into RXBUF.

0 = SCI Receiver disabled.
1 = SCI Receiver enabled.

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 = SCI Transmitter disabled.
1 = SCI Transmitter enable.

Bit 2 - **SLEEP**. SCI Sleep.
This bit controls the receive features of the multiprocessor communication modes. This bit must be cleared by the user to bring the SCI out of Sleep mode.

0 = Sleep mode disabled.
1 = Sleep mode enabled.

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 rate generator. For Isosynchronous transactions, one bit is transmitted or received per SCICLK period. For Asynchronous transactions, one bit is transmitted or received per 16 SCICLK periods. The maximum frequency for the externally sourced SCICLK is CLKIN/16. Setting this bit selects an internal SCICLK, derived from the baud rate generator. This signal can be output on the SCICLK line.

0 = External SCICLK.
1 = Internal SCICLK.

Bit 5 - **SCI SW RESET**. SCI Software Reset (Active Low).
Writing a 0 to this bit initializes the SCI state machines and operating flags to the reset condition. The CLOCK bit retains its state prior to the assertion of SCI SW RESET. If SCICLK is configured as an output, then the SCICLK resets (low level). All effected logic is held in the reset state until a 1 is written to the SCI SW RESET bit. Thus, after a System RESET, the SCI must be re-enabled by writing a 1 to this bit.

Note:

The SCI SW RESET bit must be cleared before the SCI configuration registers can be set up or altered. All configuration registers should be set up by the application program prior to setting SCI SW RESET.

Bits 6,7 -Reserved. Read data is indeterminate.

9.3.3 Baud Select Registers (BAUD MSB and BAUD LSB)

The BAUD MSB and BAUD LSB registers store the data required to generate the baud rate. The SCI uses the combined 16-bit value, BAUD RATE REG, of the baud select registers to set the SCI clock frequency as follows:

SCICLK frequency = CLKIN / [(BAUD RATE REG + 1) * 8]

where,

BAUD RATE REG = The 16 bit value in the Baud Rate Select Registers.

For example, if the CLKIN frequency is 20 MHz, then 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 Baud Rates are calculated as follows:

Asynchronous Baud Rate = CLKIN / [(BAUD RATE REG + 1)* 128]

Isosynchronous Baud Rate = CLKIN / [(BAUD RATE REG + 1)* 8]

Baud Rate Select MSB 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 Rate Select LSB 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 after RESET

9.3.4 Transmitter Interrupt Control and Status Register (TXCTL)

The TXCTL Register contains the Transmitter Interrupt Enable, the Transmitter Ready flag, and the Transmitter Empty flag. The status flags are updated each time a complete character is transmitted. A summary of the register functions and bit assignments is shown below.

Transmitter Interrupt Control and Status Register (TXCTL)
[Memory address - 1054h]

Bit # -	7	6	5	4	3	2	1	0
P054	TXRDY	TX EMPTY	---	---	---	---	---	SCI TX INT ENA
	R-1	R-1						RW-0

R=Read, W=Write, -n= Value after RESET

Bit 0 - **SCI TX INT ENA**. SCI Transmitter Ready Interrupt Enable.
This bit controls the ability of the TXRDY bit to request an interrupt, but does not prevent the TXRDY bit from being set. The SCI TX INT ENA bit is set to 0 by an SCI SW RESET or a system RESET.

0 = SCI TXRDY interrupt disabled.
1 = SCI TXRDY interrupt enabled.

Bits 1-5 - Reserved. Read data is indeterminate.

Bit 6 - **TX EMPTY**. SCI Transmitter Empty.
This bit indicates the status of the transmitter-shift register and the TXBUF register. TX EMPTY is set to 1 by a SCI SW RESET or a System RESET.

0 = the CPU has written data to the TXBUF register, the data has not been completely transmitted.
1 = TXBUF and TXSHF register empty.

Bit 7 - **TXRDY**. SCI Transmitter Ready.
The TXRDY bit is set by the transmitter to indicate that TXBUF is ready to receive another character. The bit is automatically cleared when a character is loaded into TXBUF. This flag asserts a transmitter interrupt if the interrupt enable bit SCI TX INT ENA (TXCTL.0) is set. TXRDY is a read-only flag. It is set to 1 by an SCI SW RESET or a system reset.

0 = TXBUF is full.
1 = TXBUF is ready to receive character.

9.3.5 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 after 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 = RXRDY/BRKDT interrupt disabled.
1 = RXRDY/BRKDT interrupt enabled.

Bit 1 - **RXWAKE.** Receiver Wakeup Detect.
The SCI sets this bit when a receiver wakeup 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 RX RESET, or by a system Reset. See Section 9.2.5.

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 is not enabled, the PE flag is disabled and read as 0. The PE bit is reset by an SCI SW RESET, 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, a system reset, or reading RXBUF.

0 = No Overrun error detected.
1 = Overrun error detected.

Bit 4 - **FE.** SCI Framing Error Flag.
The SCI sets this bit when a stop bit is not found when expected. Only the first stop bit is checked. The missing stop bit indicates that synchronization with the start bit has been lost and the character is incorrectly framed. It is reset by an SCI SW RESET, a system reset, or by reading RXBUF.

0 = No Framing error detected.
1 = Framing error detected.

Bit 5 - **BRKDT**. SCI Break Detect Flag.
The SCI sets this bit when a break condition occurs. A break condition occurs when the SCIRXD line remains continuously low for at least 10 bits beginning after a missing first stop bit. The occurrence of a break causes a receiver interrupt to be generated if the SCI RX INT ENA bit is a 1, but it does not cause the receiver buffer to be loaded. A BRKDT interrupt can occur even if the receiver SLEEP bit is set to 1. BRKDT is cleared by reading RXBUF, by an SCI SW RESET, or by a system reset. It is not cleared by receipt of a character after the break is detected.

Bit 6 - **RXRDY**. SCI Receiver Ready.
The receiver sets this bit to indicate that RXBUF is ready with a new character, and clears the bit when the character is read. A receiver interrupt is generated if the SCI RX INT ENA bit is a '1'. RXRDY is reset by an SCI SW RESET or a system reset.

Bit 7 - **RX ERROR**. SCI Receiver Error Flag.
The RX ERROR Flag indicates that one of the error flags in the receiver status register is set. It is a logical "or" of the parity, overrun, framing error, and break detect flags. The bit can be used for fast error condition checking during the interrupt service routine since a negative value of the status register indicates that an error condition has occurred. This error flag cannot be cleared directly, but is cleared if no individual error flags are set. This bit is cleared by an SCI SW RESET, a system reset, or reading RXBUF.

9.3.6 Receiver Data Buffer Register (RXBUF)

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

Receiver Data Buffer Register (RXBUF)
[Memory address - 1057h]

Bit #-	7	6	5	4	3	2	1	0
P057	RXDT7	RXDT6	RXDT5	RXDT4	RXDT3	RXDT2	RXDT1	RXDT0
	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

R=Read, -n= Value after RESET

9.3.7 Transmit Data Buffer Register (TXBUF)

The TXBUF register is a read/write register used to store 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.

Transmit Data Buffer Register (TXBUF)
[Memory address - 1059h]

Bit # -	7	6	5	4	3	2	1	0
P059	TXDT7	TXDT6	TXDT5	TXDT4	TXDT3	TXDT2	TXDT1	TXDT0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Read, W=Write, -n= Value after RESET

9.3.8 Port Control Register 1 (SCIPC1)

The SCIPC1 register controls the SCICLK pin functions.

SCI Port Control Register 1 (SCIPC1)
[Memory address - 105D]

Bit # -	7	6	5	4	3	2	1	0
P05D	---	---	---	---	SCICLK DATA IN	SCICLK DATA OUT	SCICLK FUNCTION	SCICLK DATA DIR
					R-0	RW-0	RW-0	RW-0

R=Read, W=Write, -n= Value after RESET

Bit 0 - **SCICLK DATA DIR.** SCICLK Data Direction.
This bit determines the data direction on the SCICLK pin if SCICLK has been configured as a general purpose I/O pin.

0 = SCICLK pin is a general purpose INPUT port.
1 = SCICLK pin is a general purpose OUTPUT port.

Bit 1 - **SCICLK FUNCTION.**
This bit defines the function of the SCICLK pin.

0 = SCICLK pin is a general purpose digital I/O port.
1 = SCICLK pin is the SCI serial clock pin.

Bit 2 - **SCICLK DATA OUT.**
This bit contains the data to be output on the SCICLK pin if the following conditions are met:
a. SCICLK pin is configured as general purpose I/O.
b. SCICLK pin data direction is defined as output.

Bit 3 - **SCICLK DATA IN.**
The SCICLK DATA IN bit contains the current value on the SCICLK pin.

Bits 4-7 - Reserved. Read data is indeterminate.

9.3.9 Port Control Register 2 (SCIPC2)

The SCIPC2 register controls the SCIRXD and SCITXD pin functions.

SCI Port Control Register 2 (SCIPC2)
[Memory address - 105E]

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 after 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 port.
1 = SCIRXD pin is a general purpose OUTPUT port.

Bit 1 - **SCIRXD FUNCTION.**
This bit defines the function of the SCIRXD pin.

0 = SCIRXD pin is a general purpose digital I/O port.
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:
a. SCIRXD pin has been defined as a general purpose I/O pin.
b. 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 port.
1 = SCITXD pin is a general purpose OUTPUT port.

Bit 5 - **SCITXD FUNCTION.**
This bit defines the function of the SCITXD pin.

0 = SCITXD pin is a general purpose digital I/O port.
1 = SCITXD pin is the SCI Transmit pin.

Bit 6 - **SCITXD DATA OUT.**
This bit contains the data to be output on the SCITXD pin if the following conditions are met:
a. SCITXD pin data direction is defined as output.
b. SCITXD pin is configured as general purpose I/O.

Bit 7 - **SCITXD DATA IN.**
This bit contains the current value on the SCITXD pin.

9.3.10 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 privileged mode.

SCI Priority Control Register (SCIPRI)
[Memory address - 105F]

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, P=Privileged State write only, -n= Value after RESET

Bits 0-3 - Reserved. Read values are indeterminate.

Bit 4 - **SCI ESPEN.** SCI Emulator Suspend Enable.
This bit has no effect except when 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.**
This bit must be cleared (0) to ensure proper operation.

10. Serial Peripheral Interface (SPI) Module

This section discusses the architecture and programming of the Serial Peripheral Interface module on TMS370 devices.

This section covers the following topics:

10.1 Serial Peripheral Interface (SPI) Module 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 using devices such as shift registers, display drivers, A/D converters, etc. 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 A-10, consists of:

- Three I/O pins:
  - SPISIMO - SPI Slave In, Master Out.
  - SPISOMI - SPI Slave Out, Master In
  - SPICLK - SPI CLOCK
- SPIBUF - SPI Buffer register
- SPIDAT - SPI Data Shift register
- State Control logic
- SPI Control registers located at P030–P03F

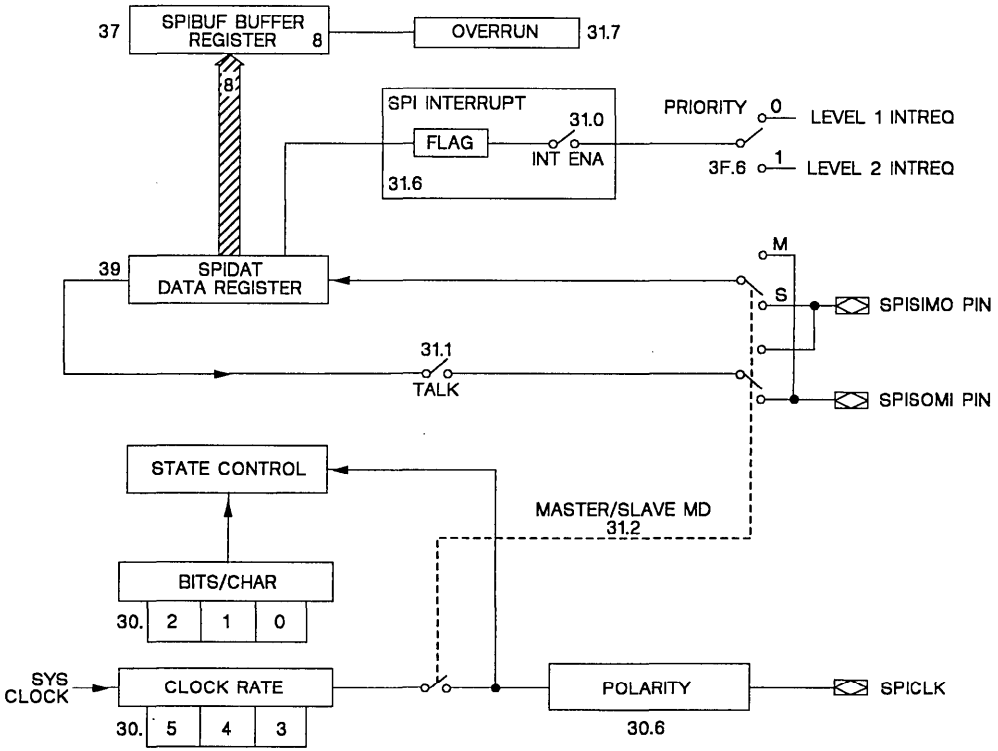


Figure 10-1. SPI Block Diagram

10

10.1.2 SPI Control Registers

The SPI Control registers occupy Peripheral File Frame 3 as shown in Figure 10-2.

Table 10-1. SPI Memory Map

Peripheral File Location	Symbol	Name
P030	SPICCR	SPI Configuration Control Register
P031	SPICTL	SPI Control Register
P032–P036		Reserved
P037	SPIBUF	Receive Data Buffer Register
P038		Reserved
P039	SPIDAT	Serial Data Register
P03A–P03C		Reserved
P03D	SPIPC1	SPI Pin Control 1
P03E	SPIPC2	SPI Pin Control 2
P03F	SPIPRI	SPI Priority Control

10.2 SPI Operation

Figure 10-2 shows a typical connection of the SPI for communications between two microcontrollers. One controller, the master, initiates data transfer by sending the SPICLK signal. Data is enabled out of both shift registers on one edge of the clock and latched into both shift registers on the opposite clock edge. Thus both controllers send and receive data at the same time. Whether or not the data is meaningful or "dummy" data depends on the application software.

There are three possible cases for data transmission:

- Master sends data and Slave sends "dummy" data
- Master sends data and Slave sends data
- Master sends "dummy" data and Slave sends data

The Master can initiate data transfer at any time because it controls the SPICLK. The manner in which the master knowns when the Slave wishes to broadcast data is determined by the software protocol.

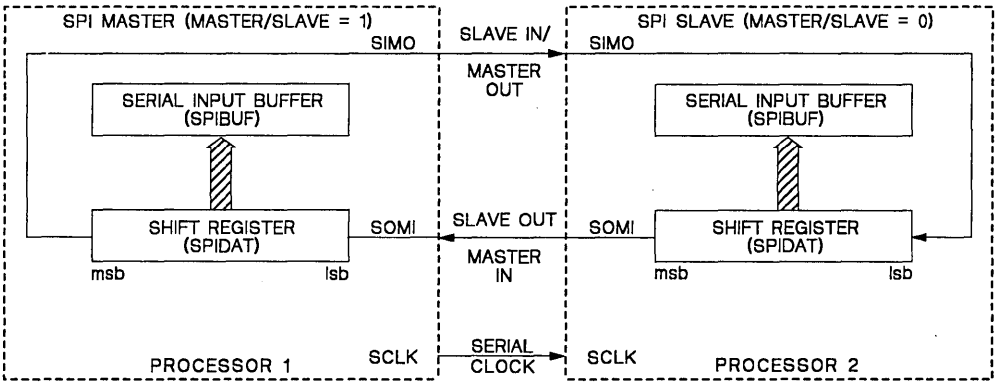


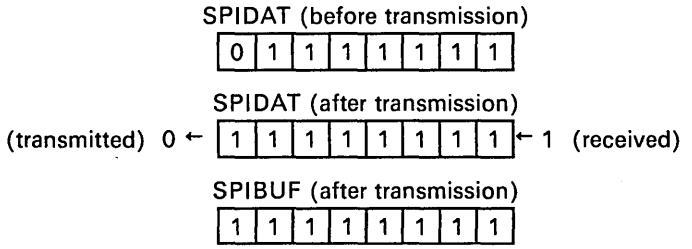
Figure 10-2. SPI Master/Slave Connection

10.2.1 SPI 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:

1) Data must be written to SPIDAT left justified.
2) Data must be read back from SPIBUF right justified.
3) SPIBUF contains the most recently received character, right justified, plus any bits left over from previous transmission(s) which have been shifted to the msb position.

For example:
If the character length = 1 bit, and
the value written into SPIDAT = 07Fh,
then;



10.2.2 SPI Interrupts

The interrupt for the SPI is controlled by bits in two registers. The SPI INT ENA bit (SPICTL.0), when set, allows assertion of an interrupt request when an interrupt condition occurs. The SPI PRIORITY bit (SPIPRI.6) determines whether SPI interrupts are level 1 or level 2 priority requests.

When a complete character has been shifted into or out of the SPIBUF register, the SPI Interrupt Flag is set and an interrupt is generated if enabled by SPI INT ENA (SPICTL.0). The interrupt flag remains set until cleared by one of the following four events.

- CPU reads the SPI receiver buffer (SPIBUF),
- CPU enters the Halt or Standby mode with an IDLE instruction,
- Software sets the SPI SW RESET bit, or
- A System resets occurs.

An interrupt request must be explicitly cleared by one of the four methods listed above to avoid generating another interrupt. An interrupt request can be temporarily disabled by clearing the SPI INT ENA bit. However, unless the SPI INT FLAG itself is cleared, the interrupt request will be reasserted when the enable bit is again set to 1.

The priority level of the SPI interrupt is specified by the SPI PRIORITY bit (SPIPRI.6). If SPI PRIORITY = 0, then a level 1 priority interrupt is generated. If SPI PRIORITY = 1, then a level 2 priority interrupt is generated.

The SPI INT FLAG bit indicates, when set, that a character has been placed into the SPIBUF register and is ready to be read. If the CPU does not read the character by the time the next complete character has been received, the new character is written into the SPIBUF and the RECEIVER OVERRUN bit (SPICTL.7) is set. This indicates that the last character of data has been overwritten with new data before the previous character could be read.

10.2.3 SPI 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:

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

where b=bit rate in SPICCR.5-3 (range 0-7).

10.2.4 SPI Operation Modes

The MASTER/SLAVE bit (SPICTL.2) selects the operating mode and the source of SPICLK. The SPI module may operate as a Master or Slave.

10.2.4.1 Master

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 SPICCR register (SPI BIT RATE0-2) determines the bit transfer rate for the network, both transmit and receive. There are eight data transfer rates that can be selected by these control bits as shown in Table 9-3 on page 9-20.

Data written to the SPIDAT register initiates data transmission on the SPISIMO pin, msb of data transmitted first. Simultaneously, received data is shifted in the SPISOMI pin into the SPIDAT register, and upon completion of transmitting the selected number of bits, 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 most significant bit first; there, it is stored right-justified in the SPIBUF.

To receive a character when operating as a master, data must be written to the SPIDAT to initiate the transaction. When the specified number of data bits have been shifted through the SPIDAT register, the following events occur:

1) The SPI INT FLAG bit is set,
2) SPIDAT contents transfer to SPIBUF, and
3) If the SPI INT ENA bit is set to one, an interrupt is asserted.

Writing to the SPIDAT register before transmission is complete corrupts the current transmission.

10.2.4.2 Slave

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 the input clock on the SPICLK pin, which is supplied from the network master. The SPICLK input frequency should be no greater than CLKIN frequency divided by 32.

Data written to the SPIDAT register is transmitted to the network when the SPICLK is received from the network master. To receive data, the SPI waits for the network master to send SPICLK and then shifts the data on the SPISIMO pin into the SPIDAT register. If data is to be transferred by the slave simultaneously, then it must be written to the SPIDAT register prior to the beginning of SPICLK.

When the TALK bit (SPICTL.1) is cleared, data transmission is disabled and the output line is put into a high impedance state. This allows many slave devices to be tied together on the network, but only one slave is allowed to talk at a time.

10.2.5 Initialization

A system reset forces the SPI peripheral module into the following default configuration.

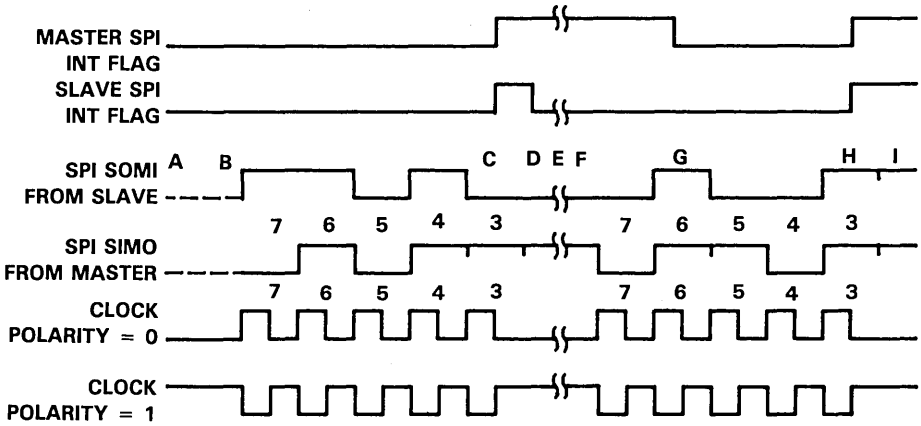
- The unit is configured as a slave module (MASTER/SLAVE = 0).
- The transmit capability is disabled (TALK = 0).
- Data is latched at the input on the falling edge of SPICLK.
- Character length is assumed to be 1 bit.
- The SPI interrupts are disabled.
- Data in the SPI Data Register is 00h.

To change this SPI configuration it is a good idea to use the SPI SW RESET bit. Set the SPI SW RST bit (SPICCR.7); make your desired changes; then clear the SPI SW RST bit. This prevents unwanted and unforeseen events from occurring during or as a result of mode change.

10.2.6 SPI Example

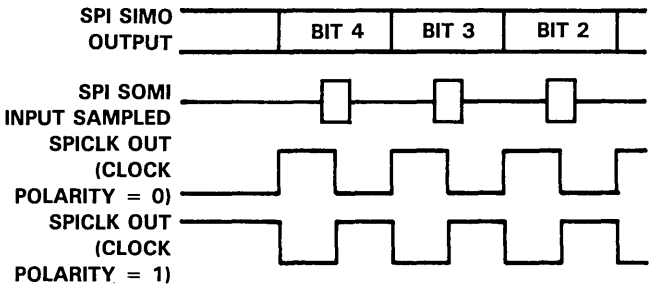
The following timing diagrams illustrate an example SPI data transfer between two TMS370 devices using a character length of five bits. The lettered notes following the first diagram are keyed to the letter labels in the diagram.

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 set the interrupt flags.
I. Master received 09h and the Slave received a 0Dh (right justified).

SIGNALS CONNECTING TO MASTER PROCESSOR.



10.3 SPI Control Registers

The SPI is controlled and accessed through registers in the Peripheral File. These registers are listed in Figure 10-3 and described in the following sections. The bits shown in shaded boxes in Figure 10-3 are Privilege Mode bits, that is, they can only be written to in the Privilege Mode.

PERIPHERAL FILE FRAME 3: SERIAL PERIPHERAL INTERFACE (SPI) CONTROL REGISTERS

ADDR	PF	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	
1030h	30	SPI SW RESET	CLOCK POLARITY	SPI BIT RATE2	SPI BIT RATE1	SPI BIT RATE0	SPI CHAR2	SPI CHAR1	SPI CHAR0	SPICCR
1031h	31	RECEIVER OVERRUN	SPI INT FLAG	---	---	---	MASTER/ SLAVE	TALK	SPI INT ENA	SPICTL
1032h TO 1036h	32 TO 36	RESERVED								
1037h	37	SPI DATA BUFFER REGISTER								SPIBUF
1038h	38	RESERVED								
1039h	39	SPI SERIAL DATA REGISTER								SPIDAT
103Ah TO 103Ch	3A TO 3C	RESERVED								
103Dh	3D	---	---	---	---	SPICLK DATA IN	SPICLK DATA OUT	SPICLK FUNCTION	SPICLK DATA DIR	SPIPC1
103Eh	3E	SPISIMO DATA IN	SPISIMO DATA OUT	SPISIMO FUNCTION	SPISIMO DATA DIR	SPISOMI DATA IN	SPISOMI DATA OUT	SPISOMI FUNCTION	SPISOMI DATA DIR	SPIPC2
103Fh	3F	SPI STEST	SPI PRIORITY	SPI ESPEN	---	---	---	---	---	SPIPRI

Figure 10-3. SPI Control Registers

10.3.1 SPI Configuration Control Register

The SPICCR register controls the setup of the SPI for operation. A summary of the register functions and bit assignments is shown below.

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 after RESET

Bits 0-2 **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.

Table 10-2. SPI Character Bit Length

CHAR2	CHAR1	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 eight bit, free-running prescaler from which eight taps are available for use as the shift clock. One data bit is shifted per SPICLK cycle.

Table 10-3. SPI Clock Frequency

SPI† BIT RATE2	SPI† BIT RATE1	SPI† BIT RATE0	SPI CLOCK FREQUENCY
0	0	0	CLKIN/8
0	0	1	CLKIN/16
0	1	0	CLKIN/32
0	1	1	CLKIN/64
1	0	0	CLKIN/128
1	0	1	CLKIN/256
1	1	0	CLKIN/512
1	1	1	CLKIN/1024

†If the SPI is a network slave, the module receives a clock on the SPICLK pin from the network master; and these bits have no effect on SPICLK. The frequency of the input clock should be no greater than the CLKIN frequency divided by 32.

Bit 6 **CLOCK POLARITY.** Shift Clock Polarity.
The CLOCK POLARITY bit controls the polarity of the SPICLK signal.

0 = The inactive level is low; data is output by the rising edge of SPICLK; input data is latched by the falling edge of SPICLK.
1 = The inactive level is high; data is output by the falling edge of SPICLK; input data is latched by the rising edge of SPICLK.

Bit 7 - **SPI SW RESET.** SPI Software Reset.
Writing a 1 to this bit initializes the SPI circuitry and operating flags to the reset condition. Specifically, the RECEIVER OVERRUN and SPI INT FLAG flags are cleared. The SPI configuration remains unchanged. If it is operating as a master, the SPICLK output level returns to its inactive level.

When a "0" is written to SPI SW RESET the SPI is ready to transmit or receive the next character. A character written to the transmitter when SPI SW RESET is a "1" will not be shifted out when SPI SW RESET bit is cleared. A new character must be written to the Serial Data Register. To change any configuration bits, this bit should be used (see Section 10.2.5, page 10-8).

10.3.2 SPI Operation Control Register

The SPI Operation Control Register contains control and status bits as shown below.

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 after 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 = disable interrupt.
1 = enable interrupt.

Bit 1 - **TALK**. Master/Slave Transmit Enable.
This bit allows data transmission (master or slave) to be disabled by placing the serial data output in a high impedance state. TALK is cleared (disabled) by a system reset.

0 = Transmission disabled; if not programmed as a general purpose I/O pin, the SPI serial output is in a high impedance state.
1 = Transmission enabled.

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 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. 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 which 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 over-written, and therefore has been lost. RECEIVER OVERRUN is cleared when the receiver buffer is read. It is also cleared by SPI SW RESET or a system reset.

10.3.3 Serial Input Buffer (SPIBUF)

The SPIBUF register contains the data received from the network ready for the CPU to read.

Serial Input Buffer, (SPIBUF)
[Memory Address - 1037h]

Bit # -	7	6	5	4	3	2	1	0
P037	RCVD7 (msb)	RCVD6	RCVD5	RCVD4	RCVD3	RCVD2	RCVD1	RCVD0 (lsb)
	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

R=Read, -n= Value after RESET

Once the Serial Data Register has received the complete character, the character is then transferred to the SPIBUF Register where it can be read. The SPI INT FLAG bit (SPICTL.6) is set to indicate that the data is available when the received character is transferred. Since data is shifted into the SPI most significant bit first, it is stored right justified in the SPIBUF.

10.3.4 Serial Data Register (SPIDAT)

The SPIDAT register is the transmit/receive shift register. Data written to the SPIDAT is shifted out on subsequent SPICLK cycles. For every bit shifted out of the SPI a bit is shifted into the other end of the shift register.

Writing to the SPIDAT performs two functions. First, it provides data to be output on the serial output pin if the TALK bit is set. Second, when the SPI is operating as a master, writing to this register initiates a transaction.

To initiate a receiver sequence, dummy data is written to the register. Since the data is not hardware justified for characters that are shorter than eight bits, transmit data must be written in left justified form and received data read in right justified form.

Serial Data Register, (SPIDAT)
[Memory Address - 1039h]

Bit # -	7	6	5	4	3	2	1	0
P039	SDAT7	SDAT6	SDAT5	SDAT4	SDAT3	SDAT2	SDAT1	SDAT0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Read, W=Write, -n= Value after RESET

10.3.5 Port Control Registers

Two Port Control Registers (SPIPC1 and SPIPC2) allow a programmer to control all functions for a SPI port pin in one write cycle. Each module pin is controlled by a nibble in one of the SPIPC's.

10.3.5.1 Port Control Register 1 (SPIPC1)

This register controls the SPICLK pin.

Port Control Register 1, (SPIPC1)
[Memory Address - 103Dh]

Bit # -	7	6	5	4	3	2	1	0
P03D	---	---	---	---	SPICLK DATA IN	SPICLK DATA OUT	SPICLK FUNCTION	SPICLK DATA DIR
					R-0	RW-0	RW-0	RW-0

R=Read, W=Write, -n= Value after 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 port.
1 = SPICLK pin is a general purpose OUTPUT port.

Bit 1 - **SPICLK FUNCTION**. SPICLK Pin Function Select.
This bit defines the function of the SCICLK pin.

0 = SPICLK pin is a general purpose digital I/O port.
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:
a. SCICLK pin has been defined as a general purpose I/O pin.
b. SCICLK 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 SCICLK 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., SPICLK.2=0) even if SPICLK FUNCTION = 0 and SPICLK DATA DIR = 0.

10.3.5.2 Port Control Register 2

The SPIPC2 register controls the SPISOMI and SPISIMO pin functions.

Port Control Register 2, (SPIPC2)
[Memory Address - 103Eh)

Bit # -	7	6	5	4	3	2	1	0
P03E	SPISIMO DATA IN	SPISIMO DATA OUT	SPISIMO FUNCTION	SPISIMO DATA DIR	SPISOMI DATA IN	SPISOMI DATA OUT	SPISOMI FUNCTION	SPISOMI DATA DIR
	R-0	RW-0	RW-0	RW-0	R-0	RW-0	RW-0	RW-0

R=Read, W=Write, -n= Value after RESET

Bit 0 - **SPISOMI DATA DIR.** SPISOMI Data Direction.
This bit determines the data direction on the SPISOMI pin if SPISOMI has been defined as a general purpose I/O pin.

0 = SPISOMI pin is a general purpose INPUT port.
1 = SPISOMI pin is a general purpose OUTPUT port.

Bit 1 - **SPISOMI FUNCTION.** SPISOMI Pin Function Select.
This bit defines the function of the SPISOMI pin. When SPISOMI is an input and SPISOMI FUNCTION and SPISOMI DATA DIR are disabled, then SPICLK still clocks the internal circuitry.

0 = SPISOMI pin is a general purpose digital I/O port.
1 = SPISOMI pin contains the SPI data.

Bit 2 - **SPISOMI DATA OUT.** SPISOMI Pin Data Out.
This bit contains the data to be output on the SPISOMI pin if the following conditions are met:
a. SPISOMI pin has been defined as a general purpose I/O pin.
b. SPISOMI pin data direction has been defined as output.

10

Bit 3 - **SPISOMI DATA IN.** SPISOMI Pin Data In.
This bit contains the current value on the SCISOMI pin regardless of the mode. A write to this bit has no effect.

Bit 4 - **SPISIMO DATA DIR.** SPISIMO Data Direction.
This bit determines the data direction on the SPISIMO pin if SPISIMO has been defined as a general purpose I/O pin.

0 = SPISIMO pin is a general purpose INPUT port.
1 = SPISIMO pin is a general purpose OUTPUT port.

Bit 5 - **SPISIMO FUNCTION.** SPISIMO Pin Function Select.
This bit defines the function of the SPISIMO pin.

0 = SPISIMO pin is a general purpose digital I/O port.
1 = SPISIMO pin contains the SPI data.

Bit 6 - **SPISIMO DATA OUT.** SPISIMO Pin Data Out.
This bit contains the data to be output on the SPISIMO pin if the following conditions are met:
a. SPISIMO pin has been defined as a general purpose I/O pin.
b. 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 SCISIMO pin regardless of the mode. A write to this bit has no effect.

10.3.6 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 privileged mode.

SPI Interrupt Priority Control Register, (SPIPRI)
[Memory Address - 103Fh]

Bit # -	7	6	5	4	3	2	1	0
P03F	SPI STEST	SPI PRIORITY	SPI ESPEN	---	---	---	---	---
	RP-0	RP-0	RP-0					

R=Read, W=Write, P=Privileged Write only, -n= Value after RESET

Bits 0-4 - Reserved. Read data is indeterminate.

Bit 5 - **SPI ESPEN.** Emulator Suspend Enable.
This bit has no effect except when 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.**
This bit must be cleared (0) to ensure proper operation.

11. Analog-To-Digital Converter Module

This section discusses the architecture and programming of the Analog-to-Digital Converter module on TMS370C050 and TMS370C850 devices.

This section covers the following topics:

11.1 Analog-To-Digital Converter (A/D) Overview

The Analog-to-Digital Converter module (A/D) is an 8 bit successive approximation converter with internal sample-and-hold circuitry. The module has eight multiplexed analog input channels which allows the processor to convert the voltage levels from up to 8 different sources.

11.1.1 A/D 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 (+VREF),
- the Analog-to-Digital Converter (A/D),
- the ADDATA register which contains the digital value of a completed conversion, and
- 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 has completed.

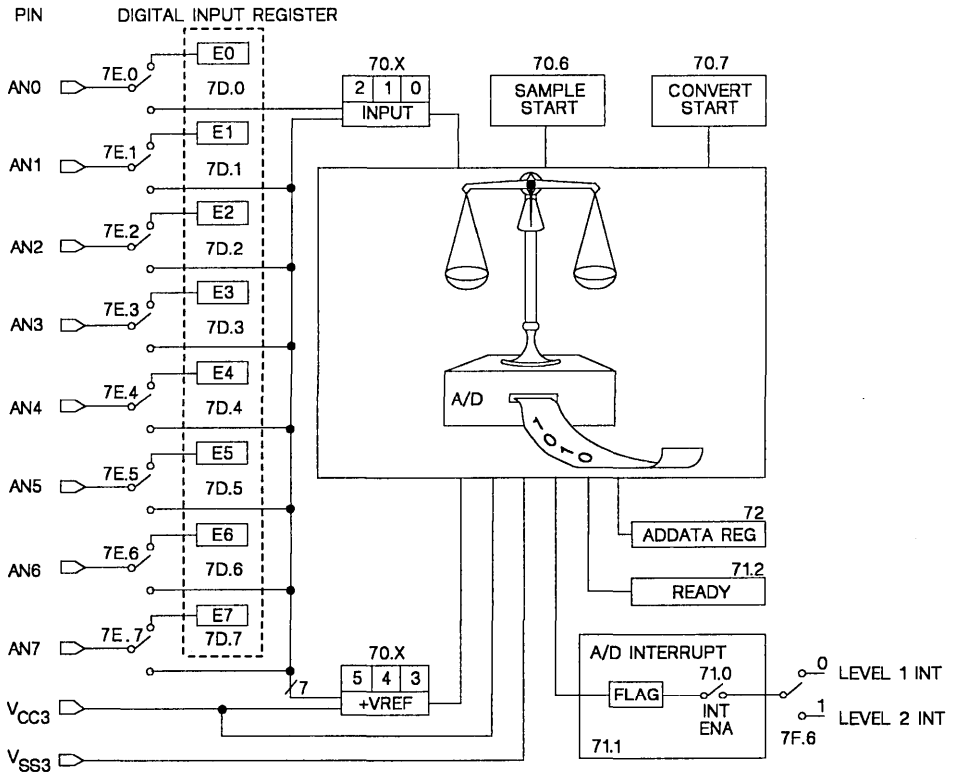


Figure 11-1. Analog-to-Digital Converter Block Diagram

11

11.1.2 A/D Control Registers

The A/D Control registers occupy Peripheral File Frame 7 as shown in Table 11-1.

Table 11-1. A/D Memory Map

Peripheral File Location	Symbol	Name
P070	ADCTL	Analog Control Register
P071	ADSTAT	Analog Status and Interrupt Register
P072	ADDATA	Analog Conversion Data Register
P073–P07C		Reserved
P07D	ADIN	Port E Data Input Register
P07E	ADENA	Port E Input Enable Register
P07F	ADPRI	Port E Interrupt Priority Register

11.2 A/D Operation

The following sections describe the functions and options of the A/D module.

11.2.1 A/D Input/Output Pins

The A/D module uses 10 pins to connect to the external world. Eight of the 10 pins (AN0–AN7) are individually configured as general purpose input pins when not used as analog inputs.

Seven of the eight analog channels (AN1–AN7) 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 the digital switching noise which may be present on the other power supply pins (V_{CC1}, V_{CC2}, V_{SS1}, and V_{SS2}). This isolation provides a more accurate conversion. Power to the V_{CC3} and V_{SS3} pins should run on separate conductors from the other power lines. Power conductors to the V_{CC3} and V_{SS3} should be as short as possible, and the two lines should be properly decoupled. Other standard noise reduction techniques should be applied to help provide a more accurate conversion.

V_{REF} can be chosen 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 A/D Sampling Time

The application program controls the length of the sample time which provides the flexibility to optimize the conversion process for both high and low impedance sources. The program should wait 1 µs for each kilohm of source output impedance or a minimum of 1 us for low impedance sources.

11.2.3 A/D Conversion

The digital result of the conversion process is given in the following formula.

Digital result = 255 * Voltage of input / Voltage of reference

The conversion process takes 164 cycles which results in a conversion time of 32.8 microseconds at 20 MHz. A maximum of 27,600 conversions per second are possible at 20 MHz including setting up the conversion, sampling, converting and saving the results.

In Ratiometric conversions, the conversion value is a ratio of the V_{REF} source to the analog input. As V_{REF} is increased, the input voltage needed to give a certain conversion value changes; but all conversion values keep the same relationship to V_{REF}. That is, one half of V_{REF} always results in the value 080h regardless of the value of V_{REF} (assuming that V_{REF} is in the range of 2.5 to 5.5 volts above V_{SS3}).

Figure 11-2 shows an example of Ratiometric conversion. In this example, the digital result of the conversion indicates the position of the potentiometer wiper even if the battery loses voltage over time. The A/D conversion always gives the ratio of the resistor values on either side of the wiper even if V_{REF} drops from 5.0 to 2.5 volts.

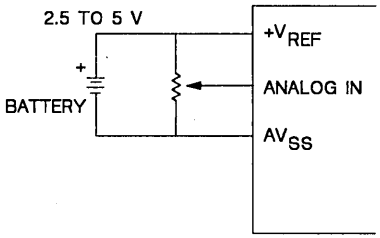


Figure 11-2. Ratiometric Conversion Example

11.2.4 A/D Interrupts

The A/D module sets the AD INT FLAG bit (ADSTAT.1) at the end of the conversion process. If both the AD INT FLAG and the AD INT ENA bit (ADSTAT.0) are set, then the module generates an interrupt request. This interrupt request may be asserted on either the high priority level 1 or the lower priority level 2 depending on the AD PRIORITY bit (ADPRI.6).

The program must clear the AD INT FLAG or else the same interrupt will cause the CPU to enter the interrupt routine again. If the AD INT ENA bit is cleared without clearing the flag, the interrupt is reasserted when the AD INT ENA bit is again set.

11.2.5 A/D Programming Considerations

The programmer should follow these steps to obtain data from the A/D converter.

1) Write to the ADCTL register to:
   - Select the Analog channel (ADCTL.2-0).
   - Select the V_{REF} source (ADCTL.5-3).
   - Set the SAMPLE START bit to 1 (ADCTL.6).
2) Wait for the sample time to elapse.
3) 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).

To begin sampling, set the SAMPLE START bit. The program should wait 1 µs for each kilohm of source output impedance or a minimum of 1 µs for low impedance sources. When the sample time completes, set both the SAMPLE START and CONVERT START bits.

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 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.

11.3 A/D Example Program

This example program samples and converts data from all 8 channels and stores the digital results into a table beginning at ATABLE. The routine stops interrupting the main program after it finishes all eight channels. If the main program wants more recent data it needs only to execute the code at RESTART and the A/D routine will again sample and convert all eight channels of data. The A/D interrupt enable bit is cleared by the A/D interrupt routine as a signal to the main program that all eight channels have been processed. The address of the label ATOD must be placed into the interrupt vector table located at 7FECh and 7FEDh.

```
ADCTL     .EQU   P070          ;A/D control register
ADSTAT    .EQU   P071          ;A/D status register
ADDATA    .EQU   P072          ;A/D conversion results
ADENA     .EQU   P07E          ;A/D input enable
          .REG   ADCHANL       ;keeps current channel number
          .REG   ATABLE,8      ;8 byte table that stores
                               ; channel data, lsb first
;
INIT      MOV    #0,ADENA      ;all channels to A/D inputs
                               ; (reset condition)
          CALL   RESTART       ;start the interrupts now
;
;         MAIN PROGRAM GOES HERE
;          .
;          .
          CALL   RESTART       ;start taking more data
;          .
;          .
;         MORE MAIN PROGRAM
;
;         SUBROUTINE SECTION
RESTART   CLR    ADCHANL       ;initialize channel
          MOV    #001h,ADSTAT  ;enable interrupts, clear
                               ;  any flag
          MOV    #040h,ADCTL   ;start sampling (approx. 2 µs
                               ;  delay)
          MOV    #0C0h,ADCTL   ;start converting now; enter
                               ;  main program
          RTS
;
;         INTERRUPT ROUTINE FOR ANALOG TO DIGITAL CONVERTER
ATOD      PUSH   A             ;save registers
          PUSH   B
          MOV    ADCHANL,B     ;get channel number
          MOV    ADDATA,A      ;get A/D conversion value
          MOV    A,ATABLE(B)   ;store in a table according to
                               ;  channel number
          INC    B             ;point to next channel
          BTJZ   #8,B,GOCNVRT  ;stop when all channels sampled
                               ;  (bit3 =1)
          CLR    ADCHANL       ;reset the A/D channel
          MOV    #0,ADSTAT     ;turn off interrupt and
                               ; clear flag
          JMP    EXITA2D       ;all 8 channels taken, enable
                               ;  set to 0 now
```

```
GOCNVRT   MOV     B,ADCHANL       ;store current A/D channel
          MOV     #01h,ADSTAT     ;clear interrupt flag to prevent
                                  ;  more interrupts
          OR      #040h,B         ;set up sample bit in value
          MOV     B,ADCTL         ;start sampling channel data
          OR      #080h,ADCTL     ;start converting data
;
EXITA2D   POP     B               ;Restore data
          POP     A
          RTI
```

11.4 A/D Control Registers

The A/D module control registers occupy peripheral file frame 7, as shown in Table 11-2. The bits shown in shaded boxes in Table 11-2 are Privilege Mode bits, that is, they can only be written to in the Privilege Mode.

Table 11-2. Peripheral File Frame 7: A-to-D Converter Control Registers

PERIPHERAL FILE FRAME 7: A-TO-D CONVERTER CONTROL REGISTERS

ADDR	PF	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	
1070h	070	CONVERT START	SAMPLE START	REF VOLT SELECT 2	REF VOLT SELECT 1	REF VOLT SELECT 0	AD INPUT SELECT 2	AD INPUT SELECT 1	AD INPUT SELECT 0	ADCTL
1071h	071	---	---	---	---	---	AD READY	AD INT FLAG	AD INT ENA	ADSTAT
1072h	072	A-TO-D CONVERSION DATA REGISTER								ADDATA
1073h TO 107Ch	073 TO 07C	RESERVED								
107Dh	07D	PORT E DATA INPUT REGISTER								ADIN
107Eh	07E	PORT E INPUT ENABLE REGISTER								ADENA
107Fh	07F	AD STEST	AD PRIORITY	AD ESPEN	---	---	---	---	---	ADPRI

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 after 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 SAMPLE START or CONVERT START is 1 invalidates the conversion in progress.

AD INPUT SELECT2	AD INPUT SELECT1	AD INPUT SELECT0	CHANNEL
0	0	0	AN0
0	0	1	AN1
0	1	0	AN2
0	1	1	AN3
1	0	0	AN4
1	0	1	AN5
1	1	0	AN6
1	1	1	AN7

Bits 3-5 - **REF VOLT SELECT0–2**. Reference Voltage ($+V_{REF}$) Select Bits 0–2. These bits select the channel the A/D uses for the positive voltage reference. REF VOLT SELECT bits must not change during the entire conversion.

REF VOLT SELECT2	REF VOLT SELECT1	REF VOLT SELECT0	$+V_{REF}$ SOURCE
0	0	0	V_{CC3}†
0	0	1	AN1
0	1	0	AN2
0	1	1	AN3
1	0	0	AN4
1	0	1	AN5
1	1	0	AN6
1	1	1	AN7

†Pin AN0 can not be selected as positive voltage reference.

Bit 6 - **SAMPLE START**. Sample Start.
Setting this bit stops any ongoing conversion and starts sampling the selected input channel to begin a new conversion. This bit is cleared by the A/D module 18 system-clock cycles after the program sets the CONVERT START bit. Entering Halt or Standby mode clears this bit and aborts any sampling in progress.

Bit 7 - **CONVERT START**. Conversion Start.
Setting this bit starts the conversion. This bit is cleared by the A/D 18 system clock cycles after the program sets the CONVERT START bit. Entering Halt or Standby mode clears this bit and aborts any conversion in progress.

11.4.2 Analog Status and Interrupt Register, (ADSTAT)

The ADSTAT register indicates the converter and interrupt status.

Analog Status and Interrupt Register, (ADSTAT)
[Memory Address - 1071h]

Bit # -	7	6	5	4	3	2	1	0
P071	---	---	---	---	---	AD READY	AD INT FLAG	AD INT ENA
						R-1	RC-0	RW-0

R=Read, W=Write, C=Clear only, -n= Value after 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 or by entering Halt or Standby mode. 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)

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 after RESET

The analog-to-digital conversion data is loaded into this register at the end of a conversion and remains until replaced by another conversion.

11.4.4 Analog Port E Data Input Register (ADIN)

The ADIN register contains digital input data when one or more of the AN0 through AN7 pins are used as digital ports.

Analog Port E Data Input Register (ADIN)
[Memory Address - 107Dh]

Bit # -	7	6	5	4	3	2	1	0
P07D	PORT E DATA AN 7	PORT E DATA AN 6	PORT E DATA AN 5	PORT E DATA AN 4	PORT E DATA AN 3	PORT E DATA AN 2	PORT E DATA AN 1	PORT E DATA AN 0
	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

R=Read, -n= Value after RESET

The ADIN register shows the data present at the pins configured for general purpose input instead of A/D channels. A bit is configured as a general purpose input if the corresponding bit of the port enable register is a 1. Pins configured as A/D channels are read as 0s. Writing to this address has no effect.

11.4.5 Analog Port E Input Enable Register (ADENA)

The ADENA register controls the function of the AN0 through AN7 pins.

Analog Port E Input Enable Register (ADENA)
[Memory Address - 107Eh]

Bit # -	7	6	5	4	3	2	1	0
P07E	PORT E INPUT ENA 7	PORT E INPUT ENA 6	PORT E INPUT ENA 5	PORT E INPUT ENA 4	PORT E INPUT ENA 3	PORT E INPUT ENA 2	PORT E INPUT ENA 1	PORT E INPUT ENA 0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R=Read, W=Write, -n= Value after RESET

The ADENA register individually configures the eight pins AN0-AN7 as either analog input channels or as general purpose input pins.

0 = The pin becomes an analog input channel for the A/D converter. When the bit is 0, the corresponding bit in the ADIN register reads as a '0'.
1 = Enables the pin as a general purpose input pin and its digital value can be read from the corresponding bit in the Port E Data Input Register.

11.4.6 Analog Interrupt Priority Register (ADPRI)

The ADPRI register selects the interrupt priority level of the A/D interrupt.

Analog Interrupt Priority Register (ADPRI)
[Memory Address - 107Fh]

Bit # -	7	6	5	4	3	2	1	0
P07F	AD STEST	AD PRIORITY	AD ESPEN		---	---	---	---
	RP-0	RP-0	RP-0					

R=Read, P=Privileged Write, -n= Value after RESET

Bits 0-4 - Reserved. Read data is indeterminate.

Bit 5 - **AD ESPEN**. Emulator Suspend Enable.
Normally, this bit has no effect. However, when 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. Assembly Language Instruction Set

An assembly language instruction set is a symbolic language that presents binary machine code in a more readable form. The TMS370 family is supported by a 73-function instruction set using a wide variety of addressing modes.

This section includes the following topics:

12.1 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 a function mnemonic followed by zero to three operands. The mnemonic specifies the type of CP operation while 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 it's own opcode.

The typical syntax for TMS370 instructions consists of the function mnemonic followed by up to three operands. A typical two-operand instruction is shown below:

MNEMONIC	SOURCE	DESTINATION
ADD	#9,	R3

The example above can be read as: add the value "9" to the contents of register number 3 and place the result back into register number 3. The destination, therefore, also serves as a second source in addition to being the final address of the result. This means that registers can be directly manipulated without having to use intermediate registers. It should be noted that this instruction form differs from the "mnemonic- destination-source" arrangement used by some microprocessors.

The following example shows how the instruction above might appear in a complete program line.

LABEL	INST.	OPERANDS	COMMENT
XXXXX	ADD	R9,R3	;comment

There should be at least one space between each entry type. The LABEL and COMMENT entries are optional, and depending on which type of instruction is used, the OPERANDS column may be blank as well.

The 73 instructions are supported by 245 opcodes providing flexible control of CPU program flow. Some instructions such as CLRC and TEST A share the same opcode to aid the user in comprehending all of the functions of an opcode. There are instructions that use 16-bit opcodes, depending on the type of instruction and/or the addressing mode used. 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.

12.2 Addressing Modes

Each TMS370 assembly language instruction includes form 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 table below shows the 14 addressing modes with a sample instruction and it's execution.

Table 12-1 describes the addressing modes of the instruction set.

Table 12-1. 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)

NOTE: PCN = 16-bit address of next instruction.
(x) = Contents of memory at address x.
((x)) = Contents of memory location designated by contents at address x.

As indicated in the table, there are 14 addressing modes divided into two classes: General, which uses an 8-bit addressing range, and Extended, which uses a 16-bit addressing range. A number of instructions use more than one addressing mode and several, such as the MOV instruction, are very versatile.

12.2.1 General Addressing Modes

Instructions using the General Addressing modes have an eight bit range of operation, and therefore deal with the register file, peripheral file, or a nearby destination. The General Addressing modes are Implied, Register, Peripheral, Immediate, Program Counter Relative, and Stack Pointer Relative. Most of these modes can use any register as a source and/or destination, preventing the bottleneck found on other microprocessors that use only one or two registers.

12.2.1.1 Implied Addressing Mode

In the Implied addressing mode, the instruction type alone determines where the data is to be found. The user does not have to specify the operands since they are inherently specified in the instruction. For example, the LDSP (Load Stack Pointer) instruction always copies the contents of the B register to the stack pointer register. Neither the source nor destination is explicitly stated because they are implied in the instruction itself. The instructions using the Implied addressing mode are the CLRC, LDSP, RTS, RTI, SETC, STSP,EINT, EINTH, and EINTL instructions. Figure 12-1 shows an example of the Implied addressing mode.

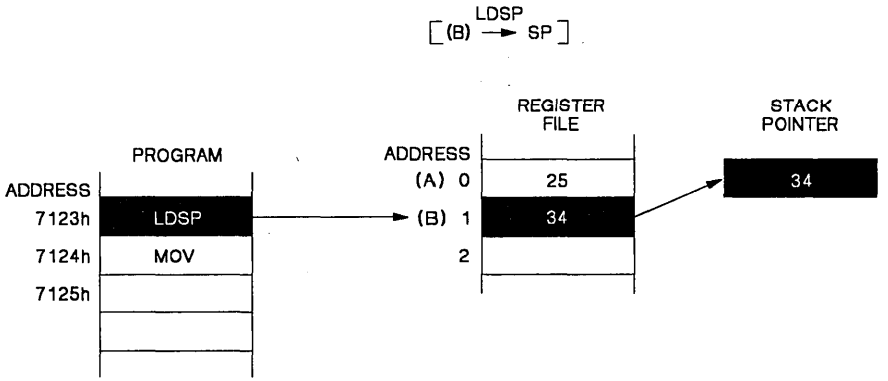


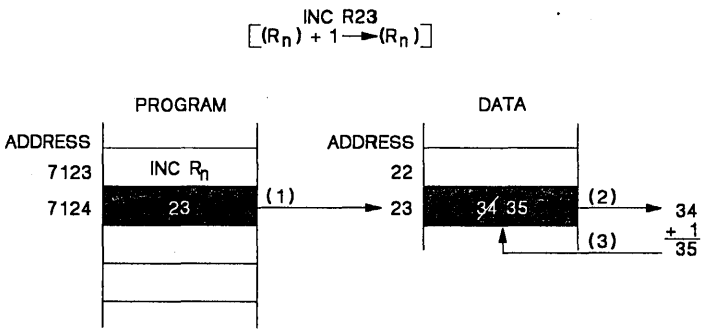
Figure 12-1. Implied Operand Addressing Mode

12.2.1.2 Register Addressing Mode

The Register file of the TMS370 consists of the the first 256 bytes of memory. In the Register addressing mode, instructions use a one byte value to specify an address (location) in the Register file. Any location in the Register file can be accessed in one memory cycle by instructions using this mode. (Extended addressing modes take 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 may 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 1020h
```

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 12-2 shows an example of the Register Addressing Mode.



NOTE: NUMBERS IN PARENTHESIS REPRESENT ORDER OF EXECUTION

Figure 12-2. Register Addressing Mode

12.2.1.3 Peripheral Addressing Mode

The Peripheral file of the TMS370 is allocated 256 bytes of memory. The Peripheral addressing mode is used for program control of the peripheral on-chip modules such as timers, interrupts, and I/O ports. A small amount of external memory can also be addressed as Peripheral file space from the TMS370Cx50. Each Peripheral file register is accessed by an 8-bit operand designated as Pn, with n being either a decimal (0-255) or hexidecimal (00-FF) number. Hexidecimal numbers require a leading zero but no suffix. The CPU assumes the most significant byte of a peripheral address to be 010h. As described in Register file addressing, the Pn designation may be substituted with a symbol using the equate (.EQU) assembler directive as shown in the example below.

```
MOV R16,P020      ;move contents of 0010h to 1020h
CAT .EQU R16      ;Equate register 16 to symbol CAT
DOG .EQU P32      ;Equate peripheral file 32 to symbol DOG
MOV CAT,DOG       ;move contents of 0010h to 1020h
```

The use of designated symbols is optional, of course, but is particularly suited for the register and peripheral addressing modes. Figure 12-3 shows an example of Peripheral-File addressing.

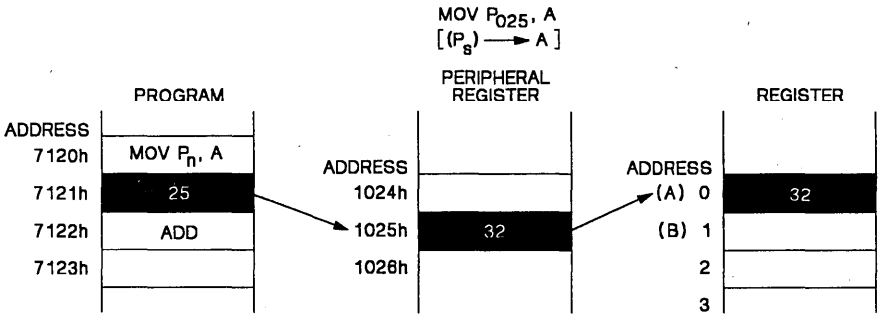


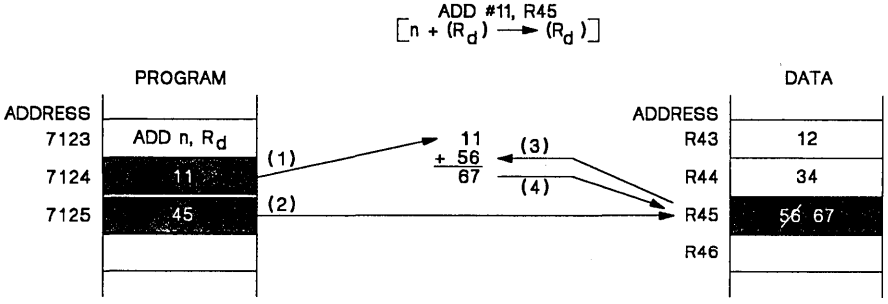
Figure 12-3. Peripheral Addressing Mode

12.2.1.4 Immediate Addressing Mode

The Immediate Addressing mode uses a constant value as the operand immediately following the function mnemonic. This mode allows non-changing data to be incorporated into the instruction. The constant may be in the form of a decimal, hexadeciaml, or symbolic label, but it is always preceded by the number sign (#). 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 sym,bol CNT
ADD #CNT,R34        ;Add the value 12 to register 34, place
                    ;result in register 34.
```

Figure 12-4 illustrates an instruction using the immediate addressing mode.



NOTE: NUMBERS IN PARENTHESIS REPRESENT ORDER OF EXECUTION

Figure 12-4. Immediate Addressing Mode

12.2.1.5 Program Counter Relative Addressing Mode

The Program Counter Relative addressing mode adds an 8-bit signed offset to the address of the next instruction to produce the address of the successive instruction. The new address is placed in the program counter register. The range of the 8-bit offset is within 128 bytes before or 127 bytes after the instruction following the jump. When labels are used, the signed offset is automatically calculated by the assembler. The PCN is the location (address) of the next instruction. Figure 12-5 illustrates object code generated by a Jump instruction using the Program Counter Relative addressing mode.

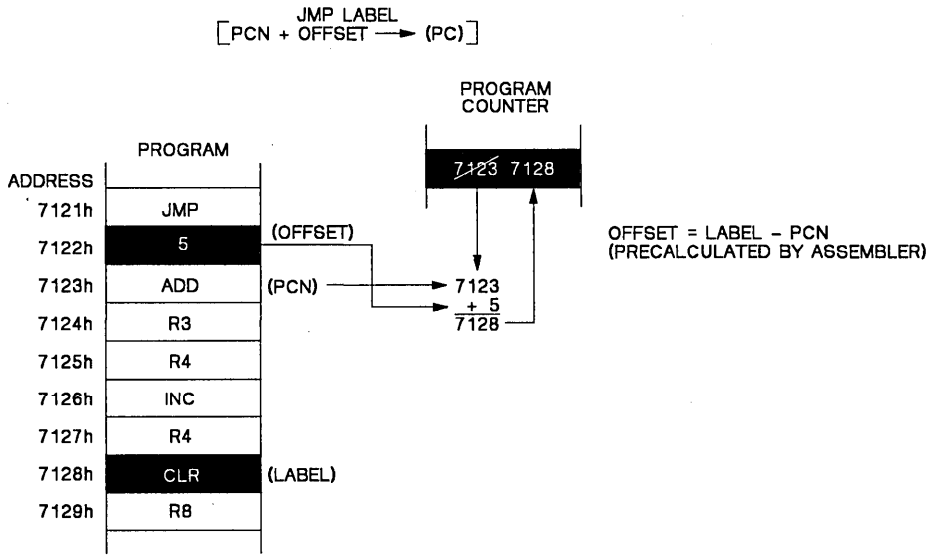


Figure 12-5. Program Counter Relative Addressing Mode

12

12.2.1.6 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. The programmer 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 case, the value of -2 plus the stack pointer equals the address of the data to be moved to register A. Figure 12-6 illustrates this instruction operation.

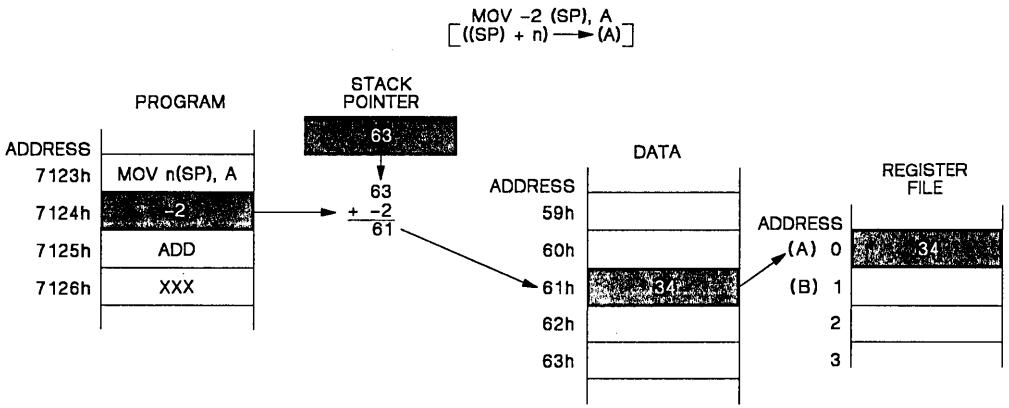


Figure 12-6. Stack Pointer Relative Addressing Mode

12.2.2 Extended Addressing Modes

The Extended Addressing modes provide sophisticated addressing capabilities of arrays, tables, and routine addresses. These modes allow the program to access data from anywhere in the memory. Extended Addressing modes consist of four main types: Direct, Indirect, Indexed, and Offset Indirect. Each of these four types can be subdivided into Absolute and Relative modes for a total of eight Extended addressing modes.

Extended Absolute addressing modes always use register A or the PC as one of the operands in generating a 16-bit address. The Extended Absolute addressing modes are used only by the Branch (BR), CALL, Compare (CMP), and Move (MOV) instructions. The BR and CALL instructions use these modes exclusively.

The Extended Relative addressing modes are similar to the Extended Absolute addressing modes but include the additional step of combining the operand with the program counter (PCN) value before placing the 16-bit address into the program counter. These modes are similar to the Program Counter Relative mode. A 16-bit signed offset is used to calculate the successive instruction address. The successive instruction address is calculated at execution time using the signed 16-bit offset according to the instruction's addressing mode.

The Extended Relative Addressing modes are useful in relocatable code since operation is based on the differences in address position instead of the addresses themselves. This makes the Extended Relative addressing modes well suited for high level languages that often use position independent code. Extended Relative addressing is used by the CALLR and JMPL instructions.

12.2.2.1 Direct Addressing Modes

Direct Addressing mode instructions use an address as the operand. The 16-bit address is written either as 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 for operation as shown in Figure 12-7 below.

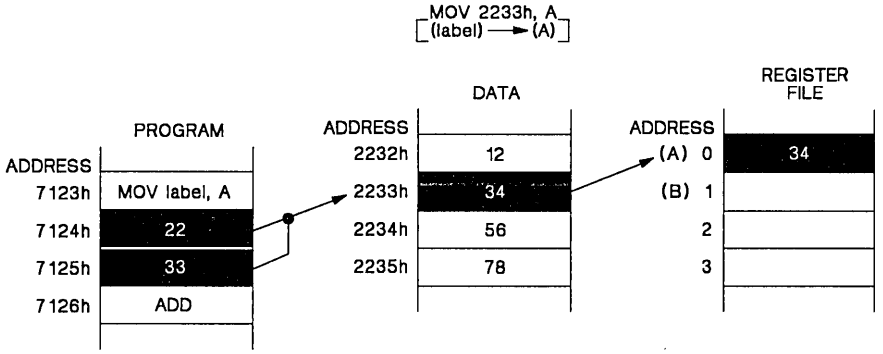


Figure 12-7. Absolute Direct Addressing Mode

The Relative Direct addressing mode (Figure 12-8) adds the address of the next instruction to the 16-bit operand to produce the address of the successive instruction. If a label is used in the instruction, the assembler automatically calculates the offset to use as the operand.

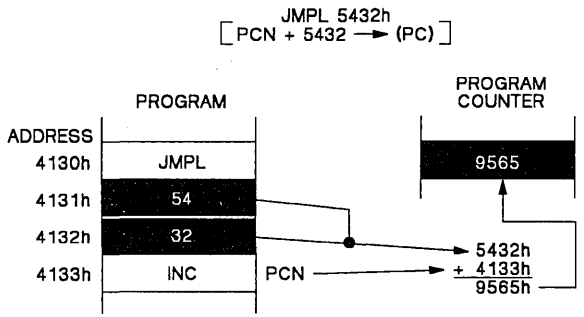
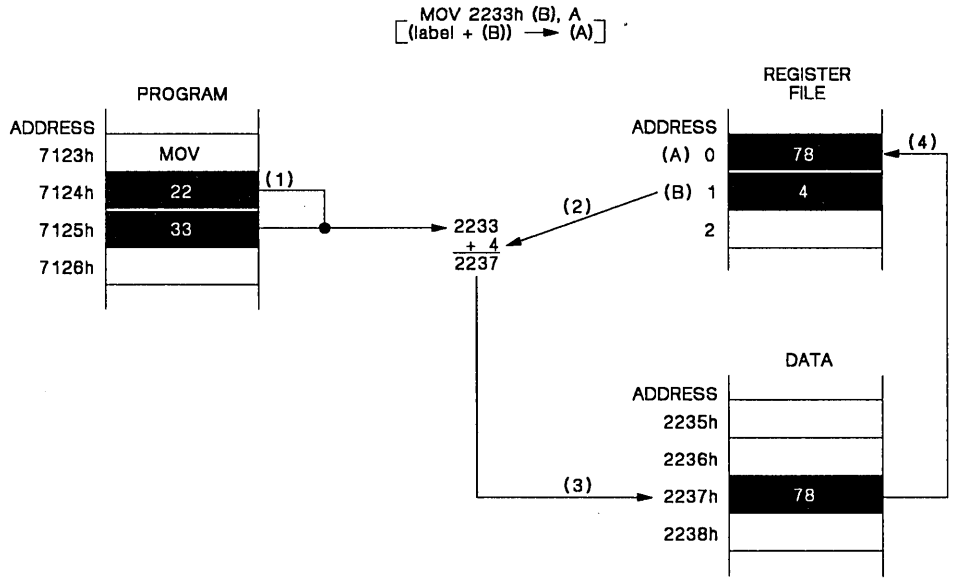


Figure 12-8. Relative Direct Addressing Mode

12.2.2.2 Indexed Addressing Modes

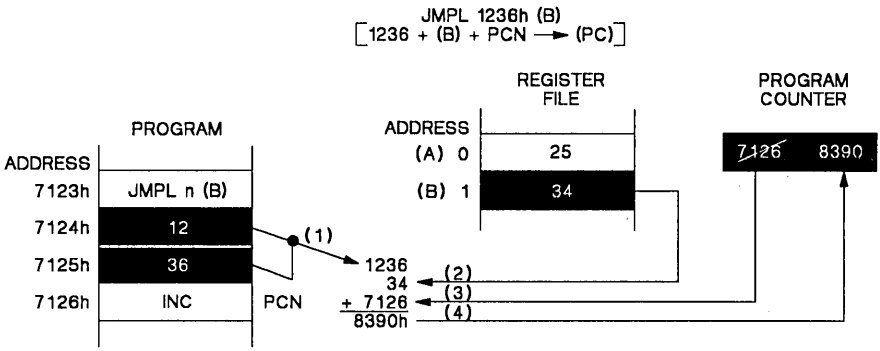
The Absolute Indexed Addressing mode generates a 16-bit address by adding the unsigned contents of the B Register to a 16-bit unsigned constant. The assembly language statement for the Indexed Addressing modes contain the direct memory address written as a 16-bit value or a label, followed by a B in parentheses: MOV 1234(B), or MOV LABEL(B). The MOV and CMP instructions can use Absolute Indexed addressing to easily step through a small table or pick out a particular array value. The instructions CALL and BR can use this mode to execute code based on a decision table and the value in register B. Figure 12-9 illustrates how the object code produced by an instruction using this mode generates a 16-bit effective address.



NOTE: NUMBERS IN PARENTHESIS REPRESENT ORDER OF EXECUTION

Figure 12-9. Absolute Indexed Addressing Mode

The Relative Indexed addressing mode includes the operation described above with the additional following step. The address of the next instruction is added to the sum of register B and the signed 16-bit constant offset, before producing the address of the successive instruction as shown in Figure 12-10.



NOTE: NUMBERS IN PARENTHESIS REPRESENT ORDER OF EXECUTION

Figure 12-10. Relative Indexed Addressing Mode

12.2.2.3 Indirect Addressing Modes

Instructions using the Indirect addressing modes 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 LSB of the address is contained in Rn, and the MSB of the address is contained in the previous register (Rn-1). The TMS370 can use any register pair as an indirect register. Figure 12-11 shows how the Absolute Indirect addressing mode uses the register pair itself in the caculation.

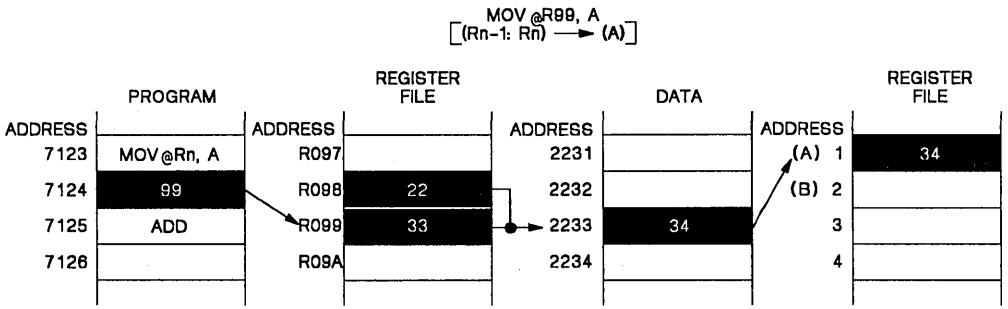


Figure 12-11. Absolute Indirect Addressing Mode

The Relative Indirect addressing mode (Figure 12-12) adds the address of the next instruction to the register pair contents before obtaining the destination address.

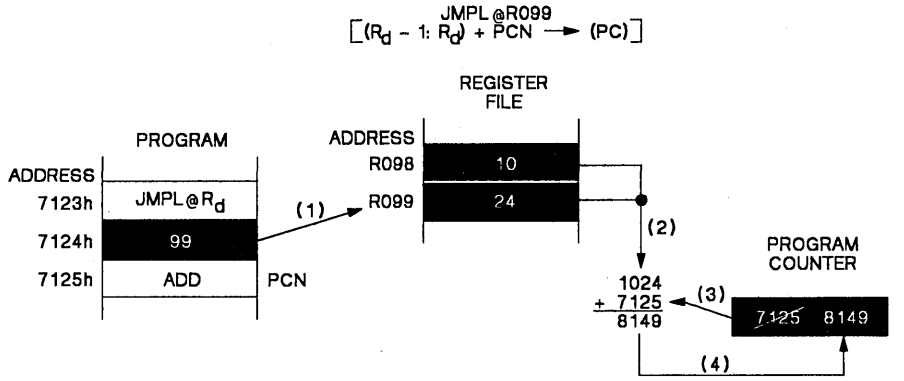
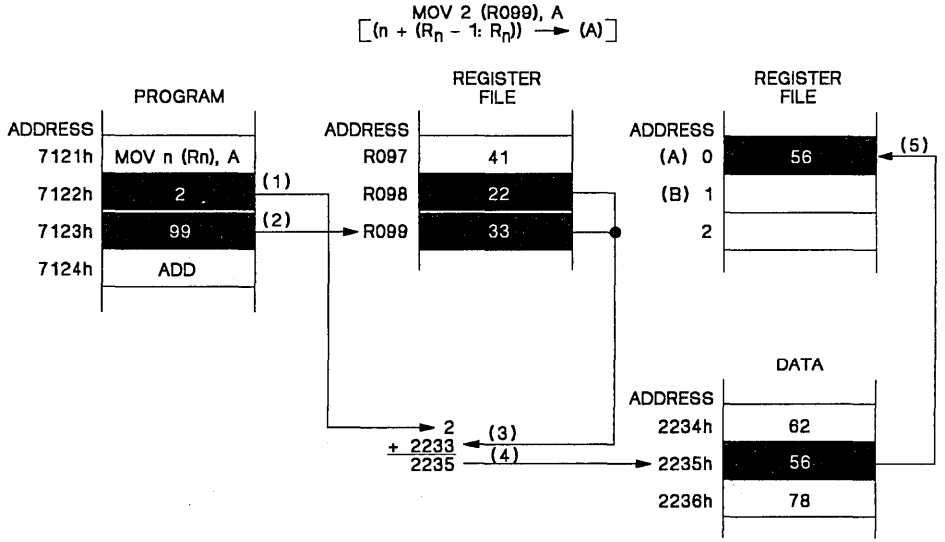


Figure 12-12. Relative Indirect Addressing Mode

12.2.2.4 Offset Indirect Addressing Modes

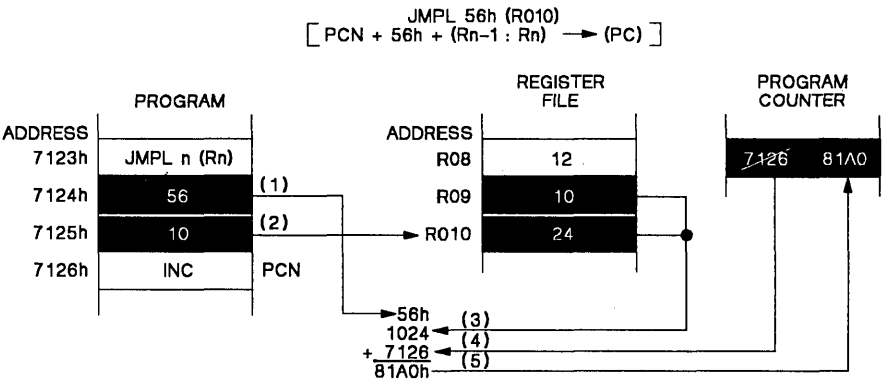
The Offset Indirect addressing modes are similar to the Indirect addressing modes previously described. The Absolute Offset Indirect Addressing mode generates a 16-bit address by adding an 8-bit signed offset to an address taken from a register pair. Offset Indirect addressing is useful in stepping through tables or finding a particular value in a table by using two values to generate the address. Figure 12-13 illustrates how the object code produced by an instruction using the Offset Indirect Addressing mode generates 16-bit effective address.



NOTE: NUMBERS IN PARENTHESIS REPRESENT ORDER OF EXECUTION

Figure 12-13. Absolute Offset 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.



NOTE: NUMBERS IN PARENTHESIS REPRESENT ORDER OF EXECUTION

Figure 12-14. Relative Offset Indirect Addressing Mode

12.2.3 Additional Addressing Modes

There are some cases where the operation of an instruction does not fit into any of the previously described addressing modes. The individual instruction description can be referenced for a list of that instruction's operations.

12.3 Instruction Set Overview

The following tables provide a listing of the instruction set symbols, a listing of the instruction set itself including pertinent characteristics, and an opcode/instruction map.

Table 12-2. TMS370 Symbol Definitions

SYMBOL	DEFINITION	SYMBOL	DEFINITION
A	Register A or R0 in Register File	B	Register B or R1 in Register File
Rn	Register n of Register File	Pn	Register n of Peripheral File ($0 \leq n \leq 255$)
s	Source operand	d/D	Destination operand (8-bit/16-bit)
Rs	Source register in Register File	Ps	Source register in Peripheral File ($0 \leq s \leq 255$)
Rd	Destination register in Register File	Pd	Destination register in Peripheral File ($0 \leq d \leq 255$)
Rps	Source register pair	Rpd	Destination register pair
iop8	8-bit Immediate operand	iop16	16-bit Immediate operand
off8	8-bit Signed Offset	off16	16-bit Signed Offset
Rp	Register pair	label	16-bit label
ST	Status Register	SP	Stack Pointer
PC	Program Counter	PCN	16-bit address of next instruction
#	Immediate operand	@	Indirect addressing operand
MSB	Most significant byte	LSB	Least significant byte
MSb	Most significant bit	LSb	Least significant bit
cnd	Condition	( )	Contents of
→	Is assigned to	←	Becomes equal to
[ ]	Indicates an optional entry. The brackets themselves are not entered.	< >	Indicates an entry that must be typed in. For example, <label> indicates that a label must be entered. The brackets themselves are not entered.
C	Carry flag	N	Sign flag
V	Overflow/borrow flag	Z	Zero flag
XADDR	16-bit address	name	symbol defined for a bit
Rname	symbol defined register bit	Pname	symbol defined peripheral bit

Table 12-3 lists all instruction formats, opcodes, byte lengths, cycles/instruction, operand types, status bits affected, and an operational description.

Table 12-3. TMS370 Family Instruction Overview

MNEMONIC		OPCODE	BYTES	CYCLES t_C	STATUS C N Z V	OPERATION DESCRIPTION
ADC	B,A	69	1	8	x x x x	(s) + (d) + (C) → (d) Add the source, destination, and carry bit together. Store at the destination address.
	Rs,A	19	2	7		
	Rs,B	39	2	7		
	Rs,Rd	49	3	9		
	#iop8,A	29	2	6		
	#iop8,B	59	2	6		
	#iop8,Rd	79	3	8		
ADD	B,A	68	1	8	x x x x	(s) + (d) → (d) Add the source and destination operands at the destination address.
	Rs,A	18	2	7		
	Rs,B	38	2	7		
	Rs,Rd	48	3	9		
	#iop8,A	28	2	6		
	#iop8,B	58	2	6		
	#iop8,Rd	78	3	8		
AND	A,Pd	83	2	9	0 x x 0	(s) AND (d) → (d) AND the source and destination operands together and store at the destination address.
	B,A	63	1	8		
	B,Pd	93	2	9		
	Rs,A	13	2	7		
	Rs,B	33	2	7		
	Rs,Rd	43	3	9		
	#iop8,A	23	2	6		
	#iop8,B	53	2	6		
	#iop8,Rd	73	3	8		
	#iop8,Pd	A3	3	10		
BR	label	8C	3	9	- - - -	XADDR → (PC) Branch to the destination address.
	@Rp	9C	2	8		
	label(B)	AC	3	11		
	off8(Rp)	F4 EC	4	16		
(1)					0 x x 0	If (s) AND (d) ≠ 0, then PCN + offset → (PC) If the AND of the source and destination operands ≠ 0 (corresponding 1 bits) The PC will add the offset, and the jump will be taken.
BTJO	A,Pd,off8	86	3	10		
	B,A,off8	66	2	10		
	B,Pd,off8	96	3	10		
	Rs,A,off8	16	3	9		
	Rs,B,off8	36	3	9		
	Rs,Rd,off8	46	4	11		
	#iop8,A,off8	26	3	8		
	#iop8,B,off8	56	3	8		
	#iop8,Rd,off8	76	4	10		
	#iop8,Pd,off8	A6	4	11		
(1)					0 x x 0	If (s) AND (not d) ≠ 0 then (PCN) + offset → (PC) destination operands ≠ 0 (jump if corresponding 1 and 0 bits). The PC will add the offset and the jump will be taken.
BTJZ	A,Pd,off8	87	3	10		
	B,A,off8	67	2	10		
	B,Pd,off8	97	3	10		
	Rs,A,off8	17	3	9		
	Rs,B,off8	37	3	9		
	Rs,Rd,off8	47	4	11		
	#iop8,A,off8	27	3	8		
	#iop8,B,off8	57	3	8		
	#iop8,Rd,off8	77	4	10		
	#iop8,Pd,off8	A7	4	11		

Note: 1.Add two to cycle count if jump is taken.

Legend:
0 Status Bit always cleared.
1 Status Bit always set.
x Status Bit cleared or set on results.
- Status Bit not affected.

Table 12-3. TMS370 Family Instruction Overview (Continued)

MNEMONIC		OPCODE	BYTES	CYCLES t_C	STATUS C N Z V	OPERATION DESCRIPTION
CALL	label	8E	3	13	- - - -	Push PC MSB, PC LSB,
	@Rp	9E	2	12		XADDR → (SP)
	label(B)	AE	3	15		
	off8(Rp)	F4 EE	4	20		
CALLR	label	8F	3	15	- - - -	Call Relative
	@Rp	9F	2	14		Push PC MSB, PC LSB,
	label(B)	AF	3	17		PCN + (XADDR) → (PC)
	off8(Rp)	F4 EF	4	22		
CLR	A	B5	1	8	0 0 1 0	0 → (d)
	B	C5	1	8		Clear the destination operand.
	Rd	D5	2	6		
CLRC		B0	1	9	0 x x 0	0 → (C)
						Clears the carry bit.
CMP	label,A	8D	3	11	x x x x	Compare; (d) - (s) computed.
	@Rp,A	9D	2	10		Set flags on the result of the source
	label(B),A	AD	3	13		operand subtracted from the destination
	off8(Rp),A	F4 ED	4	18		operand. Operands are not affected by
	off8(SP),A	F3	2	8		operation.
	B,A	6D	1	8		
	Rs,A	1D	2	7		
	Rs,B	3D	2	7		
	Rs,Rd	4D	3	9		
	#iop8,A	2D	2	6		
	#iop8,B	5D	2	6		
	#iop,Rd	7D	3	8		
CMPBIT	Rname	75	3	8	0 x x 0	Complement Bit; invert the bit
	Pname	A5	3	10		
COMPL	A	BB	1	8	x x x 0	Two's complement;
	B	CB	1	8		00h - (s) → (d)
	Rd	DB	2	10		
DAC	B,A	6E	1	10	x x x x	(s) + (d) + (C) → (d) (BCD)
	Rs,A	1E	2	9		The source, destination, and the carry bit
	Rs,B	3E	2	9		are added, and the BCD sum is stored at the
	Rs,Rd	4E	3	11		destination address.
	#iop8,A	2E	2	8		
	#iop8,B	5E	2	8		
	#iop8,Rd	7E	3	10		
DEC	A	B2	1	8	x x x x	(d) - 1 → (d)
	B	C2	1	8		Decrement destination operand by 1.
	Rd	D2	2	6		
DINT		F0 00	2	6	0 0 0 0	0 → (ST)(global interrupt enable bits)
						0 → IE1, 0 → IE2.

Note: Add two to cycle count if jump is taken.

Legend:
0 Status Bit always cleared.
1 Status Bit always set.
x Status Bit cleared or set on results.
- Status Bit not affected.

Table 12-3. TMS370 Family Instruction Overview (Continued)

MNEMONIC	OPCODE	BYTES	CYCLES t_C	STATUS C N Z V	OPERATION DESCRIPTION
DIV Rs,A	F4 F8	3	47-63	0 x x 0	A:B/Rs → A(= quo),B(= REM) Integer divide, 16 by 8 bit.
			14	1 1 1 1	Detected overflow
(1)				- - - -	
DJNZ A,off8	BA	2	10		(d) - 1 → (d);
B,off8	CA	2	10		If (d) ≠ 0, then PCN + offset → (PC)
Rd,off8	DA	3	8		Decrement and jump if not 0.
DSB B,A	6F	1	10	x x x x	(d) - (s) - 1 + (C) → (d) (BCD)
Rs,A	1F	2	9		The source operand is subtracted from the
Rs,B	3F	2	9		destination; this sum is then reduced by 1
Rs,Rd	4F	3	11		and the carry bit is then added to it. The
#iop8,A	2F	2	8		result is stored as a BCD number.
#iop8,B	5F	2	8		
#iop8,Rd	7F	3	10		
EINT	F0 0C	2	6	0 0 0 0	0Ch → (ST)(global interrupt enable bit) 1 → IE1, 1 → IE2.
EINTH	F0 04	2	6	0 0 0 0	04h → (ST)(high priority global interrupt enable bit). 1 → IE1, 0 → IE2
EINTL	F0 08	2	6	0 0 0 0	08h → (ST)(low priority global interrupt enable bit) 0 → IE1, 1 → IE2
IDLE	F6	1	6	- - - -	(PC) → (PC) until interrupt (PC) + 1 → (PC) after return from interrupt Stops µC execution until an interrupt.
INC A	B3	1	8	x x x x	(d) + 1 → (d)
B	C3	1	8		Increase the destination operand by 1.
Rd	D3	2	6		
INCW #off8,Rp	70	3	11	x x x x	(Rp) + offset → (Rp) Add 8-bit signed offset to register pair.
INV A	B4	1	8	0 x x 0	NOT(d) → (d)
B	C4	1	8		1's complement the destination operand.
Rd	D4	2	6		
(1)					
JBITO Rd,off8	77	4	10	0 x x 0	Jump If Bit = 0
Pd,off8	A7	4	11		
(1)					
JBIT1 Rd,off8	76	4	10	0 x x 0	Jump If Bit = 1
Pd,off8	A6	4	11		
JMP off8	00	2	7	- - - -	PCN + off8 → (PC) Jump unconditionally using an 8-bit offset.

Note: 1. Add two to cycle count if jump is taken.

Legend:
0 Status Bit always cleared.
1 Status Bit always set.
x Status Bit cleared or set on results.
- Status Bit not affected.

Table 12-3. TMS370 Family Instruction Overview (Continued)

MNEMONIC	OPCODE	BYTES	CYCLES t_C	STATUS C N Z V	OPERATION DESCRIPTION
JMPL label	89	3	9	- - - -	PCN + D → (PC)
@Rp	99	2	8		Jump unconditionally using
label(B)	A9	3	11		a 16-bit offset
off8(Rp)	F4 E9	4	16		
(1) Jcnd				- - -	- Conditional jump
JC	03	2	5		Carry
JEQ	02	2	5		Jump Equal
JG	0E	2	5		Greater Than, signed
JGE	0D	2	5		Greater Than or Equal, signed
JHS	0B	2	5		Higher or Same, unsigned
JL	09	2	5		Less Than, signed
JLE	0A	2	5		Less Than or Equal, signed
JLO	0F	2	5		Lower Value, unsigned
JN	01	2	5		Negative, signed
JNC	07	2	5		No Carry
JNE	06	2	5		Jump Not Equal
JNV	0C	2	5		No Overflow, signed
JNZ	06	2	5		Not Zero
JP	04	2	5		Positive, signed
JPZ	05	2	5		Positive or Zero, signed
JV	08	2	5		Overflow, signed
JZ	02	2	5		Zero
LDSP	FD	1	7	- - - -	(B) → (SP) Load stack pointer with contents of register B.
LDST #iop8	F0	2	6	x x x x	(d) → (ST) Load ST Register

Note: 1.Add two to cycle count if jump is taken.

Legend:
0 Status Bit always cleared.
1 Status Bit always set.
x Status Bit cleared or set on results.
- Status Bit not affected.

12

Table 12-3. TMS370 Family Instruction Overview (Continued)

MNEMONIC	OPCODE	BYTES	CYCLES t_C	STATUS C N Z V	OPERATION DESCRIPTION
MOV A,B	C0	1	9	0 x x 0	(s) → (d) Replace the destination operand with the source operand.
A,Rd	D0	2	7		
A,Pd	21	2	8		
A,label	8B	3	10		
A,@Rp	9B	2	9		
A,label(B)	AB	3	12		
A,off8(Rp)	F4 EB	4	16		
A,off8(SP)	F2	2	7		
Rs,A	12	2	7		
Rs,B	32	2	7		
label,A	8A	3	10		
@Rp,A	9A	2	9		
label(B),A	AA	3	12		
off8(Rp),A	F4 EA	4	17		
off8(SP),A	F1	2	7		
B,A	62	1	8		
B,Rd	D1	2	7		
B,Pd	51	2	8		
Rs,Rd	42	3	9		
Rs,Pd	71	3	10		
Ps,A	80	2	8		
Ps,B	91	2	8		
Ps,Rd	A2	3	10		
#iop8,A	22	2	6		
#iop8,B	52	2	6		
#iop8,Rd	72	3	8		
#iop8,Pd	F7	3	10		
MOVW Rps,Rpd	98	3	12	0 x x 0	(s) → (Rpd) Copy the source register word to the destination register pair.
#iop16,Rpd	88	4	13		
#iop16(B),Rpd	A8	4	15		
off8(Rs),Rpd	F4 E8	5	20		
MPY B,A	6C	1	47	0 x x 0	(s) x (d) → (A:B) Multiply the source and destination operands, store the result in Registers A (MSB) and B (LSB).
Rs,A	1C	2	46		
Rs,B	3C	2	46		
Rs,Rd	4C	3	48		
#iop8,A	2C	2	45		
#iop8,B	5C	2	45		
#iop8,Rn	7C	3	47		
NOP	FF	1	7	- - - -	No operation
OR A,Pd	84	2	9	0 x x 0	(s) OR (d) → (d) Logically OR the source and destination operands, and store the results at the destination address.
B,A	64	1	8		
B,Pd	94	2	9		
Rs,A	14	2	7		
Rs,B	34	2	7		
Rs,Rd	44	3	9		
#iop8,A	24	2	6		
#iop8,B	54	2	6		
#iop8,Rd	74	3	8		
#iop8,Pd	A4	3	10		

Legend:
0 Status Bit always cleared.
1 Status Bit always set.
x Status Bit cleared or set on result.
- Status Bit not affected.

Table 12-3. TMS370 Family Instruction Overview (Continued)

MNEMONIC		OPCODE	BYTES	CYCLES t_C	STATUS C	N	Z	V	OPERATION DESCRIPTION
POP	A	B9	1	9	0	x	x	0	((SP)) → (d)
	B	C9	1	9					(SP) -1 → (SP)
	Rd	D9	2	7					
	ST	FC	1	8	x	x	x	x	
PUSH	A	B8	1	9	0	x	x	0	(SP) + 1 → (SP)
	B	C8	1	9					(s) → ((SP))
	Rd	D8	2	7					Copy the operand onto the stack.
	ST	FB	1	8	-	-	-	-	Copy the Status Register onto the Stack
RL	A	BE	1	8	x	x	x	0	Bit(n) → Bit(n + 1)
	B	CE	1	8					Bit(7) → Bit(0) and Carry
	Rd	DE	2	6					
RLC	A	BF	1	8	x	x	x	0	Bit(n) → Bit(n + 1)
	B	CF	1	8					Carry → Bit(0)
	Rd	DF	2	6					Bit(7) → Carry
RR	A	BC	1	8	x	x	x	0	Bit(n + 1) → Bit(n)
	B	CC	1	8					Bit(0) → Bit(7) and Carry
	Rd	DC	2	6					
RRC	A	BD	1	8	x	x	x	0	Bit(n + 1) → Bit(n)
	B	CD	1	8					Carry → Bit(7)
	Rd	DD	2	6					Bit(0) → Carry
RTI		FA	1	12	x	x	x	x	Pop PCL, PCH, POP ST
									Return From Interrupt
RTS		F9	1	9	-	-	-	-	Pop PCL, PCH
SBB	B,A	6B	1	8	x	x	x	x	(d) - (s) - 1 + (C) → (d)
	Rs,A	1B	2	7					Subtract with borrow.
	Rs,B	3B	2	7					Destination minus source minus 1 plus
	Rs,Rd	4B	3	9					carry; stored at the destination address.
	#iop8,A	2B	2	6					
	#iop8,B	5B	2	6					
	#iop8,Rd	7B	3	8					
SBIT0	Rd	73	3	8	0	x	x	0	Set Bit to 0
	Pd	A3	3	10					
SBIT1	Rd	74	3	8	0	x	x	0	Set Bit to 1
	Pd	A4	3	10					
SETC		F8	1	7	1	0	1	0	Axh → (ST) Set the carry bit.
									IE1 and IE2 unchanged.

Note: 1.Add two to cycle count if jump is taken.

Legend:
0 Status Bit always cleared.
1 Status Bit always set.
x Status Bit cleared or set on results.
- Status Bit not affected.

Table 12-3. TMS370 Family Instruction Overview (Concluded)

MNEMONIC		OPCODE	BYTES	CYCLES t_C	STATUS C N Z V	OPERATION DESCRIPTION
STSP		FE	1	8	- - - -	(SP) → (B) Copy the SP into Register B.
SUB	B,A	6A	1	8	x x x x	(d) - (s) → (d) Store the destination operand minus the source operand into the destination.
	Rs,A	1A	2	7		
	Rs,B	3A	2	7		
	Rs,Rd	4A	3	9		
	#iop8,A	2A	2	6		
	#iop8,B	5A	2	6		
	#iop8,Rd	7A	3	8		
SWAP	A	B7	1	11	0 x x 0	s(7-4,3-0) → d(3-0,7-4) Swap the operand's hi and lo nibbles.
	B	C7	1	11		
	Rd	D7	2	9		
TRAP	n	EF-E0	1	14	- - - -	Vector n → (PC), n = 0 → 15 Trap to Subroutine; Push PCN Trap 0 = EF
TST	A	B0	1	9	0 x x 0	Test; Set flags from register.
	B	C6	1	10		
XCHB	A	B6	1	10	0 x x 0	(B) ←→ (Rn) Swap the contents of Register B with (Rn).
	B	C6	1	10		
	Rd	D6	2	8		
XOR	A,Pd	85	2	9	0 x x 0	(s) XOR (d) → (d) Logically exclusive OR the source and destination operands, store at the destination address.
	B,A	65	1	8		
	B,Pd	95	2	9		
	Rs,A	15	2	7		
	Rs,B	35	2	7		
	Rs,Rd	45	3	9		
	#iop8,A	25	2	6		
	#iop8,B	55	2	6		
	#iop8,Rd	75	3	8		
	#iop8,Pd	A5	3	10		

Note: 1.Add two to cycle count if jump is taken.

Legend:
0 Status Bit always cleared.
1 Status Bit always set.
x Status Bit cleared or set on results.
- Status Bit not affected.

'Table 12-4 provides an opcode-to-instruction cross reference of all 73 instructions and 245 opcodes of the TMS370 instruction set. To check the instruction of a known opcode, locate the left (high) digit across the top or bottom of thetable, then find the right (low) digit along the side of the table. The intersection contains the instruction mnemonic, operands, and byte/cycle particular to that opcode. Some opcodes, such as B0, are shared by two instructions, in which case both mnemonics are shown along with the byte/cycles count.

Table 12-4. TMS370 Family Opcode/Instruction Map

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	JMP ra 2/7							INCW #n,Rd 3/11	MOV Ps,A 2/8			CLRC TST A 1/9	MOV A,B 1/9	MOV A,Rd 2/7	TRAP 15 1/14	LDST n 2/6
1	JN ra 2/5		MOV A,Pd 2/8			MOV B,Pd 2/8		MOV Rs,Pd 3/10		MOV Ps,B 2/7				MOV B,Rd 2/7	TRAP 14 1/14	MOV n(SP),A 2/7
2	JZ ra 2/5	MOV Rs,A 2/7	MOV #n,A 2/6	MOV Rs,B 2/7	MOV Rs,Rd 3/9	MOV #n,B 2/6	MOV B,A 1/8	MOV #n,Rd 3/8			MOV Ps,Rd 3/10	DEC A 1/8	DEC B 1/8	DEC Rn 2/6	TRAP 13 1/14	MOV A,n(SP) 2/7
3	JC ra 2/5	AND Rs,A 2/7	AND #n,A 2/6	AND Rs,B 2/7	AND Rs,Rd 3/9	AND #n,B 2/6	AND B,A 1/8	AND #n,Rd 3/8	AND A,Pd 2/9	AND B,Pd 2/9	AND #n,Pd 3/10	INC A 1/8	INC B 1/8	INC Rn 2/6	TRAP 12 1/14	CMP n(SP),A 2/8
4	JP ra 2/5	OR Rs,A 2/7	OR #n,A 2/6	OR Rs,B 2/7	OR Rs,Rd 3/9	OR #n,B 2/6	OR B,A 1/8	OR #n,Rd 3/8	OR A,Pd 2/9	OR B,Pd 2/9	OR #n,Pd 3/10	INV A 1/8	INV B 1/8	INV Rn 2/6	TRAP 11 1/14	extend inst,2 opcodes
5	JPZ ra 2/5	XOR Rs,A 2/7	XOR #n,A 2/6	XOR Rs,B 2/7	XOR Rs,Rd 3/9	XOR #n,B 2/6	XOR B,A 1/8	XOR #n,Rd 3/8	XOR A,Pd 2/9	XOR B,Pd 2/9	XOR #n,Pd 3/10	CLR A 1/8	CLR B 1/8	CLR Rn 2/6	TRAP 10 1/14	
6	JNZ ra 2/5	BTJO Rs,A 3/9	BTJO #n,A 3/8	BTJO B,Rd 3/9	BTJO Rs,Rd 4/11	BTJO #n,B 3/8	BTJO B,A 2/10	BTJO #n,Rd 4/10	BTJO A,Pd 3/11	BTJO B,Pd 3/10	BTJO #n,Pd 4/11	XCHB A 1/10	XCHB TESTB 1/10	XCHB Rn 2/8	TRAP 9 1/14	IDLE 1/6
7	JNC ra 2/5	BTJZ Rs,A 3/9	BTJZ #n,A 3/8	BTJZ Rs,B 3/9	BTJZ Rs,Rd 4/11	BTJZ #n,B 3/8	BTJZ B,A 2/10	BTJZ #n,Rd 4/10	BTJZ A,Pd 3/10	BTJZ B,Pd 3/10	BTJZ #n,Pd 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,Rd 4/13	MOVW Rs,Rd 3/12	MOVW #16(B),Rd 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 @Rd 2/8	JMPL lab(B) 3/10	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 @Rs,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,@Rd 2/9	MOV A,lab(B) 3/12	COMPL A 1/8	COMPL B 1/8	COMPL Rn 2/10	TRAP 12 1/14	PUSH ST 1/8
C	JNV ra 2/5	MPY Rs,A 2/46	MPY #n,A 2/45	MPY Rs,B 2/46	MPY Rs,Rd 3/48	MPY #n,B 2/45	MPY B,A 1/47	MPY #n,Rs 3/47	BR lab 3/9	BR @Rd 2/8	BR lab(B) 3/11	RR A 1/8	RR B 1/8	RR Rn 2/6	TRAP 3 1/14	POP ST 1/8
D	JGE ra 2/5	CMP Rs,A 2/7	CMP #n,A 2/6	CMP Rs,B 2/7	CMP Rs,Rd 3/9	CMP #n,B 2/6	CMP B,A 1/8	CMP #n,Rd 3/8	CMP lab,A 3/11	CMP @Rs,A 2/10	CMP lab(B),A 3/13	RRC A 1/8	RRC B 1/8	RRC Rn 2/6	TRAP 2 1/14	LDSP 1/7
E	JG ra 2/5	DAC Rs,A 2/9	DAC #n,A 2/8	DAC Rs,B 2/9	DAC Rs,Rd 3/11	DAC #n,B 2/8	DAC B,A 1/10	DAC #n,Rd 3/10	CALL lab 3/13	CALL @Rd 2/12	CALL lab(B) 3/15	RL A 1/8	RL B 1/8	RL Rn 2/6	TRAP 1 1/14	STSP 1/8
F	JLO ra 2/5	DSB Rs,A 2/9	DSB #n,A 2/8	DSB Rs,B 2/9	DSB Rs,Rd 3/11	DSB #n,B 2/8	DSB B,A 1/10	DSB #n,Rd 3/10	CALLR lab 3/15	CALLR @Rd 2/14	CALLR lab(B) 3/17	RLC A 1/8	RLC B 1/8	RLC Rn 2/6	TRAP 0 1/14	NOP 1/7
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Note:

All conditional jumps (opcodes 01-0F), BTJO, and BTJZ instructions use two additional cycles if the branch is taken. The BTJO and BTJZ instructions have a relative address as the last operand.

Second byte of two-byte instructions (F4xx):

	E	F
8	MOVW n(Rn) 4/15	DIV Rn,A 3/14-63
9	JMPL n(Rn) 4/16	
A	MOV n(Rn),A 4/17	
B	MOV A,n(Rn) 4/16	
C	BR n(Rn) 4/16	
D	CMP n(Rn) 4/18	
E	CALL n(Rn) 4/20	
F	CALLER n(Rn) 4/22	
	E	F

ra - relative address
Rn - Register
Rs - Register containing source byte
Rd - Register containing destination byte
Ps - Peripheral register containing source byte
Pd - Peripheral register containing destination byte
Pn - Peripheral register
n - Immediate 8-bit number
#16 - Immediate 16-bit number
lab - 16-bit label

12.4 Instruction Set Descriptions

The TMS370 instruction set contains 73 instructions covered by 245 unique 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. This is to aid the programmer in understanding the operation associated with the opcode, as well as enhance the readablity of the source code. The following pages contain the individual instruction descriptions. The instructions are in alphabetical order according to the function mnemonic.

Syntax [<label>] ADC <s>,<Rd>

Execution (s) + (Rd) + (C) → (Rd)

Options

inst	operands	bytes	cycles	opcode	operation
ADC	B,A	1	8	69	(B)+(A)+(C) → (A)
ADC	Rs,A	2	7	19	(Rs)+(A)+(C) → (A)
ADC	Rs,B	2	7	39	(Rs)+(B)+(C) → (B)
ADC	Rs,Rd	3	9	49	(Rs)+(Rd)+(C) → (Rd)
ADC	#iop8,A	2	6	29	iop8+(A)+(C) → (A)
ADC	#iop8,B	2	6	59	iop8+(B)+(C) → (B)
ADC	#iop8,Rd	3	8	79	iop8+(Rd)+(C) → (Rd)

Status Bits Affected

C Set to 1 on carry-out of (s) + (Rd) + (C)
Z Set on result
N Set on result
V (C XOR N) AND (source [bit 7] XNOR destination [bit 7])

Description ADC adds the contents of the source, the contents of the destination register, and the carry bit. It stores the result in the destination register.

Adding a 0 to the destination register is equivalent to a conditional increment (increment on carry).

ADC can implement multi-precision addition of signed or unsigned integers. For example, the 16-bit integer in register pair (R2,R3) may be added to the 16-bit integer in (A,B) as follows:

```
        ADD     R3,B    ;Low order bytes added
        ADC     R2,A    ;High order bytes added
```

Examples

```
LABEL1  ADC R66,R117    ;Adds the contents of
                        ;register 66, register
                        ;117, and the carry bit,
                        ;and stores the sum in
                        ;register 117

        ADC   B,A       ;Adds the contents of
                        ;Register B, Register A,
                        ;and the carry bit, and
                        ;stores the sum in
                        ;Register A

        ADC   #03Ch,R29 ;Adds #3Ch, contents of
                        ;register 29, and the
                        ;carry bit, and stores
                        ;the sum in register 29
```

Syntax [<label>] ADD <s>,<Rd>

Execution (s) + (Rd) → (Rd)

Options

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

Status Bits Affected

C Set to 1 on carry-out of (s) + (Rd)
Z Set on result
N Set on result
V (C XOR N) AND (Source [bit 7] XNOR Destination [bit 7])

Description ADD adds two bytes and stores the result in the destination register. It can be used for signed 2's complement or unsigned addition.

Examples

```
LABEL   ADD A,B          ;Adds the contents of
                         ;Registers A and B, stores
                         ;the results in B

        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 [<label>] AND <s>,<Rd>

Execution (s) AND (Rd) → (Rd)

Options

inst	operands	bytes	cycles	opcode	operation
AND	A,Pd	2	9	83	(A) AND (Pd) → (Pd)
AND	B,A	1	8	63	(B) AND (A) → (A)
AND	B,Pd	2	9	93	(B) AND (Pd) → (Pd)
AND	Rs,A	2	7	13	(Rs) AND (A) → (A)
AND	Rs,B	2	7	33	(Rs) AND (B) → (B)
AND	Rs,Rd	3	9	43	(Rs) AND (Rd) → (Rd)
AND	#iop8,A	2	6	23	iop8 AND (A) → (A)
AND	#iop8,B	2	6	53	iop8 AND (B) → (B)
AND	#iop8,Rd	3	8	73	iop8 AND (Rd) → (Rd)
AND	#iop8,Pd	3	10	A3	iop8 AND (Pd) → (Pd)

Status Bits Affected

C ← 0
N Set on result
Z Set on result
V ← 0

Description AND logically ANDs the two 8-bit operands. Each bit in the first operand is ANDed with the corresponding bit in the second operand. This is useful for clearing bits. If you need to clear a bit in the destination operand, then put a 0 in the corresponding source bit. A 1 in a source bit will not change the corresponding destination bit.

Examples

```
LABEL   AND   #01h,R12  ;Clear all bits in R12 except
                        ;Bit 0, which will remain
                        ;unchanged

        AND   R7,A      ;AND the contents of R7 to A
                        ;and store the contents in A

        AND   B,P025    ;AND contents of B to P025,
                        ;store the contents in P025
```

12

Syntax [<label>] 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	(Rn-1:Rn)+off8 → (PC)
BR	@Rp	2	8	9C	(Rn-1:Rn) → (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 @Rn` instruction. An indexed branch instruction of the form `BR TABLE(B)` is an extremely 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. This technique may also be used 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=LSB
      BR   56(R10)     ;(PC) ← 56 + (R9:R10) R10=LSB
```

12

Syntax [<label>] BTJO <s>,<d>,<off8>

Execution If (s) AND (d) ≠ 0, then PCN +off8 → (PC), else PCN → (PC)

Options

inst	operands	bytes	cycles	opcode	Jump If
BTJO	A,Pd,label	3	10/12	86	(A) AND (Pd) ≠ 0
BTJO	B,A,label	2	10/12	66	(B) AND (A) ≠ 0
BTJO	B,Pd,label	3	10/12	96	(B) AND (Pd) ≠ 0
BTJO	Rs,A,label	3	9/11	16	(Rd) AND (A) ≠ 0
BTJO	Rs,B,label	3	9/11	36	(Rd) AND (B) ≠ 0
BTJO	Rs,Rd,label	4	11/13	46	(Rd) AND (Rs) ≠ 0
BTJO	#iop8,A,label	3	8/10	26	(A) AND off8 ≠ 0
BTJO	#iop8,B,label	3	8/10	56	(B) AND off8 ≠ 0
BTJO	#iop8,Rd,label	4	10/12	76	(Rd) AND off8 ≠ 0
BTJO	#iop8,Pd,label	4	11/13	A6	(Pd) AND off8 ≠ 0

Status Bits Affected

C ← 0
N Set on (s) AND (d)
Z Set on (s) AND (d)
V ← 0

Description BTJO jumps if at least one corresponding bit position in the source and destination are both 1. The source operand can be used as a bit mask to test for one or more 1 bits in the specified register. The operands are not changed by this instruction. If one or more corresponding 1 bits are found, the program branches to the offset (refer to the table below).

(s)	(d)	Jump?
00000001	xxxxxxx0	No
00000001	xxxxxxx1	Yes
00000011	xxxxxx00	No
11110000	1000xxxx	Yes
11110000	1001xxxx	Yes

Examples

```
LABEL   BTJO  #014,R4,ISSET ;Jump to ISSET if R4
                            ;(bit 2) or R4  (bit
                            ;4) is a 1

        BTJO  #01,A,LOOP    ;Jump to LOOP if bit 0
                            ;of Register A is a 1

        BTJO R37,R113,START ;Jump to START if any
                            ;1 bit of R113 corre-
                            ;sponds to a 1 bit
                            ;in R37
```

12

Syntax [<label>] 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,label	3	10/12	87	(Pd) AND NOT(A) ≠ 0
BTJZ	B,A,label	2	10/12	67	(A) AND NOT(B) ≠ 0
BTJZ	B,Pd,label	3	10/12	97	(B) AND NOT(Pd) ≠ 0
BTJZ	Rd,A,label	3	9/11	17	(A) AND NOT(Rd) ≠ 0
BTJZ	Rd,B,label	3	9/11	37	(B) AND NOT(Rd) ≠ 0
BTJZ	Rs,Rd,label	4	11/13	47	(Rd) AND NOT(Rs) ≠ 0
BTJZ	#iop8,A,label	3	8/10	27	(A) AND NOT off8 ≠ 0
BTJZ	#iop8,B,label	3	8/10	57	(B) AND NOT off8 ≠ 0
BTJZ	#iop8,Rd,label	4	10/12	77	(Rd) AND NOT off8 ≠ 0
BTJZ	#iop8,Pd,label	4	11/13	A7	(Pd) AND NOT off8 ≠ 0

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 which 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 unchanged by this instruction. The jump is calculated starting from the opcode of the instruction just 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,R23,ZERO    ;If any 1 bits in A
                             ;correspond to 0 bits
                             ;in R23, 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 [<label>] CALL <XADDR>

Execution

```
(SP) + 1    → (SP)
PCN MSB     → ((SP))
(SP) + 1    → (SP)
PCN LSB     → ((SP))
XADDR       →  (PC)
```

(The Stack contains the address of the instruction immediately following the CALL.)

Options

inst	operands	bytes	cycles	opcode	operation
CALL	label	3	13	8E	label → (PC)
CALL	label(B)	3	15	AE	label+(B) → (PC)
CALL	off8(Rd)	4	20	F4 EE	(Rd-1:Rd)+off8 → (PC)
CALL	@Rd	2	12	9E	(Rd-1:Rd) → (PC)

Note : offset = signed 16-bit value
(B) = unsigned 8-bit value
off8 = signed 8-bit value

Status Bits Affected None

Description CALL invokes a subroutine and pushes the PC contents on the stack. The operand indicates the starting address of the subroutine. The extended addressing modes of the CALL instruction allow powerful transfer of control functions.

Examples

```
LABEL   CALL   LABEL4      ;Push PC; (PC) ← LABEL4

        CALL   5432h       ;Push PC; (PC) ← 5432h

        CALL   LABEL5(B);Push PC; (PC) ← LABEL5 + (B)

        CALL   1234h(B)    ;Push PC; (PC) ← 1234h + (B)

        CALL   @R12        ;Push PC; (PC) ← (R11:R12)
                           ;R12=LSB

        CALL   56(R10)     ;Push PC; (PC) ← 56 +
                           ;(R9:R10)   R10=LSB
```

12

Syntax [<label>] CALLR <XADDR>

Execution

(SP) + 1 → (SP)
PCN MSB → ((SP))
(SP) + 1 → (SP)
PCN LSB → ((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	(Rd-1:Rd) + off8 + PCN → (PC)
CALLR @Rd	2	14	9F	(Rd-1:Rd) + 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 uses a value relative to the current program counter (PCN). The extended relative addressing modes of the CALLR instruction allow powerful transfer of control functions. This is useful for relocatable code produced by linkers, compilers or other high language structures. The assembler automatically calculates the correct offset value for the two modes 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=LSB

Offset Indirect Addressing
        CALLR 56(R10)       ;push PC ; (PC) ← PCN
                            ;+ 56 + (R9:R10)
                            ;R10=LSB
```

12

Syntax [<label>] CLR <Rn>

Execution 0 → (Rn)

Options

inst	operands	bytes	cycles	opcode	operation
CLR	A	1	8	B5	0 → (A)
CLR	B	1	8	C5	0 → (B)
CLR	Rn	2	6	D5	0 → (Rn)

Status Bits Affected

C	← 0
N	← 0
Z	← 1
V	← 0

Description CLR clears or initializes to 0 any register including Registers A and B.

Examples

```
LABEL   CLR  B       ;Clear Register B
        CLR  A       ;Clear Register A
        CLR  R105    ;Clear register 105
```

Syntax [<label>] CLRC

Execution Set status bits

Options

inst	operands	bytes	cycles	opcode
CLRC	none	1	9	B0

Status Bits Affected

C ← 0
N Set on value of Register A
Z Set on value of Register A
V ← 0

Description CLRC clears the carry flag. This may be required before an arithmetic or rotate instruction. The logical and move instructions typically clear the carry bit. The CLRC opcode is equivalent to the TST A opcode.

Example

```
LABEL   CLRC        ;Clear the carry bit
```

Syntax [<label>] CMP <s>,<d>

Execution (d) - (s) computed but not stored

Options

inst	operands	bytes	cycles	opcode	operation
General:					
CMP	B,A	1	8	6D	(A)-(B)
CMP	Rs,A	2	7	1D	(A)-(Rs)
CMP	Rs,B	2	7	3D	(B)-(Rs)
CMP	Rs,Rd	3	9	4D	(Rd)-(Rs)
CMP	#iop8,A	2	6	2D	(A)-iop8
CMP	#iop8,B	2	6	5D	(B)-iop8
CMP	#iop8,Rd	3	8	7D	(Rd)-iop8
Extended:					
CMP	label,A	3	11	8D	(A)-(label)
CMP	label(B),A	3	13	AD	(A)-(label+(B))
CMP	off8(Rp),A	4	18	F4 ED	(A)-((Rn-1:Rn)+off8)
CMP	@Rp,A	2	10	9D	(A)-((Rn-1:Rn))
CMP	off8(SP),A	2	8	F3	(A)-((SP)+off8)

Note: Operations are computed but not stored. Status bits are set on results.

Status Bits Affected

C 1 if (d) $\geq$ (s)
N Sign of result
Z 1 if (d) = (s)
V (C XOR N) AND (Source [bit 7] XOR Destination [bit 7])

Description CMP compares the destination operand to the source operand and sets the status bits. The CMP instruction is usually used in conjunction with a Jump instruction. Table 12-5 shows which Jump instructions can be used on status conditions set by CMP execution. There are only seven possible outcomes of the status register after a compare instruction. The jump instructions JC and JHS are equivalent after a compare.

Table 12-5. 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 statusd bits  on result
                       ;of (A) minus (TABLE + (B))
```

Syntax [<label>] CMPBIT <name>

Execution NOT <name> → <name>

Options

inst	operands	bytes	cycles	opcode	operation
CMPBIT	Rname	3	8	75	NOT (bit) → (bit) Reg. bits
CMPBIT	Pname	3	10	A5	NOT (bit) → (bit) Per. bits

Status Bits Affected

C ← 0
N Set on result of (Mask XOR (s))
Z Set on result of (Mask XOR (s))
V ← 0

Description CMPBIT is an assembler constructed instruction that conviently complements the value of the named bit without having to specify a register or mask. This enhances the readability of the software program. The CMPBIT instruction assembles to the instructions XOR #iop8,Rd or XOR #iop8,Pd. The name for the bit is defined by the .DBIT assembler directive.

Examples

```
INT1ENA  .DBIT 7,P017  ;Interrupt 1 bit is now
                       ;named INT1ENA

TEST     .DBIT  4,R33  ;Bit 4 of register 33 is now
                       ;named TEST

LABEL CMPBIT TEST      ;Invert the value of the TEST
                       ;bit.

        CMPBIT INT1ENA ;Change the interrupt 1
                       ;enabled condition.
```

Syntax [<label>] COMPL <Rn>

Execution 0 - Rn → Rn

Options

inst	operands	bytes	cycles	opcode	operation
COMPL	A	1	8	BB	NOT(A)+1 → (A)
COMPL	B	1	8	CB	NOT(B)+1 → (B)
COMPL	Rn	2	6	DB	NOT(Rn)+1 → (Rn)

Status Bits Affected

C	Set on result
N	Set on result
Z	Set on result
V	← 0

Description COMPL provides a logical or 2's complement of the operand. This is the equivalent of an inversion of all the bits followed by an increment. The instruction is useful in doing arithmetic with signed numbers.

Examples

```
LABEL   COMPL  A    ;Complement register A
        COMPL  B    ;Complement register B
        COMPL  R82  ;Complement register 82
```

Syntax [<label>] DAC <s>,<Rd>

Execution (s) + (Rd) + (C) → (Rd), Produces a decimal result

Options

inst	operands	bytes	cycles	opcode	operation
DAC	B,A	1	10	6E	(B)+(A)+(C) → (A)
DAC	Rs,A	2	9	1E	(Rs)+(A)+(C) → (A)
DAC	Rs,B	2	9	3E	(Rs)+(B)+(C) → (B)
DAC	Rs,Rd	3	11	4E	(Rs)+(Rd)+(C) → (Rd)
DAC	#iop8,A	2	8	2E	iop8+(A)+(C) → (A)
DAC	#iop8,B	2	8	5E	iop8+(B)+(C) → (B)
DAC	#iop8,Rd	3	10	7E	iop8+(Rd)+(C) → (Rd)

Status Bits Affected

C 1 if value of (s) + (Rd) + C > 99
N Set on result
Z Set on result
V Undefined

Description DAC adds bytes in binary-coded decimal (BCD) form. Each byte is assumed to contain two BCD digits. DAC is not defined for non-BCD operands. DAC with an immediate operand of zero value is equivalent to a conditional increment of the destination operand (increment destination on carry). The DAC instruction automatically performs a decimal adjust on the binary sum of (s) + (d) + **C**. The carry bit is added to facilitate adding multi-byte BCD strings, and so the carry bit must be cleared before execution of the first DAC instruction.

Examples

```
LABEL  DAC #024h,A    ;If register A contains 097h
                      ;and C = 0,then the final result
                      ;put into A is 021h and the carry
                      ;bit is set

       DAC  R55,R7    ;Add the BCD value of R55,
                      ;and the carry bit to the
                      ;BCD value of R7

       DAC  B,A       ;Add the carry bit to the
                      ;BCD value in Register B
                      ;to Register A
```

12

Syntax [<label>] DEC <Rn>

Execution (Rn) - 1 → (Rn)

Options

inst	operands	bytes	cycles	opcode	operation
DEC	A	1	8	B2	(A)-1 → (A)
DEC	B	1	8	C2	(B)-1 → (B)
DEC	Rn	2	6	D2	(Rn)-1 → (Rn)

Status Bits Affected

C 0 if (Rn) decrements from 00h to FFh; 1 otherwise
N Set on result
Z Set on result
V 1 if (Rn) decrements from 80h to 7Fh; 0 otherwise

Description DEC subtracts 1 from any register. It is useful in 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 [<label>] DINT

Execution 0 → (ST)

Options

inst	operands	bytes	cycles	opcode
DINT	none	2	6	F0 00

Status Bits Affected

C ← 0
N ← 0
Z ← 0
V ← 0
IE1 ← 0
IE2 ← 0

Description DINT simultaneously disables all interrupts. Since the interrupt enable flags are stored in the Status Register, the POP ST or RETI instructions may re-enable interrupts even though a DINT instruction has been executed. During the interrupt service, the interrupt enable bit is automatically cleared after the old Status Register value has been pushed onto the stack. The DINT instruction is equal to the LDST #00 instruction.

Example

```
LABEL   DINT   ;Disable high and low level interrupts.
```

Syntax [<label>] DIV (Rs), A

Execution A:B/(Rs) → A(=quo), B(=rem)

Options

inst	operands	bytes	cycles	opcode	operation
DIV	Rn,A	3	47-63	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 prior to execution and if an overflow is detected, the operands are left unchanged and the status bits C,N,Z, and V are set to 1 and the instruction is aborted. Execution time varies from 47-63 cycles depending on the operands, with an overflow condition taking only 14 cycles.

Example

```
LABEL  DIV  R10,A     ;R10 is divided into the
                      ;A:B register pair (A = MSB)

       JC OVERFLOW    ;Carry is 1 on overflow conditions
```

Syntax [<label>] DJNZ <Rn>,<off8>

Execution (Rn) - 1 → (Rn)
If (Rn) ≠ 0, then PCN + (off8) → (PC), else PCN → (PC)

Type Single Operand

Options

inst	operands	bytes	cycles	opcode	operation
DJNZ	A,LABEL	2	10/12	BA	(A)-1 → (A), jump if (A) ≠ 0
DJNZ	B,LABEL	2	10/12	CA	(B)-1 → (B) jump if (B) ≠ 0
DJNZ	Rn,LABEL	3	8/10	DA	(Rn)-1 →(Rn) jump if (Rn) ≠ 0

Status Bits Affected None

Description DJNZ is used for looping control. It combines the DEC and the JNZ instructions, providing a faster and more compact instruction. DJNZ does not change the status bits.

Examples

```
LABEL   DJNZ  R15,THERE ;Decrement R15. If R15 ≠ 0,
                        ;jump to THERE

        DJNZ  A,AGAIN   ;Decrement A; if A ≠ 0,
                        ;jump to AGAIN

        DJNZ  B,BACK    ;Decrement B; if B ≠ 0,
                        ;jump to BACK
```

Syntax [<label>] DSB <s>,<Rd>

Execution (Rd) - (s) - 1 + (C) → (Rd) (decimal result)

Options

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

Status Bits Affected

C 1 if no borrow required, 0 if borrow required
N Set on result
Z Set on result
V Undefined

Description DSB performs multiprecision BCD subtraction. A 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, so if no borrow in is required, the carry bit should be set to 1. This can be accomplished by executing the SETC instruction. The DSB instruction is undefined for non-BCD operands.

Examples

```
LABEL   DSB   R15,R76     ;R76 minus R15 minus 1 plus
                          ;the carry bit is stored
                          ;in R76

        DSB   A,B         ;Register B minus Register
                          ;A minus 1 plus the carry
                          ;bit is stored in
                          ;Register B

        DSB   #0,R5       ;R5 - 1 → R5, if C = 0
                          ;R5 → R5 if C = 1
```

Syntax [<label>] EINT

Execution 0Ch → (ST)

Options

inst	operands	bytes	cycles	opcode
EINT	none	2	6	F0 0C

Status Bits Affected

C ← 0
N ← 0
Z ← 0
V ← 0
IE1 ← 1
IE2 ← 1

Description EINT simultaneously enables all global interrupts. Since the interrupt enable flags are stored in the Status Register, the POP ST or RETI instructions may disable interrupts even though an EINT instruction has been executed. During the interrupt service, the interrupt enable bit is automatically cleared after the old Status Register value has been pushed onto the stack. Thus, the EINT instruction must be included inside the interrupt service routine to permit nested or multilevel interrupts. This instruction is equivalent to the LDST #00Ch instruction.

Example

```
LABEL  EINT   ;All interrupts are enabled.
```

Syntax [<label>] EINTH

Execution 04h → (ST)

Options

inst	operands	bytes	cycles	opcode
EINTH	none	2	6	F0 04

Status Bits Affected

C ← 0
N ← 0
Z ← 0
V ← 0
IE1 ← 1
IE2 ← 0

Description EINTH is similar to the EINT instruction but enables only high level (1) interrupts and disables low level interrupts. This is equal to the LDST 04h instruction.

Example

```
LABEL  EINTH   ;All level 1 interrupts are enabled.
```

Syntax [<label>] EINTL

Execution 08h → (ST)

Options

inst	operands	bytes	cycles	opcode
EINTL	none	2	6	F0 08

Status Bits Affected

C ← 0
N ← 0
Z ← 0
V ← 0
IE1 ← 0
IE2 ← 1

Description EINTL is similar to the EINT instruction but enables only low level (2) interrupts while disabling high level interrupts. This is equal to the LDST #08h instruction.

Example

```
LABEL   EINTL    ;All level 2 interrupts are enabled.
```

Syntax [<label>] IDLE
(PC) + 1 → (PC) after return from interrupt

Options

inst	operands	bytes	cycles	opcode
IDLE	none	1	6 (min.)	F6

Status Bits Affected None

Description The IDLE instruction causes the device to enter one of three modes. Two of these modes, HALT and STANDBY, use only a fraction of the normal operating power. In STANDBY, the on-chip oscillator remains active. In HALT, the oscillator is off and the chip consumes the least amount of power. Appropriate interrupts must be enabled before entering IDLE. For more information on the low power modes refer to section 4.4.

Example

```
LABEL   IDLE        ;Enter Idle mode and
                    :wait for interrupt
```

Syntax [<label>] INC <Rn>

Execution (Rn) + 1 → (Rn)

Options

inst	operands	bytes	cycles	opcode	operation
INC	A	1	8	B3	(A)+1 → (A)
INC	B	1	8	C3	(B)+1 → (B)
INC	Rd	2	6	D3	(Rn)+1 → (Rn)

Status Bits Affected

C 1 if (Rd) incremented from FFh to 00h; 0 otherwise
N Set on result
Z Set on result
V 1 if (Rn) incremented from 7Fh to 80h; 0 if otherwise

Description INC increments the value of any register. It is useful for incrementing counters.

Examples

```
LABEL   INC   A       ;Increment Register A by 1
        INC   B       ;Increment Register B by 1
        INC   R43     ;Increment Register 43 by 1
```

12

Syntax [<label>] INCW #off8,<Rp>

Execution (Rp) + #off8 → (Rp)

Options

inst	operands	bytes	cycles	opcode	operation
INCW	#off8,Rp	3	11	70	off8+(Rn-1:Rn) → (Rn-1:Rn) off8= signed 8 bit value

Status Bits Affected

C Set to 1 on carry out of off8 + (Rp)
N Set on result
Z Set on result
V (C XOR N) AND (MSBIT off8 XNOR MSBIT (Rd))

Description INCW increments the value of any register pair by the amount specified. The register pair can be incremented by as much as 127 or decremented by as much as 128. This instruction is useful for incrementing counters into large tables. The off8 is sign extended in order to perform 16-bit two's complement addition.

Examples

```
LABEL   INCW #1,R10      ;Increment R10 by 1

        INCW #-1,R10     ;Decrement register R10 by 1

        INCW #100,R255   ;Increment register pair
                         ;R254:R255
```

Syntax [<label>] INV <Rn>

Execution NOT(Rn) → (Rn)

Options

inst	operands	bytes	cycles	opcode	operation
INV	A	1	8	B4	NOT(A) → (A)
INV	B	1	8	C4	NOT(B) → (B)
INV	Rn	2	6	D4	NOT(Rn) → (Rn)

Status Bits Affected

C	← 0
N	Set on result
Z	Set on result
V	← 0

Description INV performs a one's complement of the operand. A one's complement inverts the value of every bit in the register. A two's complement of the operand can be made by following the INV instruction with an increment (INC), or simply using the COMPL instruction.

Examples

```
LABEL   INV  A    ;Invert Register A (0s become 1s,
                  ;1s become 0s)

        INV B     ;Invert Register B

        INV R82   ;Invert register 82
```

Syntax [<label>] JBIT0 <name>, <off8>

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

Options

inst	operands	bytes	cycles	opcode
JBIT0	Rname	4	10	77
JBIT0	Pname	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 conviently jumps to the label if the value of the named bit is zero. This enhances the readability of the software program since the source does not have to specify both the register containing the bit and a mask. The instruction 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 [<label>] JBIT1 <off8>

Execution bit (name) = 1 then PCN + off8 → (PC) else PCN → (PC)

Options

inst	operands	bytes	cycles	opcode	operation
JBIT1	Rname	3	10	76	register bits
JBIT1	Pname	3	12	A6	peripheral bits

Note: Add two cycles if Jump is taken,

Status Bits Affected

C	← 0
N	Set on (s) AND (Rd)
Z	Set on (s) AND (Rd)
V	← 0

Description The JBIT1 is an assembler constructed instruction that conviently jumps to the label if the value of the named bit is one. This instruction enhances the readability of the software program since the source does not have to specify both the register containing the bit and a mask. This instruction assembles to BTJO #iop8,Rd,label or BTJO #iop8,Pd,label. The name for the bit is defined by the .DBIT assembler directive.

Example

```
BUSYP .DBIT 7,P01C        ;Busy bit in PEECTL
                          ;(program EEPROM) is now
                          ;named BUSYP

BIT0  .DBIT 0,R100        ;Bit 0 of register 100 is
                          ;now named BIT0

LABEL BIT1 BIT0, THERE    ;Jump to THERE if bit 0 in
                          ;register 100 is a one.

      JBIT1 BUSYP,HERE    ;Jump to HERE if the program
                          ;EEPROM is busy.
```

12

Syntax [<label>] JMP <off8>

Execution PCN + off8 → (PC)

Options

inst	operands	bytes	cycles	opcode	operation
JMP	off8	2	7	00	PCN+off8 → (PC)

Status Bits Affected None

Description JMP jumps unconditionally to the address specified in the operand. The second byte of the JMP instruction contains the 8-bit relative address of the operand. The operand address must therefore be within -128 to +127 bytes of the location of the instruction following the JMP instruction. The assembler will indicate an error if the target address is beyond -128 to +127 bytes from the next instruction. For a longer jump the BR (branch) or the JMPL instructions can be used.

Example

```
LABEL   JMP THERE       ;Load the PC with the address
                        ;of THERE
```

Syntax [<label>] JMPL <XADDR>

Execution PCN + D → (PC)

Options

inst	operands	bytes	cycles	opcode	operation
JMPL	label	3	9	89	off16+PCN → (PC)
JMPL	label(B)	3	11	A9	off16+(B)+PCN → (PC)
JMPL	off8(Rp)	4	16	F4 E9	(Rn-1:Rn)+off8+PCN → (PC)
JMPL	@Rd	2	8	99	(Rn-1:Rn)+PCN → (PC)

Note: offset = signed 16 bit value
off8 = signed 8 bit value
(B) = unsigned 8 bit value

Status Bits Affected None

Description JMPL is similar to JMP instruction but generates a 16-bit (instead of 8-bit) signed offset to the program counter.

Example

```
LABEL JMPL  LABEL4    ;(PC) ← PCN + offset
                      ;offset=LABEL4-PCN

      JMPL 5432h      ;(PC) ← PCN + 5432h

      JMPL LABEL5(B);(PC) ← PCN + off8 + (B)
                      ;offset=LABEL5 - PCN

      JMPL  1234h(B);(PC) ← PCN + 1234h + (B)

      JMPL  @R12      ;(PC) ← PCN + (R11:R12)
                      ;R12=LSB

      JMPL  56(R10)   ;(PC) ← PCN + 56 + (R9:R10)
                      ;R10=LSB

      JMPL -2(R10)    ;(PC) ← PCN -2 + (R9:R10)
                      ;R10 = LSB
```

12

Syntax [<label>] J<cnd> <off8>

Execution If tested condition is true, (PC) + off8 → (PC), else PCN → (PC)

Status Bits Affected None

Description The J<cnd> 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 may be used to test if the value moved was equal to zero. JN and JPZ may be used in this case to test the sign bit of the value moved. The program may check the overflow bit V after using an arithmetic instruction with the JV or JNV instructions.

Conditional Jump Instructions

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 Non-zero	JNZ	06	X	X	0	X	
Jump if Zero	JZ	02	X	X	1	X	
Jump if Lower	JLO	0F	0	X	0	X	(C = 1) OR (Z = 1)
Jump if Higher or same	JHS	0B	-	X	-	X	
							--Signed Operation--
Jump if Greater	JG	0E	X	-	-	-	Z OR (N XOR V) = 0
Jump if Greater or equal	JGE	0D	X	-	X	-	N XOR V = 0
Jump if Less	JL	09	X	-	X	-	N XOR V = 1
Jump if Less or Equal	JLE	0A	X	-	-	-	Z OR (N XOR V) = 1
Jump if Negative	JN	01	X	1	X	X	
Jump if Positive	JP	04	X	0	0	X	
Jump if Positive or Zero	JPZ	05	X	0	X	X	
Jump if No Overflow	JNV	0C	X	X	X	0	
Jump if Overflow	JV	08	X	X	X	1	

Signed Number Jumps

CONDITION	TRUE	FALSE
d < s	JL	JGE
d ≤ s	JLE	JG
d = s	JEQ	JNE
d ≥ s	JGE	JL
d > s	JG	JLE
Negative	JN	JPZ
Positive	JP	1
Pos or 0	JPZ	JN

Unsigned Number Jumps

CONDITION	TRUE	/FALSE
d < s	JLO	JHS
d ≤ s	2	3
d = s	JEQ	JNE
d ≥ s	JHS	JLO
d > s	3	2

Notes 1.JZ LABEL, JN LABEL
2.JEQ LABEL, JLO LABEL
3.JEQ LABEL, JHS LABEL

12

Status Bit Jumps

BITS	TRUE	FALSE
C	JC	JNC
N	JN	JPZ
Z	JZ	JNZ
V	JV	JNV

Examples

```
LABEL   JNC   TABLE   ;If the carry bit is clear,
                      ;jump to TABLE

        JP    HERE    ;If the negative and zero flags
                      ;are clear, jump to HERE

        JZ    NEXT    ;If the zero flag is set, jump
                      ;to NEXT
```

Syntax [<label>] LDSP

Execution (B) → (SP)

Options

inst	operands	bytes	cycles	opcode
LDSP	none	1	7	FD

Status Bits Affected None

Description LDSP copies the contents of Register B to the Stack Pointer register. Use LDSP to initialize the Stack Pointer.

Example

```
MOV #080h,B     ;Register B = SP value.

LABEL LDSP      ;Copy Register B to the stack
                ;pointer.
```

Syntax [<label>] LDST #iop8

Execution (iop8) → (ST)

Options

inst	operands	bytes	cycles	opcode	operation
LDST	none	2	6	F0	iop8 → (ST)

Status Bits Affected

C	Set on value loaded
N	Set on value loaded
Z	Set on value loaded
V	Set on value loaded
IE1	Set on value loaded
IE2	Set on value loaded

Description The LDST copies the immediate value operand to the Status register. Any combination of bits may be loaded into the status register using this command. Some instructions such as EINT, EINTL, EINTH or DINT are assembled into this instruction.

Example

```
LABEL  LDST #08Ch ;Copy immediate value to
                  ;the Status Register and
                  ;set IE2 bit
```

Syntax [<label>] MOV <s>,<d>

Execution (s) → (d)

Options

inst	operands	bytes	cycles	opcode	operation
REGISTER:					
MOV	A,B	1	9	C0	(A) → (B)
MOV	A,Rd	2	7	D0	(A) → (Rd)
MOV	B,A	1	8	62	(B) → (A)
MOV	B,Rd	2	7	D1	(B) → (Rd)
MOV	Rs,A	2	7	12	(Rs) → (A)
MOV	Rs,B	2	7	32	(Rs) → (B)
MOV	Rs,Rd	3	9	42	(Rs) → (Rd)
MOV	#iop8,A	2	6	22	iop8 → (A)
MOV	#iop8,B	2	6	52	iop8 → (B)
MOV	#iop8,Rd	3	8	72	iop8 → (Rd)
PERIPHERAL:					
MOV	A,Pd	2	8	21	(A) → (Pd)
MOV	B,Pd	2	8	51	(B) → (Pd)
MOV	Rs,Pd	3	10	71	(Rs) → (Pd)
MOV	Ps,A	2	8	80	(Ps) → (A)
MOV	Ps,B	2	8	91	(Ps) → (B)
MOV	Ps,Rd	3	10	A2	(Ps) → (Rd)
MOV	#iop8,Pd	3	10	F7	iop8 → (Pd)
EXTENDED:					
MOV	A,@Rn	2	9	9B	(A) → ((Rn-1:Rn))
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(Rd)	4	16	F4 EB	(A) → (off8+(Rd-1:Rd))
MOV	@Rs,A	2	9	9A	((Rs-1:Rs)) → (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(Rn),A	4	17	F4 EA	(off8 +(Rn-1:Rn)) → (A)

Status Bits Affected

C	← 0
N	Set on value loaded
Z	Set on value loaded
V	← 0

Description MOV transfers values within the memory space. Immediate values may be loaded directly into the registers. In extended addressing modes the processor must use register A. A MOV instruction that uses Register A or B as an operand requires fewer bytes. The MOV Pn,Rn and MOV Rn,Pn instructions have the operands reversed when assembled into machine code.

Examples

```
LABEL   MOV   A,B          ;Move the contents of Register
                           ;A to Register B

        MOV   R32,R105     ;Move the contents of register
                           ;32 to register 105

        MOV   #010h,R3     ;Move #010h to register 3
```

Syntax [<label>] MOVW <s>,<Rd>

Execution (s) → (Rd)

Options

inst	operands	bytes	cycles	opcode	operation
MOVW	#iop16,Rpd	4	13	88	iop16 → (Rd-1:Rd)
MOVW	Rps,Rpd	3	12	98	(Rs-1:Rs) → (Rd-1:Rd)
MOVW	#off8(Rs),Rpd	5	20	F4 E8	(Rs-1:Rs)+off8 → (Rd-1:Rd)
MOVW	#iop16(B),Rpd	4	15	A8	(B) + iop16 → (Rd-1:Rd) ;blank.1

Status Bits Affected

C	← 0
N	Set on MSB moved
Z	Set on MSB 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 LSB of the destination register pair. The source may 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 the B register, and the result 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 = LSB

      MOVW #TAB(B),R3        ;TAB + (B) → (R2:R3)
                             ;R3=LSB

      MOVW #127(R200),R34    ;127 + (R199:R200) →
                             ;(R33:R34)

      MOVW #-128(R200),R34   ;(R199:R200) - 128 →
                             ;(R33:R34)
```

12

Syntax [<label>] MPY <s>,<Rn>

Execution (s) x (Rn) → (A:B) Result always stored in A,B A = MSB

Options

inst	operands	bytes	cycles	opcode	operation
MPY	B,A	1	47	6C	(A) X (B) → (A:B)
MPY	Rs,A	2	46	1C	(A) X (Rs) → (A:B)
MPY	Rs,B	2	46	3C	(B) X (Rs) → (A:B)
MPY	Rs,Rd	3	48	4C	(Rd) X (Rs) → (A:B)
MPY	#iop8,A	2	45	2C	(A) X iop8 → (A:B)
MPY	#iop8,B	2	45	5C	(B) X iop8 → (A:B)
MPY	#iop8,Rd	3	47	7C	(Rd) X iop8 → (A:B)

Status Bits Affected

C ← 0
N Set on MSB of results (Register A)
Z Set on MSB of results (Register A)
V ← 0

Description MPY performs an 8-bit multiply for a general source and destination operand. The 16-bit result is placed in the A, B register pair with the most significant byte in A. Multiplying by a power of two is a convenient means of performing double-byte shifts. If a double byte shift is three places or less, then it may be faster to use RLC or RRC instead of multiply. If a single byte needs shifting then it is almost always faster to use RLC or RRC.

Examples

```
LABEL   MPY   R3,A     ;Multiply (R3) with (A), store
                       ;result in A, B register pair

        MPY #032h,B    ;Multiply 32h with (B), store
                       ;in register pair A, B

        MPY   R12,R7   ;Multiply (R12) with (R7) and
                       ;store in A, B register pair
```

Syntax [<label>] NOP

Execution (PC) + 1 → (PC)

Options

inst	operands	bytes	cycles	opcode
NOP	none	1	7	FF

Status Bits Affected None

Description NOP is useful as a pad instruction during program development, to "patch out" unwanted or erroneous instructions or to leave room for code changes during development. It is also useful in software timing loops.

Example

```
LABEL   NOP
```

Syntax [<label>] OR <s>,<Rd>

Execution (s) OR (Rd) → (Rd)

Options

inst	operands	bytes	cycles	opcode	operation
OR	A,Pd	2	9	84	(A) OR (Pd) → (Pd)
OR	B,Pd	2	9	94	(B) OR (Pd) → (Pd)
OR	B,A	1	8	64	(B) OR (A) → (A)
OR	Rs,A	2	7	14	(Rs) OR (A) → (A)
OR	Rs,B	2	7	34	(Rs) OR (B) → (B)
OR	Rs,Rd	3	9	44	(Rs) OR (Rd) → (Rd)
OR	#iop8,A	2	6	24	iop8 OR (A) → (A)
OR	#iop8,B	2	6	54	iop8 OR (B) → (B)
OR	#iop8,Rd	3	8	74	iop8 OR (Rd) → (Rd)
OR	#iop8,Pd	3	10	A4	iop8 OR (Pd) → (Pd)

Status Bits Affected

C	← 0
N	Set on result
Z	Set on result
V	← 0

Description OR logically ORs the two operands. The OR operation is used to set bits in a register. If a register needs a 1 in the destination then a 1 is placed in the corresponding bit location in the source operand.

Examples

```
LABEL   OR    A,R12   ;OR the A Register with R12,
                      ;store in R12

        OR    #00Fh,A ;Set lower nibble of A to 1s,
                      ;leave upper nibble unchanged

        OR    R8,B    ;OR (R8) with (B), store in B
```

Syntax [<label>] POP <d>

Execution
((SP)) → (d)
(SP) - 1 → (SP)
(Move value then decrement SP)

Options

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

Status Bits Affected

C ← 0
N Set on value POPed
Z Set on value POPed
V ← 0

> **Note:** POP ST affects all status bits.

Description POP pulls a value from the top of the stack. The stack can be used to save or pass values between routines. The Status Register may be replaced with the contents on the stack by the statement `POP ST`. 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 [<label>] PUSH <s>

Execution
(SP) + 1 → (SP)
(s) → ((SP))
(Increment SP then move value)

Options

inst	operands	bytes	cycles	opcode	operation
PUSH	A	1	9	B8	(SP) + 1 → (SP); (A) → ((SP))
PUSH	B	1	9	C8	(SP) + 1 → (SP); (B) → ((SP))
PUSH	Rn	2	7	D8	(SP) + 1 → (SP); (Rn) → ((SP))
PUSH	ST	1	8	FB	(SP) + 1 → (SP); (ST) → ((SP))

Status Bits Affected
C ← 0
N Set on value PUSHed
Z Set on value PUSHed
V ← 0
Note: Status bits are unchanged for PUSH ST

Description PUSH places a value on the top of the stack. The stack is used to save or pass values between routines.

The Status Register may be pushed on the stack with the statement PUSH ST This one-byte instruction is usually executed in conjunction with a subsequently performed POP ST instruction.

Examples

```
LABEL   PUSH  A   ;Move (A) to top of stack
        PUSH  ST  ;Move status to top of stack
```

Syntax [<label>] RL <Rn>

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

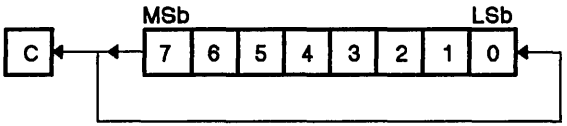
Options

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
```

12

Syntax [<label>] RLC <Rn>

Execution
Bit(n) → Bit(n+1)
Carry → Bit(0)
Bit(7) → Carry

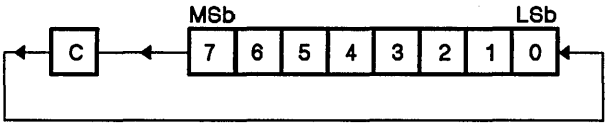
Options

inst	operands	bytes	cycles	opcode
RLC	A	1	8	BF
RLC	B	1	8	CF
RLC	Rn	2	6	DF

Status Bits Affected

C Set to bit 7 of the original operand
N Set on result
Z Set on result
V ← 0

Description RLC circularly shifts the destination contents one bit to the left and through the carry. The original carry bit contents shift into the LSb, and the original MSb shifts into the carry bit.



For example, if Register B contains the value 93h and the carry bit is a zero, then the RLC instruction changes the operand value to 26h and the carry to one.

Rotating left effectively multiplies the value by 2. Using multiple rotates, any power of 2 (2, 4, 8, 16,...) can be achieved. This type of multiply can be faster than the MPY (multiply) instruction. This instruction is also useful in rotates where a value is contained in more than one byte such as an address or in multiplying a large multibyte number by 2. Care must be taken to assure that the carry is at the proper value. The SETC or CLRC instructions may be used to setup the correct value.

Examples

```
LABEL   RLC   R72

        RLC   A

        RLC   B
```

Syntax [<label>] RR <Rn>

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

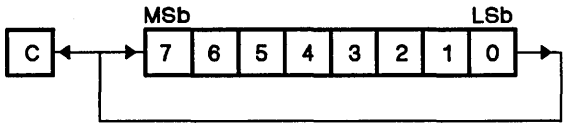
Options

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
```

12

Syntax [<label>] RRC <Rn>

Execution
Bit(n+1) → Bit(n)
Carry → Bit(7)
Bit(0) → Carry

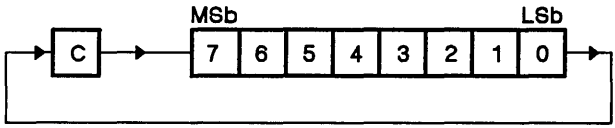
Options

inst	operands	bytes	cycles	opcode
RRC	A	1	8	BD
RRC	B	1	8	CD
RRC	Rn	2	6	DD

Status Bits Affected

C Set to bit 0 of the original value
N Set on result
Z Set on result
V ← 0

Description RRC circularly shifts the destination contents one bit to the right through the carry. The carry bit contents shift into the MSb, and the LSb is shifted into the carry bit.



For example, if Register B contains the value 93h and the carry bit is zero, then RRC changes the operand value to 49h and sets the carry bit.

When the carry is 0 this instruction effectively divides the value by two. A value of 80h becomes 40h. By repetitive use of this instruction, the value can be divided by any power of two. Care must be taken to assure the correct value in the carry bit.

Example

```
LABEL   RRC R32
```

Syntax [<label>] RTI

Execution

((SP)) → (PC LSB)
(SP) - 1 → (SP)
((SP)) → (PC MSB)
(SP) - 1 → (SP)
((SP)) → (ST)
(SP) - 1 → (SP)

Options

inst	operands	bytes	cycles	opcode
RTI	none	1	12	FA

Status Bits Affected Status Register is loaded from the stack

Description RTI is typically the last instruction executed in an interrupt service routine. RTI restores the Status Register to its state 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 POP's and PUSH's 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 [<label>] RTS

Execution

((SP)) → (PC LSB)
(SP) - 1 → (SP)
((SP)) → (PC MSB)
(SP) - 1 → (SP)

Options

inst	operands	bytes	cycles	opcode
RTS	none	1	9	F9

Status Bits Affected None

Description RTS is typically the last instruction executed in a subroutine. RTS branches to the location immediately following the subroutine call instruction. In the called subroutine there must be an equal number of POPs and PUSHes so that the stack is pointing to the return address and not some other data.

Example

```
LABEL  RTS  ;Return to main program from subroutine
```

Syntax [<label>] SBB <s>,<Rd>

Execution (Rd) - (s) - 1 + (C) → (Rd)

Options

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

Status Bits Affected

C Set to 1 if no borrow; 0 otherwise
N Set on result
Z Set on result
V ((C XOR N) AND (Source[Bit 7] XOR Destination[Bit 7]))

Description SBB performs multibyte 2's complement subtraction. An SBB instruction with an immediate operand of zero value is equivalent to a conditional decrement of the destination operand, dependent 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 can be thought of 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 16
        SBB   R2,R20  ;bit numbers. SUB subtracts
                      ;the LSB and the SBB will
                      ;use the carry as a borrow
                      ;during the subtract of
                      ;the MSB.
```

12

Syntax [<label>] SBIT0 <name>

Execution 0 → <name>

Options

inst	operands	bytes	cycles	opcode	operation	
SBIT0	Rname	3	8	73	0 → <bit>	Register bits
SBIT0	Pname	3	10	A3	0 → <bit>	Peripheral bits

Status Bits Affected

C ← 0
N Set on result
Z Set on result
V ← 0

Description SBIT0 is an assembler constructed instruction that conveniently clears the value of the named bit without having to specify a register or mask. This enhances the readability of the software program. This instruction assembles to the instructions AND #iop8,Rd or AND #iop8,Pd. The name for the bit is defined by the .DBIT assembler directive.

Examples

```
INT1ENA   .DBIT 7,P01C   ;The interrupt 1 enable
                         ;bit is now
                         ;named INT1ENA

TEST      .DBIT 4,R33    ;Bit 4 of register 33
                         ;is now named TEST

LABEL     SBIT0 TEST     ;Clears the value of the
                         ;TEST bit

          SBIT0 INT1ENA  ;Disables Interrupt 1
```

Syntax [<label>] SBIT1 <name>

Execution 1 → <name>

Options

inst	operands	bytes	cycles	opcode	operation
SBIT1	Rname	3	8	74	1 → <bit> Register bits
SBIT1	Pname	3	10	A4	1 → <bit> Peripheral bits

Status Bits Affected

C	← 0
N	Set on result
Z	Set on result
V	← 0

Description SBIT1 is an assembler constructed instruction that conveniently sets the value of the named bit without having to specify a register or mask. This enhances the readability of the software program. This instruction assembles to the instructions OR #iop8,Rd" or OR #iop8,Pd. The name for the bit is defined by the .DBIT assembler directive.

Examples

```
INT1ENA   .DBIT 7,P01C   ;The interrupt 1 enable bit
                         ;is now named INT1ENA

TEST      .DBIT 4,R33    ;Bit 4 of register 33 is now
                         ;named TEST

LABEL     SBIT1 TEST     ;Sets the value of the TEST
                         ;bit to 1

          SBIT1 INT1ENA  ;Enables Interrupt 1
```

Syntax [<label>] SETC

Execution 1 → (C)

Options

inst	operands	bytes	cycles	opcode
SETC	none	1	7	F8

Status Bits Affected

C ← 1
N ← 0
Z ← 1
V ← 0

Description SETC sets the carry flag. May be used before an arithmetic or rotate instruction. The IE1 and IE2 enable bits are not affected.

Example

```
LABEL   SETC    ;Set the carry bit in  the status
                ;register
                ;Status register = 0Axh
```

Syntax [<label>] STSP

Execution (SP) → (B)

Options

inst	operands	bytes	cycles	opcode
STSP	none	1	8	FE

Status Bits Affected None

Description STSP copies the contents of the stack pointer to Register B. This instruction can be used to test the stack size. The indexed addressing mode may be used to reference operands on the stack.

Example

```
LABEL   STSP    ;Copy the contents of stack pointer
                ;to Register B
```

Syntax [<label>] SUB <s>,<Rd>

Execution (Rd) - (s) → (Rd)

Options

inst	operands	bytes	cycles	opcode	operation
SUB	B,A	1	8	6A	(A) - (B) → (A)
SUB	Rs,A	2	7	1A	(A) - (Rs) → (A)
SUB	Rs,B	2	7	3A	(B) - (Rs) → (B)
SUB	Rs,Rd	3	9	4A	(Rd) - (Rs) → (Rd)
SUB	#iop8,A	2	6	2A	(A) - iop8 → (A)
SUB	#iop8,B	2	6	5A	(B) - iop8 → (B)
SUB	#iop8,Rd	3	8	7A	(Rd) - iop8 → (Rd)

Status Bits Affected

C Set to 1 if no borrow, otherwise set to 0
N Set on result
Z Set on result
V ((C XOR N) AND (Source[Bit 7] XOR Destination[Bit 7]))

Description SUB performs 2's complement subtraction. The carry bit is set to 0 if a borrow is required. The carry bit could be thought as a "no-borrow" bit in this case.

Examples

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

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

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

Syntax [<label>] SWAP <Rn>

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

Options

inst	operands	bytes	cycles	opcode
SWAP	A	1	11	B7
SWAP	B	1	11	C7
SWAP	Rn	2	9	D7

Status Bits Affected

C Set to bit 4 of original register or Bit 0 of result register
N Set on results
Z Set on results
V ← 0

Description SWAP exchanges the first four bits with the second four bits. This instruction is equivalent to four consecutive RL (rotate left) instructions. It manipulates four bit operands, especially useful for packed BCD operations.

Examples

```
LABEL   SWAP   R45   ;Switch Lo and Hi nibbles of R45

        SWAP   A     ;Switch Lo and Hi nibbles of A

        SWAP   B     ;Switch Lo and Hi nibbles of B
```

Syntax [<label>] TRAP <n> where n = 0 thru 15

Execution

```
(SP) + 1        → (SP)
(PC MSB)        → ((SP))
(SP) + 1        → (SP)
(PC LSB)        → ((SP))
(Entry vector)  → (PC)
```

Options

inst	operands	bytes	cycles	opcode	Entry-vector MSB	LSB
TRAP	0	1	14	EF	7FDE	7FDF
TRAP	1	1	14	EE	7FDC	7FDD
TRAP	2	1	14	ED	7FDA	7FDB
TRAP	3	1	14	EC	7FD8	7FD9
TRAP	4	1	14	EB	7FD6	7FD7
TRAP	5	1	14	EA	7FD4	7FD5
TRAP	6	1	14	E9	7FD2	7FD3
TRAP	7	1	14	E8	7FD0	7FD1
TRAP	8	1	14	E7	7FCE	7FCF
TRAP	9	1	14	E6	7FCC	7FCD
TRAP	10	1	14	E5	7FCA	7FCB
TRAP	11	1	14	E4	7FC8	7FC9
TRAP	12	1	14	E3	7FC6	7FC7
TRAP	13	1	14	E2	7FC4	7FC5
TRAP	14	1	14	E1	7FC2	7FC3
TRAP	15	1	14	E0	7FC0	7FC1

Status Bits Affected None

Description Trap is a one-byte subroutine call. The operand <n> is a trap number that identifies a location in the trap vector table, addresses 07FC0h to 07FDFh in memory. The contents of the two-byte vector location form a 16-bit trap vector to which a subroutine call is performed. The TRAP is more efficient than a CALL when invoking the same routine more than once because less bytes are needed. The subroutine addresses are stored like all other addresses in memory, with the least significant byte in the higher-addressed location, as indicated above.

12

Example

```
LABEL  TRAP 0              ;Execute subroutine at TRAPONE

.sect trap, 07FC0h         ;Define section starting
                           ;at 7FC0h

.word TRAP 15,TRAP 14      ;Define TRAPS 15 AND 14
                           ;subroutine entry points
```

Syntax [<label>] TST < [A],[B]>

Execution C,N,Z,V bits affected

Options

inst	operands	bytes	cycles	opcode
TST	A	1	9	B0
TST	B	1	10	C6

Status Bits Affected

C ← 0
N Set or cleared based on operand
Z Set or cleared based on operand
V ← 0

Description TST sets the status bits according to the value in Register A or B. This allows conditional Jumps on the value in the register.

Example

```
LABEL   TST A     ;Check for zero and negative
                  ;conditions in register A

        TST  B    ;Check for zero and negative
                  ;conditions in register B
```

Syntax [<label>] XCHB <Rn>

Execution (B) ←→ (Rn)

Options

inst	operands	bytes	cycles	opcode	operation
XCHB	A	1	10	B6	(A) ← → (B)
XCHB	B	1	10	C6	(B) ← → (B) (TST B)
XCHB	Rn	2	8	D6	(Rn) ← → (B)

Status Bits Affected

C ← 0
N Set on original contents of B
Z Set on original contents of B
V ← 0

Description XCHB exchanges a register with Register B without going through an intermediate location. The XCHB instruction with the B Register as the operand is equivalent to the TST B instruction.

Examples

```
LABEL   XCHB   A    ;Exchange Register B with
                    ;Register A

        XCHB   R3   ;Exchange Register B with R3
```

Syntax [<label>] 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 be used to complement bits in the destination operand. This operation can also toggle a bit in a register. If the bit value in the destination needs to 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
```

12

13. Design Aids

This section contains sample TMS370 applications to aid the programmer in system development.

This section covers the following topics:

13.1 Microprocessor Interface Example

The following exercise is one method of interfacing the TMS370 family with common memory. The goals of this example are as follows:

- Interface with the maximum amount of memory
- Use the least expensive logic elements
- Use a minimum amount of parts
- Maintain sufficient system speed

The example shown in Figure 13-1 illustrates a balance of these goals. In this case, the TMS370C850 is used with three TMS27C256s to provide 96K bytes of EPROM and two HM6264LP-15s to give 16K of RAM. Peripheral devices using up to 64 bytes of memory space may also interface to the bus. This gives a total memory of 116K; 112K of external memory and 4K memory internal to the microprocessor. The current timings for the EPROM and RAM memory devices are given in Table 13-1. Since specifications change from time to time, always check the latest data sheets for the devices used.

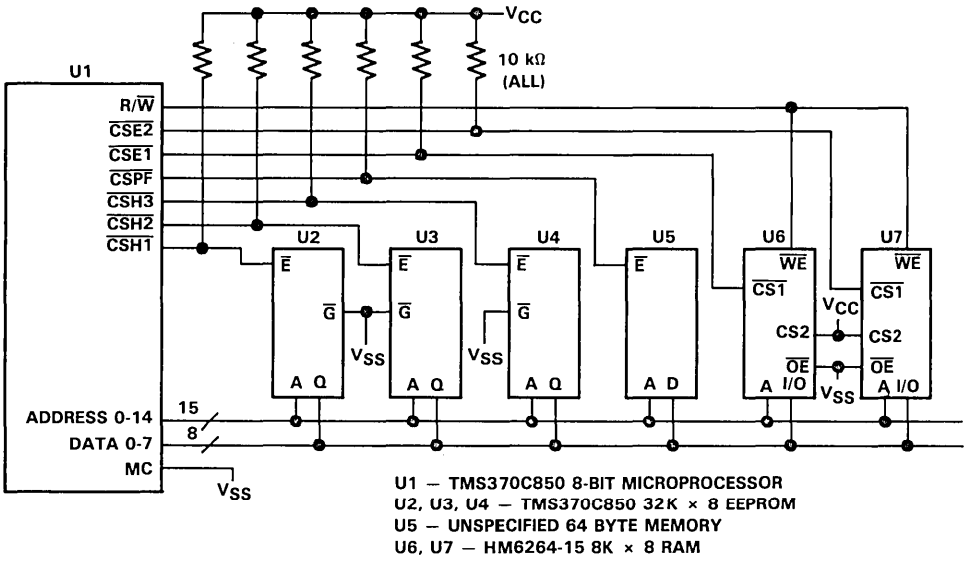


Figure 13-1. Microprocessor Interface Example

13

The devices used in the TMS370/Interface Example Circuit are:

TMS370C850 - 8-bit CMOS microcomputer

TMS27C256 - 32K x 8 EPROM

HM6264LP - Hitachi 8K x 8 RAM

The timing specifications for the TMS27C256-30 EEPROM devices are as follows:

Symbol	Description	Min	Max
$t_a(A)$	Access time from address	--	300 ns
$t_a(E)$	Access time from enable	--	300 ns
t_{dis}	Output disable time	0 ns	105 ns
$t_v(A)$	Output data valid after addr. change	0 ns	--

Reference: 1986 TI MOS Memory Data book

The timing specifications for the HM6264P-15 RAM device are as follows:

Symbol	Description	Min	Max
t_{AA}	Address access time	--	150 ns
t_{OHZ}	Out disable to output in high Z	0	
t_{C01}	Chip selection to output	--	150 ns
t_{HZ1}	Chip Deselection to output in high Z	0 ns	50 ns
t_{CW}	Chip select to end of write	100 ns	--
t_{WP}	Write pulse width	90 ns	--
t_{DW}	Data to write time overlap	60 ns	--
t_{DH}	Data hold from write time	0 ns	--

Reference: #M10 Hitachi Memory Data Book

The TMS370 family is designed to use a clock speed of 20 MHz. This means that slower peripheral devices may not be able to react quick enough to operate properly. The TMS370C050 has the ability to insert Wait states to slow the bus accesses in three different ways. The first way uses the AUTOWAIT DISABLE bit at SCCR1.4 to add 1 wait state to all external accesses. The second way uses the PF AUTOWAIT bit at SCCR0.5 to add 2 wait states to the external peripheral file access in order to accommadate slower devices. The third way allows the external device to pull the $\overline{WAIT}$ pin low and add as many wait states as is required to service the slower device. The table below shows the various combinations.

Table 13-1. Wait State Control Bits

Wait State Control Bits		No. of Clock Cycles per Access	
PF Auto Wait	Autowait Disable	PF File	External Memory
0	0	3	3
0	1	2	2
1	0	4	3
1	1	4	2

Table 13-2. Memory Interface Timing

SYMBOL	PARAMETER	MIN (nS)	MAX (nS)
t_c:†	CLKOUT (system clock)cycle time	200	2000
t_w(COH)	CLKOUT high pulse duration	$.5t_c$	$.5t_c+20$
t_w(COL)	CLKOUT low pulse duration	$.5t_c-20$	$.5t_c$
t_d(COL-A)	Delay time. CLKOUT low to address, R/$\overline{W}$, and $\overline{OCF}$		$.25t_c+40$
t_v(A)	Address valid to $\overline{EDS}$, $\overline{CSE1}$, $\overline{CSE2}$, $\overline{CSH1}$ $\overline{CSH2}$,$\overline{CSH3}$, and $\overline{CSPF}$low	$.5t_c-50$ $.5t_c-50$	
t_{su}(D)	Write data setup time to $\overline{EDS}$ high	$.75t_c-40$‡	
t_h(EH-A)	Address, R/$\overline{W}$, and $\overline{OCF}$ hold time from $\overline{EDS}$, $\overline{CSE1}$, $\overline{CSE2}$, $\overline{CSH1}$,$\overline{CSH2}$, $\overline{CSH3}$, and $\overline{CSPF}$ high	$.5t_c-40$	
t_h(EH-D)W	Write data hold time from $\overline{EDS}$ high	$.75t_c+15$	
t_d(DZ-EL)	Delay time, data bus high impedance to $\overline{EDS}$ low (read cycle)	$.25t_c-30$	
t_d(EH-D)	Delay time,$\overline{EDS}$ high to data bus enable (read cycle)	$1.25t_c-40$	
T_d(EL-DV)	Delay time, $\overline{EDS}$ low to read data valid		t_c-65‡
t_h(EH-D)R	Read data hold time from $\overline{EDS}$ high	0	
t_{su}(WT-COH)	$\overline{WAIT}$ setup time to CLKOUT high	$.25t_c+75$§	
t_h(COH-WT)	$\overline{WAIT}$ hold time from CLKOUT	0	
t_d(EL-WTV)	Delay time, $\overline{EDS}$ low to $\overline{WAIT}$ valid		$.5t_c-70$
t_w	Pulse duration; $\overline{EDS}$, $\overline{CSE1}$, $\overline{CSE2}$, $\overline{CSH1}$, $\overline{CSH2}$, $\overline{CSH3}$, and $\overline{CSPF}$ low	t_c-40‡	t_c+40‡
t_d(AV-DV)R	Delay time, address valid to read data valid		$1.5t_c-75$‡
t_d(AV-WTV)	Delay time, address valid to $\overline{WAIT}$ valid		t_c-85
t_d(AV-EH)	Delay time, address valid to $\overline{EDS}$ high (end of write)	$1.5t_c-40$‡	

Notes: † $t_c$is defined to be $2/f_{OSC}$ and may be referred to as a machine state or simply a state.
‡ If wait state, PF Wait, or Auto-Wait feature is used, add t_c to this value for each wait state invoked.
§ If the Auto-Wait feature is enabled, the $\overline{WAIT}$ input may assume a "Don't Care" condition until the third cycle of the access.

13

The following paragraphs discuss the more important signal timings that need to be considered when 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 here reflect worst case specifications and typical values have been avoided where possible.

13.1.1 Read Cycle Timing

The TMS370 requires a minimum amount of address-to-data access time dependent on the CPU clock speed and the number of wait states used. When interfacing the TMS370 with external memory devices, the following requirements need to be met or incorrect data may be read. These requirements are based on a 20 MHz clock frequency.

13.1.1.1 Valid Address To Data Read Time Requirement

The valid address to data read time requirement is the basic read cycle requirement that must be met by the external device. This requirement is described as the period from the instant the TMS370 outputs a valid address until the TMS370 requires data on the data bus pins. This requirement is variable by using wait states to delay the moment the TMS370 reads data.

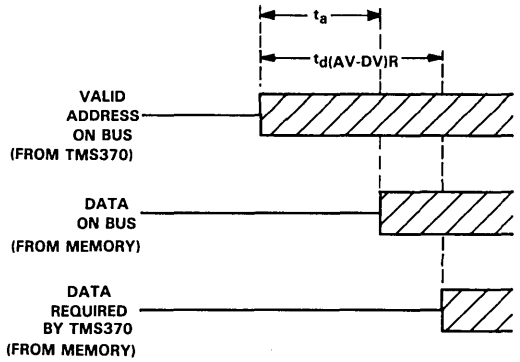


Figure 13-2. Valid Address-To-Data Read Timing

Name	Description	Formula	Time
t_d(AV-DV)R	TMS370 (0 wait) requires data	$1.5t_c$-75	225 ns(too fast)
t_d(AV-DV)R	TMS370 (1 wait) requires data	$2.5t_c$ -75	425 ns(ok)
t_d(AV-DV)R	TMS370 (PF wait) requires data	$3.5t_c$ -75	625 ns(ok)
t_a(A)	TMS27C256-45 provides data		450 ns(too slow)
t_a(A)	TMS27C256-30 provides data		300 ns(ok)
t_{AA}	HM6264-15 provides data		150 ns(ok)

As indicated above, the EPROM (TMS27C256) cannot provide the the data quick enough when the TMS370 device runs at full speed (zero wait states.) Therefore, the TMS370 device should use the Auto-Wait feature (SCC1.4) to add a wait state (one clock cycle) to the timing, in order to slow the bus accesses. The wait state extends the access time (data required by TMS370) until 425 ns, and by that time the EPROM is ready with the data. The Auto-Wait feature allows the TMS370 to be used in low cost applications where cheaper, slower memory devices are to be used. The HM6264-15 can exceed the TMS370's minimum address-to-data setup time with no wait states. The Auto- Wait feature may be turned off when accessing external RAM comparable to the Hitachi device to speed system throughput.

A peripheral device may have up to 625 ns to respond to the TMS370 if the Peripheral Wait states are enabled. If the extra wait states are not needed, the TMS370 treats the peripheral device like other memory.

13.1.1.2 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 ($\overline{CSxx}$ or $\overline{EDS}$) signal(s) must be used with external memory to validate the memory cycle. Connnecting the Chip Select (CSxx) pin of the TMS370 to the EPROM's enable ($\overline{E}$) pin allows 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 operating mode at any one time. The HM6264 also enters a low-power standby mode whenever the $\overline{CS1}$ pin is used with a TMS370 chip select pin.

Name	Description	Formula	Time
t_d(EL-DV)	TMS370 (0 wait) requires data	t_c-65	135 ns(too fast)
t_d(EL-DV)	TMS370 (1 wait) requires data	$2t_c$-65	335 ns(ok)
t_d(EL-DV)	TMS370 (pf wait) requires data	$3t_c$-65	535 ns(ok)
t_a(E)	TMS27C256-30 provides data		300 ns(ok)
t_{C01}	HM6264-15 provides data		150 ns(ok)

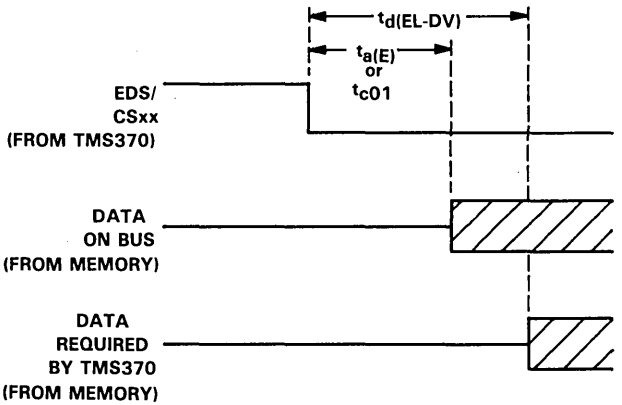


Figure 13-3. Chip Select Low To Data Read Timing

13.1.1.3 Chip Select High To Next Data Bus Drive Requirements

The TMS370 and the Memory device should not drive the bus at the same time. This can lead to increased stress and noise spiking on the Vcc and Vss lines, reducing the reliability of the device. Memory devices often continue to drive the bus for a short time after the chip select signal goes high. This normally doesn't present a problem unless the chip select signal is delayed by interface circuitry and the data is not. If the chip select high transition is delayed long enough (and the data is not), the TMS370 will have initiated a write cycle while the memory is still providing (reading) data.

Name	Description	Formula	Time
$t_{d(EH-D)}$	TMS370 (all) drives bus	$1.25t_C$-40	210 ns
t_{dis}	TMS27C256-45 releases bus		130 ns
t_{dis}	TMS27C256-30 releases bus		105 ns
t_{OHZ}	HM6264-15 releases bus		50 ns

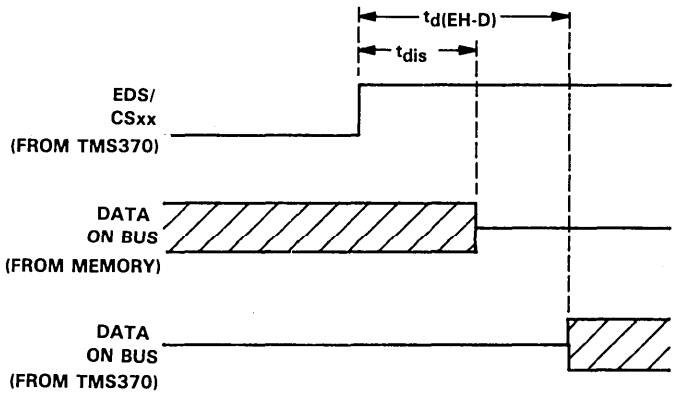


Figure 13-4. Chip Select High To Next Data Bus Drive Timing

13.1.1.4 Read Data Hold After Chip Select High Requirements

13

The high transition of the Chip Select signal indicates the end of a data transfer (in this case, a Read) cycle. The memory device must provide data up to this point, otherwise, incorrect data may be read. Most memories will continue to hold (or drive) the data bus for a short time after they are deselected, althought the data may or may not be valid. After that period, the memories put their data outputs into the high-impedence state.

Name	Decsription	Formula	Time
$t_{d(EH-D)R}$	TMS370 (all) needs data	-	0 ns
$t_V(A)$	TMS27C256-30 data	-	0 ns
$t_{HZ}1$	HM6264-15 holds data	-	0 ns

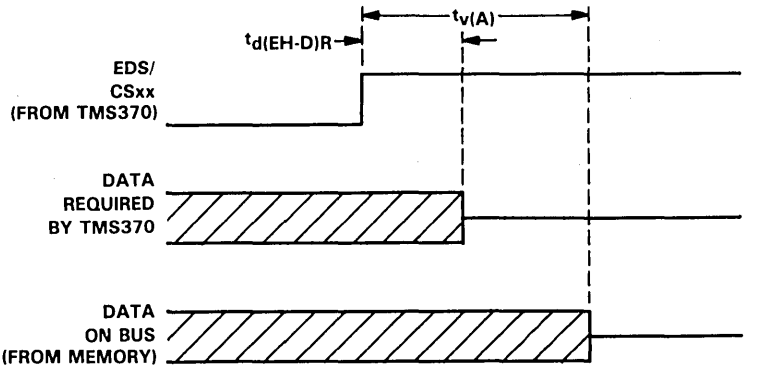


Figure 13-5. Read Data Hold After Chip Select High Timing

13.1.2 Write Cycle Timing

The write cycle timing is defined primarily by the characteristics of the RAM interfacing with the TMS370. The Hitachi HM6264 used in the example offers two types of write cycles and this application uses the type where the output enable ($\overline{OE}$) pin is always fixed low. With the CS2 pin tied to Vcc, the $\overline{CS1}$ and R/$\overline{W}$ signals determine the read and write cycle boundaries. A seperate address decoder may be used instead of the chip select functions, but the $\overline{EDS}$ must be used to validate the memory cycle. The $\overline{EDS}$ signal has the same timing as the chip select signals. Figure 13-6 shows the write cycle parameters that need to be met and are discussed in the following paragraphs.

Name	Description	Formula	Time
t_W	TMS370 (no wait) pulse width provided	t_C-40	160 ns
t_W	TMS370 (pf wait) pulse width provided	$3t_C$-40	560 ns
t_{CW}	HM6264-15 pulse width required		100 ns

13.1.2.1 Write Data Setup Time Requirements

The write data setup time is the period the RAM needs to receive data before the chip select signal goes high (inactive).

Name	Description	Formula	Time
t_{SU}(D)	TMS370 (no wait) provides data	.75t_C-40	110 ns
t_{SU}(D)	TMS370 (pf wait) provides data	2.75t_C-40	510 ns
t_{DW}	HM6264-15 requires data		60 ns

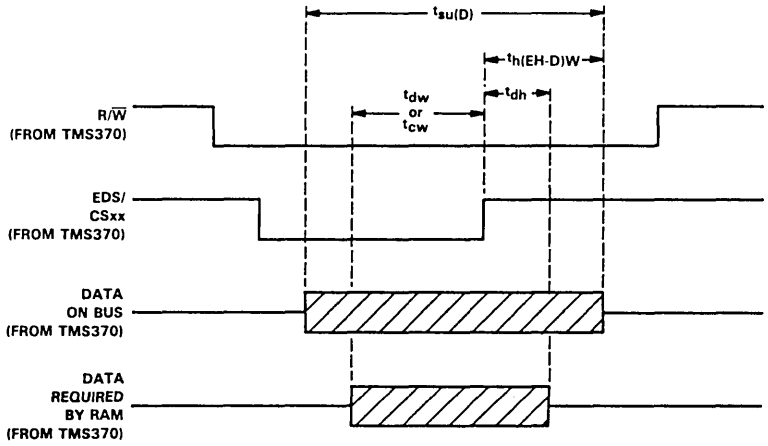


Figure 13-6. Write Data Setup Timing

In the interface example the TMS370, even with no wait state, satisfies the HM6264-15 RAM's setup requirement. In a system design where bus tranceivers have been added, however, setup timing becomes more important.

13.1.2.2 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 RAM 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 RAM.

Name	Description	Formula	Time
$t_{h(EH-D)W}$	TMS370 (all) provides data	$.75t_C+15$	165 ns
t_{DH}	HM6264-15 requires data		0 ns

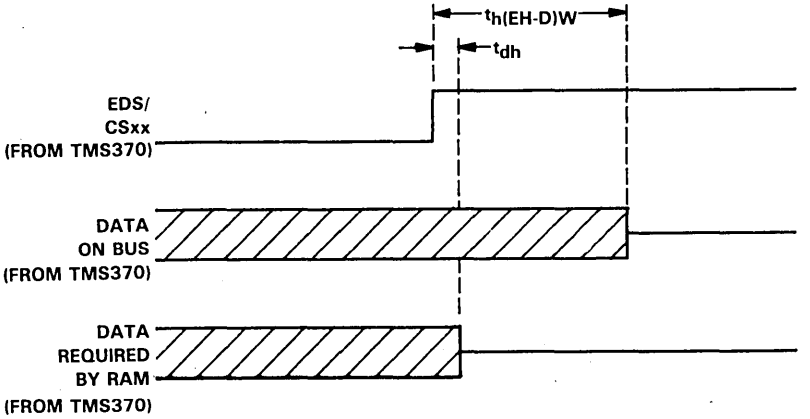


Figure 13-7. Write Data Hold After Chip Select High

13.1.3 Design Options

The interface example ilustrated in Figure 13-1 shows a compromise of system speed and cost. As with all projects, a priority of design goals must be established. Below are some suggestions for optimizing a system toward these goals.

13.1.3.1 Lower Cost

If system cost becomes more important, then slower ROMs which are less expensive should be used. The slowest EPROM for this device is the TMS27C256-45 with 450 ns access time. However, even with one wait state the TMS370 needs data before this EPROM can supply it. A 19 MHz or lower crystal oscillator solves the problem by extending the clock cycle time. The EPROM's $\overline{E}$ pin can no longer be used as enable strobe because of the slower response time. The system must use the EPROM's $\overline{G}$ pin which provides sufficient time.

A designer still desiring the low power standby mode needs to connect the $\overline{E}$ pins of all of the EPROM's to one or more general purpose I/O pins on the TMS370. Software can then turn off the EPROMs when not in use. Since the RAMs have no trouble meeting the requirements of a 20 MHz clock, a slower crystal speed presents no problem.

A. Access time from address to valid data (@ 19 MHz, t_c=210.5)

TMS370 (1 wait) requires data	t_D(AV-DV)R	2.5t $_C$-75	451 ns
TMS27C256-45 provide data	t_A(A)		450 ns (ok)

B. Access time from enable low to valid data (@ 19 MHz, tc=210.5)

TMS370 (1 wait) requires	td(EL-DV)	2t_C-65	335 ns

TMS27C256-45 provides data	$t_A(\overline{E})$	$\overline{E}$ pin	450 ns(not ok)
TMS27C256-45 provides data	$t_{EN}(\overline{G})$	$\overline{G}$ pin	135 ns(ok)

13.1.3.2 Faster Speed

If the main objective is system speed, then the slowest EPROM that will work with the TMS370 running without wait states should be used. The TMS370 at 20 MHz has a read access time requirement of 225 ns, therefore the TMS27C256-20 EPROM which provides data in 200 ns should be used. As with the low cost suggestions above, the EPROM's $\overline{E}$ pin is not fast enough to use the chip select strobe. The EPROM's $\overline{G}$ pin should be used instead. To get a low power standby mode with the EPROMs, use general purpose output lines from the TMS370 to the EPROM's $\overline{E}$ pin. The pins should be software enabled before entering the EPROM's program.

A. Access time from address to valid data:

TMS370 (no wait) requires data	t_D(AV-DV)R	$1.5t_c$-75	225 ns
TMS27C256-20 provide data	t_A(A)		200 ns (ok)

B. Access time from enable low to valid data:

TMS370 (no wait) requires	t_D(EL-DV)	t_c-65	135 ns
TMS27C256-20 provides data	$t_A(\overline{E})$	$\overline{E}$ pin	200 ns (not ok)
TMS27C256-20 provides data	$t_{EN}(\overline{G})$	$\overline{G}$pin	75 ns (ok)

13.1.4 Software Examples For Bank Switching

The following programs show how memory bank switching can be used by the circuit in Figure 13-1. Memory bank switching allows two or more memory devices to share the same addresses. The programmable chip select ($\overline{CSHx}$, $\overline{CSEx}$, and $\overline{CSPF}$) signals are used to enable the memory devices or "banks" one at a time during a read or write cycle.

In the interface example, the three EPROM devices share addresses 8000h though FFFFh. Only one EPROM device (or bank), selected by either $\overline{CSH1}$, $\overline{CSH2}$, or $\overline{CSH3}$, will be reading data at any one time. The two RAM devices are each mapped at addresses 2000h through 3FFFh. The write and read cycles involve one RAM device at a time as determined by the $\overline{CSE1}$ and $\overline{CSE2}$ signals. The $\overline{CSPF}$ 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, the TMS370Cx50 must be configured for the microcomputer mode so that the chip select signals are available for use. The external memory devices must have 3-state outputs, since these devices share the data bus.

13.1.4.1 Initialize to EPROM/RAM Bank 1

The following program initializes the ports to use bank 1 of the EPROM and the RAM as used in Figure 13-1. The TMS370 must be in the microcomputer mode since 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
                          ;auto-wait cycles
      AND   #0EFh,P011    ;Enable General memory wait
                          ;cycles (default condition
                          ;after reset)
      MOV   #0FFh,P021    ;Set Port A up as a Data Bus
      MOV   #0FFh,P025    ;Set Port B up as the Low
                          ;Address bus
      MOV   #07Fh,P029    ;Set Port C 0-6 up as the High
                          ;address bus
      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.
```

13.1.4.2 Changing to EPROM Bank 2

The following program illustrates how to change the EPROM bank while leaving the RAM banks unaffected. In this example, the program runs out of internal memory and disables all EPROM banks and then enables EPROM bank 2 for use. For this reason, the program must not reside in an EPROM. The program could test various EPROM bank 2 memory locations before executing the branch instruction in order to verify that EPROM bank 2 exists within the system.

```
AND   #0B9h,P02D    ;Disable all EPROM banks (cannot
                    ;be done in EPROM banks.
OR    #004h,P02D    ;Enable EPROM bank 2. When turned off,
                    ;pin outputs a 1 because of the
BR    ROM2          ;initial setup above, could be done
                    ;in 1 instruction if conditions of
                    ;other chip selects was known.
```

13.1.4.3 Change To EPROM Bank 3 and RAM Bank 2

This routine provides switching from one EPROM bank to another while operating from an EPROM bank. Only one instruction (in EPROM bank 2) is needed. The code within the EPROM banks must be synchronized, and the instruction at the address after the move instruction must be a valid instruction within the new EPROM bank.

```
GOROM3  MOV  #003h,P02D ;Enable ROM bank 3 and RAM bank 2.
ROM3                    ;This address must be the same
                        ;as the beginning routine address
                        ;in bank 3 if executing from EPROM.
```

13.1.4.4 Change RAM Banks

This method shows how to change RAM banks without affecting the execution out of the current EPROM bank. The RAM banks are selected and deselected the same as EPROM banks. When changing RAM or EPROM banks, the software must insure that only one bank is selected at any one time. This example disables the $\overline{CSE1}$ and $\overline{CSE2}$ signals and enables the $\overline{CSE2}$ signal.

```
AND   #07Eh,P02D  ;Turn off all RAM banks (execute
                  ;from EPROM or on chip)
 OR   #001h,P02D  ;Turn on  RAM bank 2. When turned off,
                  ;pin outputs a 1 because of the
                  ;initial set-up above
```

13.2 Programming with the TMS370 Family

The following example demonstrates the self-programming ability of the TMS370 family. This feature can be used to program any byte of the onboard data and program EEPROM by passing the appropriate data and address to this routine.

This program consists of 3 major sections: the procedure that loads the core program into RAM (RAMJAM), the procedure that determines the bits that need to be changed (PROGRAM), and the procedure that changes these bits (RAMPROG).

RAMJAM is a routine that moves the 19 byte core programming routine into RAM starting at address 20h in the Register area. It only needs to run once during the initialization portion of the program.

PROGRAM attempts to save programming time by checking which portions of the 2 step programming procedure have to occur. If the data already in the array is the same as the new wanted data then no programming need occur. If the program can omit a 'write ones' or a 'write zeros' operation then 10 ms is removed from the total 20 ms programming time. Every programming step that this routine omits saves 10 ms.

RAMPROG is the RAM resident routine which initiates, times and then stops the actual EEPROM programming. During this section of code the interrupts should be disabled to avoid having to use the Program memory. All program memory is disabled while programming the program EEPROM so neither a routine execution or interrupt vector access can occur during the program cy-

cle. RAMPROG resides in RAM because it needs to program both Data and Program EEPROM for this general purpose example.

All read and write access to an entire EEPROM array are disabled while any one byte in the array is being programmed. This means that a program cannot execute out of Program EEPROM while programming it. Likewise the program cannot execute out of Data EEPROM if it is being programmed. The only other place to locate the core routine which does the actual programming is in the RAM. This general purpose core program takes only 19 bytes of RAM and can program both the data and program EEPROM arrays. This core routine could reside in program memory if only data EEPROM needed programming and vise versa.

Unprotected data EEPROM may be programmed using only the Vcc power supply. Enter the WPO mode by placing 12V on the MC pin when programming program EEPROM or protected data EEPROM.

The Program EEPROM array cannot be used while it is programming so the actual program code must reside in other memory, the most general being RAM. This section resides in the initialization routine and loads the code to program EEPROM into the RAM. (if only Data EEPROM needs programming the RAMPROG code can reside in regular ROM and the RAMJAM section removed.)

```
TEMP1      .EQU R3                ;general purpose 16 bit
                                  ;register
ETYPE      .EQU R7                ;EEPROM array type 0= data,
                                  ;2=program
EECTL      .EQU 101Ah             ;index address for eeprom
                                  ;control register
RAMJAM     MOV  #19,B             ;Transfer 19 bytes
FILLRAM    MOV  @RAMPROG-1(B),A   ;Move small program from ROM
                                  ;into RAM starting at 20h
           MOV  A,@20h-1(B)       ;
           DJNZ B,FILLRAM         ;fill RAM
           JMP  MOREINIT          ;Goto more initialization
                                  ;program
```

The processor must be in single chip mode for correct operation during this core routine.

```
                                  ;A= EECTL value  xx/blk/
                                  ;ones/execute
                                  ;ETYPE = EEPROM array type 0=Data
                                  ;2=Program
                                  ;Routine's real address is 20h,
                                  ;EEPROG=20h
RAMPROG    MOV   ETYPE,B          ;EECTL index to B (data=0,
                                  ;program =2)
           MOV   A,EECTL(B)       ;Load proper EECTL register
                                  ;Wait 10 ms for eeprom write
           MOVW  #2778,TEMP1
WAIT10     INCW  #-1,TEMP1        ;11cy: (18 cycles *.2 us/cycle)
                                  ; * 2778 = 10 ms
           JC    WAIT10           ;7cy:   ( at 20MHz)
           CLR   A                ;stop programming pulse
           MOV   A,EECTL(B)       ;clear out EECTL
EXITRAM    RTS                    ;exit from internal Ram program
                                  ;19 bytes total
```

The following program is used to write to any location in the data or program EEPROM.

Parameters used: ADDR1 = EEPROM address; A= data to write to eeprom array

ETYPE = 0= data eeprom; 2= program eeprom
ETYPE set before entering this routine

```
TEMP2    .EQU  R4                  ;general purpose temporary
                                   ;register
ADDR1    .EQU   R6                 ;contains address for
                                   ;program operations
EEPROG   .EQU   20h                ;starting address of RAM
                                   ;code which programs eeprom.
PROGRAM  CLR     ETYPE             ;initialize eeprom type to
                                   ;Data EERPOM
         CMP     #01Fh,ADDR1-1     ;Data EEPROM resides at
                                   ;1F00h to 1FFFh
         JEQ     ISSAME            ;
         MOV     #2,ETYPE          ;Set to Program EEPROM type
ISSAME   MOV     A,TEMP2           ;save data
         MOV     @ADDR1,A          ;read current data
         XOR     TEMP2,A           ;different bits=1
         JZ      EXITW             ;if byte is already equal
                                   ;then exit
         INV     A                 ;different bits=0
         OR      TEMP2,A           ;bits that change from
                                   ;1 to 0 = 0
         BTJZ    #0FFh,A,WRITE0    ;If any 0s, then must
                                   ;program the zero's
         JMP     ONES              ;If all 1s, advance to
                                   ;program 1s part
WRITE0   DINT                      ;No interruptions
         MOV     A,@ADDR1          ;Move data to array location
         MOV     #1,A              ;EECTL value = 1 (program 0s)
         CALL    EEPROG            ;do the write of only the
                                   ;needed 0s
         EINT                      ;interrupt can happen now
ONES     MOV     @ADDR1,A          ;get the current data
         XOR     TEMP2,A           ;bits that change = 1
         AND     TEMP2,A           ;bits that change from
                                   ;0 to 1 = 1
         JZ      LASTCHK           ;are there any 1s to program?
WRITE1   DINT                      ;No interruptions
         MOV     A,@ADDR1          ;Move data to array location
         MOV     #3,A              ;EECTL value=3 (program 1s)
         CALL    EEPROG            ;do the write of only the
                                   ;needed 1s
         EINT                      ;interrupt can happen now
LASTCHK   MOV    @ADDR1,A          ;Check new memory against
                                   ;wanted memory
         CMP     TEMP2,A           ;if equal then exit
         JEQ     EXITW
                                   ;ERROR handling routine here
EXITW    RTS                       ;back to calling program
```

The following example is the same as the PROGRAM routine above but with actual values given for each step. The values shown are the LSB nibble of a byte expressed in binary and choosen because they give all possible bit combinations. In this example the memory address already has x1100 and we want to program x1010 to that address. Before calling the EEPROG routine, the program writes new data to the EEPROM address located in register ADDRESS and then passes data in register A which specifies either a write ones or a write zeros operation.

```
                                  A     TEMP2   @ADDRESS
                          ;       x1010   x       x1100
PROGRAM  CLR     ETYPE    ;       x1010   x       x1100   Data EEPROM type
         CMP     #01Fh,ADDR1-1;   x1010   x       x1100   Data EERPOM at 1Fxxh
         JEQ     ISSAME   ;       x1010   x       x1100   Keep as data EERPOM
         MOV     #2,ETYPE ;       x1010   x       x1100   Set Program EEPROM
                          ;                               type
```

```
ISSAME  MOV     A,TEMP2     ;   x1010 x1010   x1100   save data
        MOV     @ADDR1,A    ;   x1100 x1010   x1100   read current data
        XOR     TEMP2,A     ;   x0110 x1010           different bits = 1
        JZ      EXITW       ;                         if A =0 data the same
        INV     A           ;   x1001                 different bits = 0
        OR      TEMP2,A     ;   x1011 x1010           0 is bit to change to
                            ;                         0
        BTJZ    #FF,A,WRITE0 ;x1011                   program 0's if any
                            ;                         0's
        JMP     ONES        ;                         check write1's
WRITE0  DINT                ;   x1011 x1010   x1100   disable interupt
        MOV     A,@ADDR1    ;   x1011 x1010   x1011   move data to address
        MOV     #1,A        ;   x0001                 program to write0's
        CALL    @EEPROG     ;                         do write0's
        EINT                ;   x0000 x1010   x1000   enable interupts
ONES    MOV     @ADDR1,A    ;   x1000 x1010   x1000   read new current data
        XOR     TEMP2,A     ;   x0010 x1010           bits that change = 1
        AND     TEMP2,A     ;   x0010 x1010           1 is bit to change to 1
        JZ      LASTCHK     ;
WRITE1  DINT                ;   x0010 x1010   x1000   disable interupts
        MOV     A,@ADDR1    ;                 x0010   move data to address
        MOV     #3,A        ;   x0011                 program to write1's
        CALL    @EEPROG     ;                         do write1's
        EINT                ;   x0000 x1010   x1010   enable interupts
LASTCHK  MOV    @ADDR1,A    ;   x1010 x1010   x1010   read new current data
        CMP     TEMP2,A     ;                         compare to data wanted
        JEQ     EXITW       ;                         the same then return
        CALL    @ERROR      ;If the program gets here there has been an
                            ;error
EXITW   RTS                 ;
```

13.3 Serial Communications

All devices in the TMS370 family provide serial communication capability with peripheral devices. The TMS370Cx10 series provides one serial (SPI) port providing communication capability with peripheral devices. The TMS370Cx50 series provides two serial (SPI and SCI) ports for enhanced communications capability with peripheral devices.

13.3.1 SPI Port Interfacing

The SPI port provides synchronous communication with peripherals such as shift registers, display drivers, A/D converters, and another CPU. Synchronous transmission is supported by programmable parameters such as character length (one to eight bits) and bit transfer rate (eight options). In the example below, the SPI port is configured as a Master/Slave dual CPU interface. This full-duplex setup has the master CPU initiating data transfer by sending the SPICLK signal to the slave. Data is then transmitted between the CPUs simultaneously until the clock signal stops. Either or both of the data lines may send valid or dummy data, depending on the software.

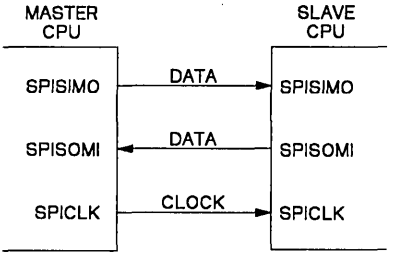


Figure 13-8. Master/Slave CPU Interface Example

13.3.2 SCI Port Interfacing

The SCI port (TMS370Cx50 only) provides communication with peripheral devices in either an Asynchronous or Isosynchronous format. This makes it especially suited for communicating with a variety of devices. The format parameters of the SCI are software programmable and are as follows:

Parameter	Options
Mode	Asynchronous, Isosynchronous
Baud rate	64K possibilities
Character length	1 to 8 bits
Parity	Even, Odd, Off
No. of stop bits	1 or 2
Interrupt priorities	Receiver/transmitter

In the figure below, the SCI port is configured for a RS-232-C type interface. 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 format, the clock signal does not need to be transmitted, but is generated locally at both ends.

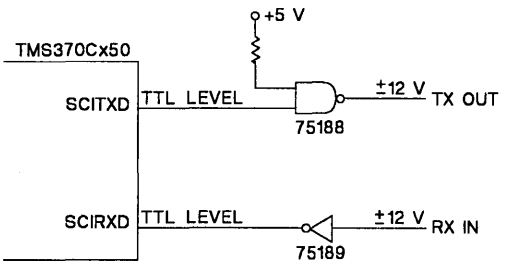


Figure 13-9. SCI/RS-232 Interface Example

The following routine automatically calculates the baud rate for the SCI port by timing the length of the start bit. Many times this eliminates the need for external select switches which can cause confusion.

This routine converts the Receiver pin to a general purpose input pin and then samples this pin until it finds the start bit. Sampling is controlled by the baud rate 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, it's low state is sampled until the high state of the first data bit (of an odd ASCII value) is detected. The number of times the start bit is sampled is used by the baud rate registers to figure the baud rate.

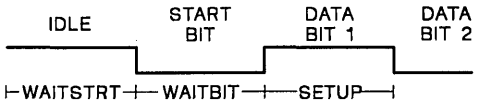


Figure 13-10. Auto Baudrate Waveform

```
0050            SCICCR   .EQU P050 ;SCI communication control register
0051            SCICTL   .EQU P051 ;SCI control register
0052            BAUDMSB  .EQU P052 ;baud rate counter MSB
0053            BAUDLSB  .EQU P053 ;baud rate counter LSB
0054            TXCTL    .EQU P054 ;Transmitter control
0055            RXCTL    .EQU P055 ;Receiver control
0057            RXBUF    .EQU P057 ;Receiver buffer
0057            TXBUF    .EQU P059 ;Transmitter buffer
005D            SCIPC1   .EQU P05D ;Port control 1 (SCLK)
005E            SCIPC2   .EQU P05E ;Port control 2 (TXD,RXD)
005E            SCIPR1   .EQU P05E ;priority register
005E            COUNT    .EQU R04  ;temporary counting register
0000
0000            .TEXT    07000h    ;INITIALIZE SCI PORT WITH
0000                               ;A CR (RETURN)
7000            AUTOBAUD           ;Baud automaticaly
7000                               ;set on Odd ASCII
7000                               ;character
7000 D504       CLR  COUNT         ;clear count register
7002 D503       CLR  COUNT 1       ;COUNT-1
7004 F7005E     MOV  #0,SCIPC2     ;set RxD to general
7007                               ;purpose input pin
7007 A6085EFC   WAITSTRT BTJO #8,SCIOC2,WAITSTRT ;wait for
700B                               ;a start bit to go low
700B
700B B3         WAITBIT  INC    A ;dummy, gives 32
700C                               ;clock states
700C                               ;(1 min baud)
700C 700104 INCW #1, COUNT         ;increment counter
700F A7085EF8  BTJZ #8,SCIPC2,WAITBIT ;wait until start bit
7013                               ;ends (ASCII char=odd)
7013 70FF04 SETUP INCW #-1,COUNT   ;one less than count
7016                               ;into baud reg
7016 715304 MOV COUNT,BAUDLSB      ;since the SCI starts
7019                               ;from count 0.
7019 715203 MOV COUNT-1,BAUDMSB    ;initialize baud rate
701C                               ;registers.
701C F7225E MOV #22h,SCIPC2        ;Enable Rx and Tx pins
701F F7025D MOV #2,SCIPC1          ;enable SCLK pin (if needed)
7022 F77750 MOV #01110111b,SCICCR;8-bits length, even parity,
7025                               ;1 stop ;bit
7025                               ;only even, odd, or none parity
7025                               ;determined
7027                               ;by SCICCR value
```

```
7025 F73351 MOV #00110011b,SCICTL;enable Tx, Rx, SCLK = internal
702C                           ;program after input character
7028                           ;finishes)
7028 F70154 MOV #1,TXCTL       ;enable TX interrupts
702B F70155 MOV #1,RXCTL       ;enable RX interrupts
702E 8057   MOV RXBUF,A        ;clear out garbage from SCI (Place in
7032                           ;program after input character finishes)
7032 F00C   EINT
```

This routine can be improved to give greater flexibility and accuracy using some of the following suggestions:

1. Time more than one bit and then divide by the number of bits to give a greater accuracy. This means that a more carefully choosen character must be used to start the autobaud routine. The current routine can use 50 per cent of the ASCII values (all odd ASCII values).

2. Add 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.

3. Add routines to compare the count of another bit in the character to the start bit count as an accuracy check. This gives the same problems as before.

13.4 Analog/Digital Converter

The A/D converter provides the TMS370Cx50 with built-in data acquisition capability with 8-bit resolution. Any or all of the A/D pins may be used as a single-ended input with reference to analog ground ($V_{SS}3$). Pins not used for A/D conversion may be software configured as a standard digital input pin. The high reference voltage (V_{REF}) may be either $V_{CC}3$ or supplied by one of the inputs. If the sampled input is higher than V_{REF}, the conversion value placed in the A/D data register is FFh, indicating full scale. If the sampled input is lower than $V_{SS}3$, then the value 00h is placed in the A/D data register.

A variety of functions may be performed by the CPU using the A/D converter. Industrial applications may include temperature sensing, fluid level monitoring, and recharging circuit status as indicated in the figure below. If the sending units are designed for greater than V_{ref}, then a resistance network may be needed to keep the A/D input voltage within the meaningful range of V_{ref} to ground. This is especially true in the case of a fluid level sensor, where the full linear range may be required.

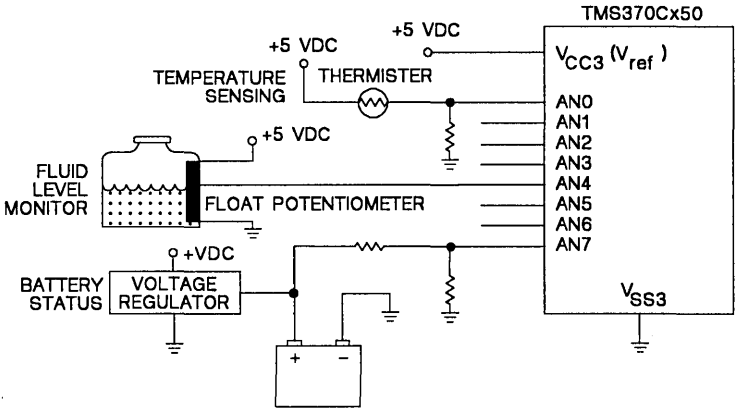


Figure 13-11. A/D Converter Sample Applications

13.5 Sample Routines

The following section contains sample routines that show the various ways the TMS370 handles common software tasks.

13.5.1 T1PWM Pin Setup

The following examples start and stop 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 in the examples 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
```

The examples below show the two instructions needed when changing 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 examples keep the current value on the pin when starting or stopping the PWM functin. 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.
```

13.5.2 Clear RAM

This routine clears all the internal RAM registers. It can be used at the beginning of a program to initialize the RAM to a known value.

REGISTER	FUNCTION
A	Holds the initialization value
B	Index into the RAM

```
0000            0001      .TEXT 7000h    ;absolute start address
7000 52FE       0002 CLEAR MOV #254,B    ;number of register to clear
                0003                     ;-2
7002 B5         0004       CLR   A       ;load the initialization
                0005                     ;value of zero
7003 AB0001     0006 LOOP MOV    A,1(B)   ;clear the location indexed
                0007                     ;by B+1
7006 CAFB       0008       DJNZ  B,LOOP  ;loop until all RAM is
                0009                     ;cleared
7008            0010                     ;A and B end up as zeros.
```

13.5.3 RAM Self Test

This routine performs a simple alternating 0/1 test on the RAM. The RAM is tested by writing a AA,55 pattern to the entire RAM and then checking the RAM for this pattern. The inverted pattern is then written to RAM and rechecked. Finally, the entire RAM is cleared. If an error is found, a bit is set in the flag register. The error flag bit should be cleared before the routine is started.

REGISTER	BEFORE	AFTER NO ERROR	AFTER ERROR
A	XX	0	?
B	XX	0	?
Rn	XX	0	?
FLAG	XX	0	Bit 0=1

Passing data	None
Registers affected	All
Ending data	All registers = 0 Bit 0 in FLAG = 1 if error was found

```
0000            0001        .TEXT 7000H    ;absolute start address
000A            0002 FLAG   .EQU  R10      ;error register
7000 2255       0003        MOV   #55h,A   ;Start RAM fill with 55h
7002 52FD       0004 FILLR  MOV   #0FDh,B  ;Set RAM start address - 2
7004            0005                       ;(don't change register A or B)
7004 AB0002     0006 FILL1  MOV   A,2(B)    ;fill RAM with aa to 55 pattern
7007 BC         0007        RR    A        ;change to beginning number
7008 CAFA       0008        DJNZ  B,FILL1  ;fill entire ram with pattern
700A BC         0009        RR    A        ;change to beginning number
700B 52FD       0010        MOV   #0FDh,B  ;refresh index
700D AD0002     0011 COMPAR CMP   2(B),A    ;check for errors
7010 06**       0012        JNE   ERROR    ;exit if values don't match
7012 BC         0013        RR    A        ;change from 55 to AA to 55
7013 CAF8       0014        DJNZ  B,COMPAR ;check the entire RAM
7015 B0         0015        CLRC           ;is reg A now 55, AA or 00?
7016 01EA       0016        JN    FILLR    ;=AA, change to opposite pattern
7018 02**       0017        JZ    EXIT     ;=00,
701A B5         0018 FILL0  CLR    A       ;=55,clear the ram now
```

```
701B 00E5      0019        JMP    FILLR     ;repeat the fill and check routine
701D 74010A    0020 ERROR  OR     #1,FLAG   ;set bit zero in the flag
7020           0021                         ;register
7020           0022 EXIT   .EQU   $         ;continue program here
```

13.5.4 ROM Checksum

This routine checks the integrity of the ROM by performing a checksum on the entire ROM. All ROM bytes from 7002h to 7FDFh are added together in a 16-bit word. The sum is checked against the value at the beginning of the ROM (7000h, 7001h). If the values don't match, then an error has occured and a bit is set in a register. The error flag bit should be cleared before the start of the routine.

> **Note:** Addresses 7FE0h through 7FEBh are reserved for TI use only and should not be used in a checksum calculation.

REGISTER	BEFORE	NO ERROR	ERROR
A	XX	0	?
B	XX	0	?
R2	XX	CHKSUM MSB	CHKSUM MSB
R3	XX	CHKSUM LSB	CHKSUM LSB
R4	XX	70h	70h
R5	XX	01h	01h
R6	XX	FFh	FFh
R7	XX	FFh	FFh
FLAG	XX	Bit 1=0	Bit 1=1

```
0000           0001          .TEXT 7000h   ;absolute start address
000F           0002 FLAG     .EQU  R15     ;error status
3039           0003 CHECKSUM .EQU  12345   ;value to be checked
7000 3039      0004          .WORD CHECKSUM;put correct checksum into
7002           0005                        ;ROM
7002           0006                        ;other initialization
7002           0007                        ;program here
7002 887FDF05  0008 ROMCHK   MOVW #7FDFh,R5 ;starting address (end of
7006           0009                        ;memory)
7006 880FDD07  0010       MOVW  #0FDDh,R7  ;number of bytes to add + 1
700A 88000003  0011       MOVW  #0,R3      ;reset summing register
700E           0012                        ;
700E 9A05      0013ADDLOP MOV  @R5,A       ;get ROM byte
7010 480003    0014       ADD   A,R3       ;add to 16-bit sum
7013 790002    0015       ADC   #0,R2      ;add any carry
7016 70FF05    0016       INCW  #-1,R5     ;decrement address
7019 70FF07    0017       INCW  #-1,R7     ;decrement byte counter
701C 03F0      0018       JC    ADDLOP     ;continue until byte count
701E           0019                        ;goes past 0
701E           0021                        ;
701E 8A7000    0022       MOV   7000h,A    ;compare MSB stored to MSB
7021           0023                        ;sum
7021 4D0002    0024       CMP   A,R2       ;
7024 06**      0025       JNE   ERROR      ;set error bit if different
7026 8A7001    0026       MOV   7001h,A    ;compare LSB stored to
7029           0027                        ;LSB 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
```

13.5.5 Binary-to-BCD Conversion

This program converts a 16-bit binary word to a packed 6 nibble value.

REGISTER	BEFORE	AFTER
A	XX	BCD MSB
B	XX	BCD
R2	XX	BCD LSB
R3	BINARY MSB	ZERO
R4	BINARY LSB	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
```

13.5.6 BCD-To-Binary Conversion

This routine converts a four digit number to binary. The maximum BCD number is 9999 decimal. Operands originate and are stored in general purpose RAM. The BCD number is composed of the four digits D3, D2, D1, and D0 contained in the bytes DH and DL. The binary number is calculated by dividing the number into powers of ten (Binary = D3*1000 + D2*100 + D1*10 + D0*1). Multiplying by 10 is easier if the number is further broken up in other numbers so that D2*10 = D2*(8+2) = D2*8+D2*2. Likewise, multiplying by 1000 can be calculated by D3*(1000) = D3*(1024-24) = D3* (1024-(8+16)) = D3*1024- (D3*8 + D3*16). This may seem complex but it works quickly and uses few bytes.

```
0000                0010        .TEXT 7000h
0002                0011 BH     .EQU  R2          ;Binary number MSB
0003                0012 BL     .EQU  R3          ;Binary number LSB
0004                0013 DH     .EQU  R4          ;Decimal number MSB
0005                0014 DL     .EQU  R5          ;Decimal number LSB
7000                0017                          ;D0=ones, D1=tens,
700C                0018                          ;D2=hundreds, D3=thousands
700C D502           0023 TOP    CLR   BH          ;clear out binary MSB
700E 420503         0024        MOV   DL,BL       ;D0 to B0
7011 730F03         0025        AND   #0Fh, BL    ;convert D0
7014                0026                          ;
7014 1205           0027        MOV   DL,A        ;D1*10=D1*8+D1*2
7016 23F0           0028        AND   #0F0h,A     ;isolate D1
7018 C0             0029        MOV   A,B         ;B=D1*16
7019 D701           0030        SWAP  R1          ;B=D1
701B BC             0031        RR    A           ;A=D1*16/2=D1*8
701C CE             0032        RL    B           ;B=D1*2
701D 68             0033        ADD   B,A         ;A=D1*10   (D1*8+D1*2)
701E 480003         0034        ADD   R0,BL       ;D1:D0 converted
7021                0035                          ;
7021 3204           0036        MOV   DH,B        ;get upper two digits
7023 530F           0037        AND   #0Fh,B      ;isolate D2
7025 5C64           0038        MPY   #100,B      ;R0:R1=D2*100
```

```
7027 480103     0039        ADD   R1,BL      ;add to current total
702A 490002     0040        ADC   R0,BH      ;D2:D1:D0 converted
702D            0041                         ;
702D 1204       0042        MOV   DH,A       ;isolate D3
702F 23F0       0043        AND   #0F0h,A    ;A=D3 * 16
7031 C0         0044        MOV   A,B        ;B=D3 * 16
7032 CD         0045        RRC   B          ;B=D3 * 8
7033 68         0046        ADD   B,A        ;A=D3 * 24
7034 4A0003     0047        SUB   R0,BL      ;BH:BL=BH:BL-24*D3
7037 7B0002     0048        SBB   #0,BH      ;
703A B0         0049        CLRC             ;setup for rotate
703B CD         0050        RRC   B          ;B=D3*4
703C 480102     0051        ADD   R1,BH      ;BH:BL=BH:BL+D3*4*256
703F            0052                         ;    .
```

13.5.7 BCD String Addition

The following subroutine uses the addition instruction to add two multi-digit numbers together. Each number is a packed BCD string, 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 favors storing all numbers and addresses with the most significat byte in the lower numbered location.

REGISTER	BEFORE	AFTER	FUNCTION
A	XX	??	Accumulator
B	XX	0	Length of string
R2	XX	??	Temporary save register
STR1	BINARY MSB	no change	BCD string
STR2	BINARY LSB	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. Also the need 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.

13.5.8 Fast Parity

This routine presents a quick way to determine the parity of a byte. by exclusive ORing all the bits of the byte together, a single bit will be derived which is the even parity of the word. When exclusive ORing an even number of 1s will combine to form a 0, leaving either an odd 1 or 0 bit. This routine keeps splitting the byte in half and exclusive ORing the two halves.

REGISTER	BEFORE	AFTER	FUNCTION
A	TARGET	??	Passing byte from program
B	XX	??	Length of string
CARRY	XX	Parity	Status bit,result to calling routine

```
*********************************************************************
*       STEP 1                                          SUBROUTINE
*       Byte bits    7654 3210                          TO FIND
*                XOR      7654 [MSN above]             EVEN PARITY
*                  ===========
*                    xxxx ABCD
*       STEP 2                 ----->   AB CD
*                                XOR     AB [MS bits above]
*                                  =======
*                                    xx ab
*       STEP 3                              --->   a b
*                                              XOR   a [MS bit]
*                                                =====
*                                                  x P   {answer}
*
*********************************************************************
0000            0001       .TEXT 7000h;absolute start address
7000 C0         0002 PARITY MOV  A,B ;duplicate the target byte
7001 B7         0003        SWAP A   ;line up the ms nibble with the ls
7002            0004                 ;nibble
7002 65         0005        XOR  B,A ;exclusive OR the nibbles to get   a
7003            0006                 ;nibble answer
7003 C0         0007        MOV  A,B ;duplicate the nibble answer
7004 BC         0008        RR   A   ;line up bits 0,1 of the answers to
7005            0009                 ;bits
7005 BC         0010        RR   A   ;2, 3 of the answer
7006 65         0011        XOR  B,A ;XOR to get a new 2-bit answer
7007 C0         0012        MOV  A,B ;duplicate this 2 bit answer
7008 BC         0013        RR   A   ;line up bit 0 with bit 1
7009 65         0014        XOR  B,A ;XOR to get final even parity answer
700A BC         0015        RR   A   ;rotate answer into the carry bit
700B            0016                 ;and bit 7
700B F9         0017        RTS      ;carry = 0 = even # of 1's
700C            0018                 ;carry = 1 = odd # of 1's
700C            0019                 ;use JC, JN or JNC  in next executed
700C            0020                 ;instruction
```

13.5.9 Bubble Sort

This routine will sort up to 256 bytes using the bubble sort method. Longer tables could be sorted using the Indirect Addressing mode.

REGISTER	FUNCTION
A	Temporary Storage Register
B	Index into the Table
R2	Holds flag to indicate a byte swap has been made

```
0000            0001       .TEXT 7000h        ;absolute start address
2000            0002 TABLE .EQU  2000h        ;start of data table in RAM
0002            0003 FLAG  .EQU  R2           ;'swap has been made' flag
7000 D502       0004 SORT  CLR   FLAG         ;reset swap flag
7002 52FF       0005       MOV   #0FFh,B      ;load table offset value
7004 AA2000     0006 LOOP1 MOV   TABLE(B),A   ;look at entry in table
```

```
7007 AD1FFF     0007        CMP   TABLE-1(B),A ;look at next lower byte
700A 0B**       0008        JHS   LOOP2        ;if higher or equal, skip to
700C            0009                           ;next value
700C D302       0010        INC   FLAG         ;entry is not lower, set swap
700E            0011                           ;flag
700E B8         0012        PUSH  A            ;store upper byte
700F AA1FFF     0013        MOV   TABLE-1(B),A ;take lower byte
7012 AB2000     0014        MOV   A,TABLE(B)   ;put where upper was
7015 B9         0015        POP   A            ;get the old upper byte
7016 AB1FFF     0016        MOV   A,TABLE-1(B) ;put where the lower byte was
7019 CAE9       0017        LOOP2 DJNZ  B,LOOP1;loop until all the table is
701B            0018                           ;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
```

13.5.10 Table Search

Table searches are efficiently performed by using the CMPA (Compare Register A Extended) instruction. In the following example, a 150 byte table is searched for a match with a 6-byte string.

REGISTER	BEFORE	AFTER	FUNCTION
A	XX	??	
B	XX	??	
R2	XX	??	Table Length
TABLE	XX	no change	Long string in table
STRING	XX	no change	Target string, 6 bytes max

```
0000            0001          .TEXT 7000h    ;absolute start address
2000            0002 TABLE    .EQU  2000     ;start of data table in RAM
000A            0003 STRING   .EQU  R10      ;start of target string,
7000            0004                         ;6 bytes max
7000 729602     0005 SEARCH   MOV   #150,R2 ;table length = 150 bytes
7003 5206       0006 LOOP1    MOV   #6,B     ;string length = 6 bytes
4005 D602       0007 LOOP2    XCHB  R2       ;swap pointers, long string in B
7007 C2         0008          DEC   B        ;reduce index into table
7008 07**       0009          JNC  NOFIND    ;table end? if so, no match found
700A AA2000     0010          MOV TABLE(B),A;load test character
700D D602       0011          XCHB  R2       ;swap pointers, string pointer in
700F AD0009     0012          CMP STRING-1(B),A ;match?
7012 06EF       0013          JNE   LOOP1    ;if not, reset string pointer
7014                                         ;else test
7014 CAEF       0014          DJNZ  B,LOOP2  ;next character
7016            0015 MATCH    .EQU  $        ;match found
7016            0016 NOFIND   .EQU  $        ;no match found
7016            0017
```

The Indexed Addressing mode is used in this example and has the capability to search a 256-byte string, if needed. Register B alternates between a pointer into the 6 byte test string and a pointer into the longer table string.

13.5.11 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, and Register A and B are altered.

```
*******************************************************
*   16-BIT MPY:                  XH    XL       X VALUE
*                           X    YH    YL       Y VALUE
*                                --------------
*                                XLYLm XLYL1     1 = LSB
*                           XHYLm XHYL1          m = MSB
*                           XLYHm XHYH1
*          +  XHYHm  XHYH1
*          ------------------------------
*             RSLT3  RSLT2  RSLT1  RSLT0
*******************************************************
XH        .EQU      R2          ;higher operand of X
XL        .EQU      R3          ;lower operand of X
YH        .EQU      R4          ;higher operand of Y
YL        .EQU      R5          ;lower operand of Y
RSLT3     .EQU      R6          ;MSB of the final result
RSLT2     .EQU      R7
RSLT1     .EQU      R8
RSLT0     .EQU      R9          ;LSB of the final result

MPY32     CLR       RSLT2       ;clear the present value
          CLR       RSLT3
          MPY       XL,YL       ;multiply LSB's
          MOV       B,RSLT0     ;store LSB in result register 0
          MOV       A,RSLT1     ;store MSB in result register 1
          MPY       XH,YL       ;get XHYL
          ADD       R1,RSLT1    ;add to existing result XLYL
          ADC       R0,RSLT2    ;add carry if 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 MSB's
          ADD       R1,RSLT2    ;add once again to the result register
          ADC       R0,RSLT3    ;do the final add to the result reg
          RTS                   ;return to call subroutine
```

13.5.12 Keyboard Scan

This routine reads a 16 key keyboard, returns the hex digit of the key and debounces the key to avoid noise. A 'valid key' flag is set when a new key is found.

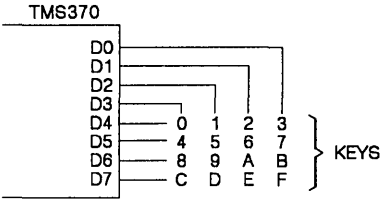


Figure 13-12. Keyboard Scan Values

REGISTER	BEFORE	AFTER NO KEY	AFTER NEW KEY	FUNCTION
A	XX	0	COLUMN	Temporary
B	XX	0	ROW	Temporary
R2	XX	16	KEY #	Temp store for Key value
R3	OLD KEY	0FFh	KEY #	Holds key pressed now
R4	DEBOUNCED	0	0	Debounce counter, old key or new
R5	GENERAL BITS	?xxxxxxx0	?xxxxxxx1	One bit of register is 1 if new key

```
0000           0001        .TEXT 07000h     ;
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
```

13.5.13 Divide 1

The routine divides a 16-bit number by an 8-bit number to give a 16-bit quotient and an 8-bit remainder. The DIV instruction is used to accomplish this task.

```
0000            0021          .TEXT 7000h ;
700B            0022 OVERFLOW .EQU R7     ;
700B            0023                      ;divisor     -R3, quotient LSB-R5
700B            0024                      ;dividend MSB-R1, quotient MSB-R4
700B            0025                      ;dividend LSB-R2, remainder -B
700B            0026                      ;uses R0
700B            0027                      ;
700B B5         0028 Divide8 CLR A        ;clear MSB of first dividend
700C F4F803     0029         DIV R3,A     ;divide MSB of dividend to get MSB
700F 08**       0030         JV OVERF     ;exit if overflow
7011 D004       0031         MOV A,R4     ;quotient. Move MSB of quotient
7013            0032                      ;to storage.
7013 62         0033         MOV B,A      ;move remainder to MSB of dividend
7014 3202       0034         MOV R2,B     ;move dividend LSB to LSB position
7016 F4F803     0035         DIV R3,A     ;divide to get quotient LSB and
7019            0036                      ;remainder
7019 08**       0037         JV OVERF     ;exit if overflow
701B D005       0038         MOV A,R5     ;store the quotient LSB next to MSB
701D F9         0039         RTS          ;remainder in B
701E            0040                      ;
701E D311       0041 OVERF   INC OVERFLOW ;set bit 0 if overflow
7020 F9         0042         RTS          ;
```

13.5.14 Divide Instruction 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 numbers. 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 MSB
700B            0029 ;    B =                 Remainder LSB
700B            0030 ;    R2= Dividend MSB    Quotient  MSB
700B            0031 ;    R3= Dividend LSB    Quotient  LSB
700B            0032 ;    R4= Divisor  MSB    Divisor   MSB
700B            0033 ;    R5= Divisor  LSB    Divisor   LSB
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 MSB,
701C F8         0048         SETC        ;Correct by subtracting divisor
701D            0049                     ;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 MSBs of dividend
7021            0053                     ;and divisor
7021 07**       0054         JNC DIVEND  ;Jump if divisor is bigger
7023 06**       0055         JNE MSBNE   ;If equal compare LSBs.
7025 3D05       0056         CMP R5,B    ;Compare LSBs.
```

```
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
```

14. Development Support

Texas Instruments provides extensive development support for the TMS370 family. The TMS370 series unified development support tools consists of the following components:

- Assembly Language Tools
- Extended Development Support (XDS[3]) System with associated software
- EEPROM Programmer
- TMS370C810 and TMS370C850 Devices (for prototyping)

These development tools are designed to work with an IBM, IBM compatible or TI PC. The TMS370 system designer can use a text editor to generate the assembler source code, then use the assembly language tools to assemble the source modules and link the assembled modules. The object file may then be tested with either the XDS System or a TMS370C8x0 device, both which provide full speed in-circuit emulation. The XDS and debugger software provides realtime breakpoint/trace/timing functions to facilitate hardware and software integration during system development.

The EEPROM Programmer provides the means of programming the device used for prototyping and emulation. The TMS370C8x0 devices can be used for prototyping and emulation of masked ROM parts, as well as a medium for submitting the program to TI for masked ROM production.

This section discusses key features of the TMS370 development tools. For a detailed desciption of system components, refer to the documents listed in Section 1.5 on page 1-8. The topics included in this section are as follows:

[3] XDS is a registered trademark of Texas Instruments Incorporated.

14.1 The Assembly Language Tools

The TMS370 assembly language tools (Figure 14-1) include an assembler, linker, archiver, and a code conversion utility. These tools are available from TI on a 5 1/4 inch floppy diskette for IBM, IBM compatible and TI PC's. The PC should be running PC-DOS or MS-DOS version 2.1 or later, and have at least 512K bytes of memory space available for the assembler and linker operation.

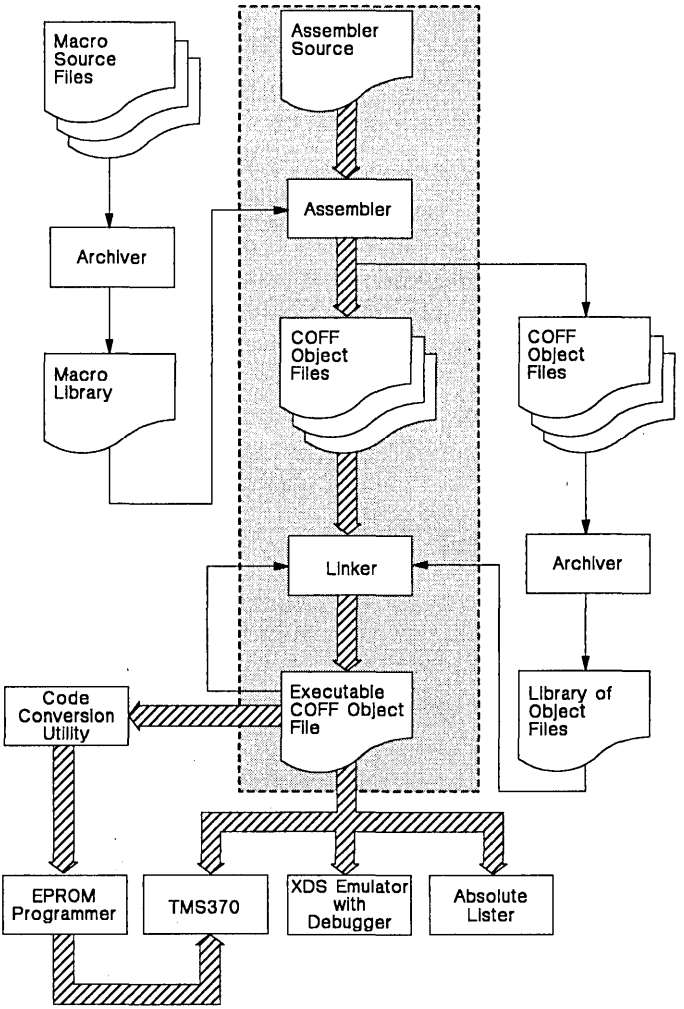


Figure 14-1. Software Development Flow

14.1.1 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 how the machine code is placed into the TMS370 memory space.

The assembler is a one-pass assembler. 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 XDS debugger to provide the user with **symbolic debugging**. The XDS 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 TI EEPROM programmer to form a PC memory image of the data loaded for programming.

14.1.2 The Linker

The TMS370 linker creates executable modules by combining COFF object files. The concept of user definable COFF *sections* is basic to the linker operation. The linker accepts several types of files as input :

- Relocatable COFF object files produced by the TMS370 assembler
- Command files
- Archive object libraries
- Output modules created by a previous linker run (these are referred to as partially linked files)

As the linker combines object files, it performs the following tasks:

- Allocates sections into the target system's configured memory
- Relocates symbols and sections to assign them to final addresses
- Resolves undefined external references between input files

The linker supports a C-like command language that controls memory configuration, section definition, and address binding. The language supports expression assignment and evaluation, and provides two powerful directives, MEMORY and SECTIONS, that allow you to:

- Define a memory model that conforms to target system memory

- Combine object file sections
- Allocate sections into specific areas of memory
- Define overlayed memory structures
- Define or redefine global symbols at link time

Figure 14-2 shows the operation of the linker on two source code files. Each file has been assembled and contains four default sections and one named section. The various sections are arranged in the order dictated either by the linker's default method or by a user supplied control file. The executable object module shows the combined sections, and the memory map indicates the location of the sections in memory.

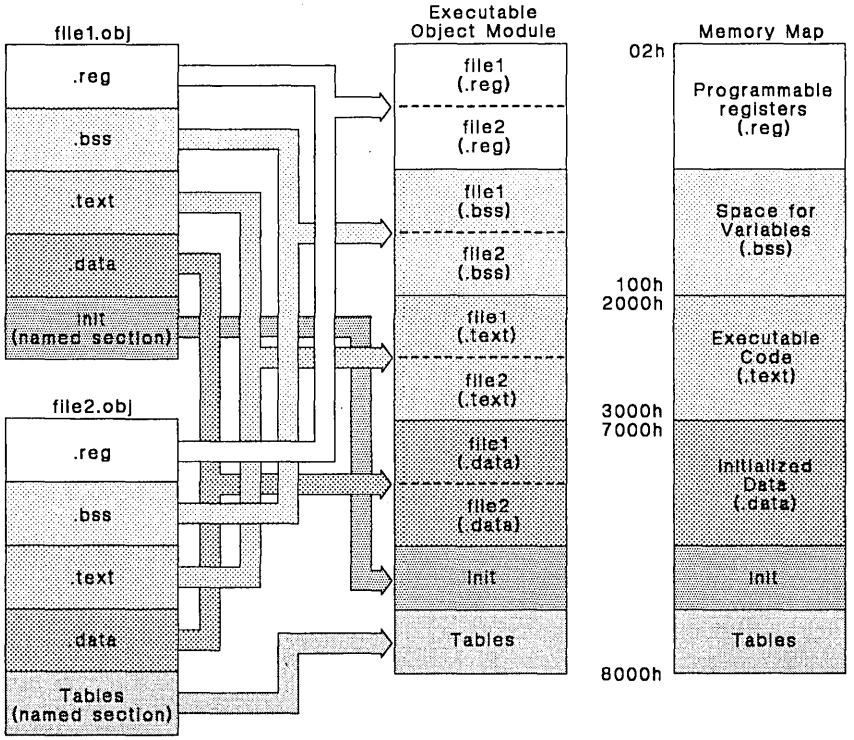


Figure 14-2. Linker Output Generation

14.1.3 The Archiver

The archiver provides file management by allowing a group of files to be collected into a single library. For example, macros can be collected by the archiver, then fetched by the assembler as directed by the source file. Object modules can also be collected into a library for convenient access by the linker. While not necessary for program development, the archiver can provide valuable organization in the building of the executable COFF object file.

14.1.4 Code Conversion Utility

The code conversion utility converts COFF object files to the Intel hex object format. Code conversion is necessary when not using the TI EEPROM Programmer, since most other (non-TI) EPROM programmers do not accept COFF object files as input. Code in the Intel hex object format can be downloaded to most EPROM programmers.

14.2 The XDS System

The XDS System is a self contained package that provides full-speed, in-circuit emulation and debugging functions required for program development of the TMS370 family devices. Key features of the XDS emulation function include:

- 20 MHz full-speed in-circuit emulation of all TMS370 family members
- Realtime hardware breakpoint/trace/time analysis capibilities
- Execution of programs from internal XDS memory (64K) or target memory
- Support of both microcomputer and microprocessor modes
- Large trace buffer, 2048 samples
- Full logic tracing with logic analyzer interface cable

The XDS System hardware includes a chassis, power supply, power and interfacing cables, and a three board set consisting of an emulator, communications board, and a breakpoint/trace/timing board. At the heart of the XDS system is a special system emulator chip containing all the peripheral modules and I/O line circuits that precisely duplicate the TMS370's logic and performance. The internal XDS memory can be used to emulate on-chip ROM and/or external memory.

The XDS debugger function is provided by software which runs on a PC. The software provides interactive control of the emulator with the following features:

- Window oriented user interface with menu-driven command structure
- Direct symbolic referencing from downloaded assembly symbol tables
- Full symbolic expression analysis that recognizes all assembly language operators
- Symbolic reverse assembler
- Ability to display and change registers and memory

14.2.1 XDS System Configuration Requirements

A functional XDS System configuration consists of the XDS System and the following user-supplied components:

- IBM, IBM compatible or TI PC with 512K bytes minimum and serial communication port
- MS/PC-DOS version 2.1 or later
- Monitor (preferably color, to better highlight field and value changes)

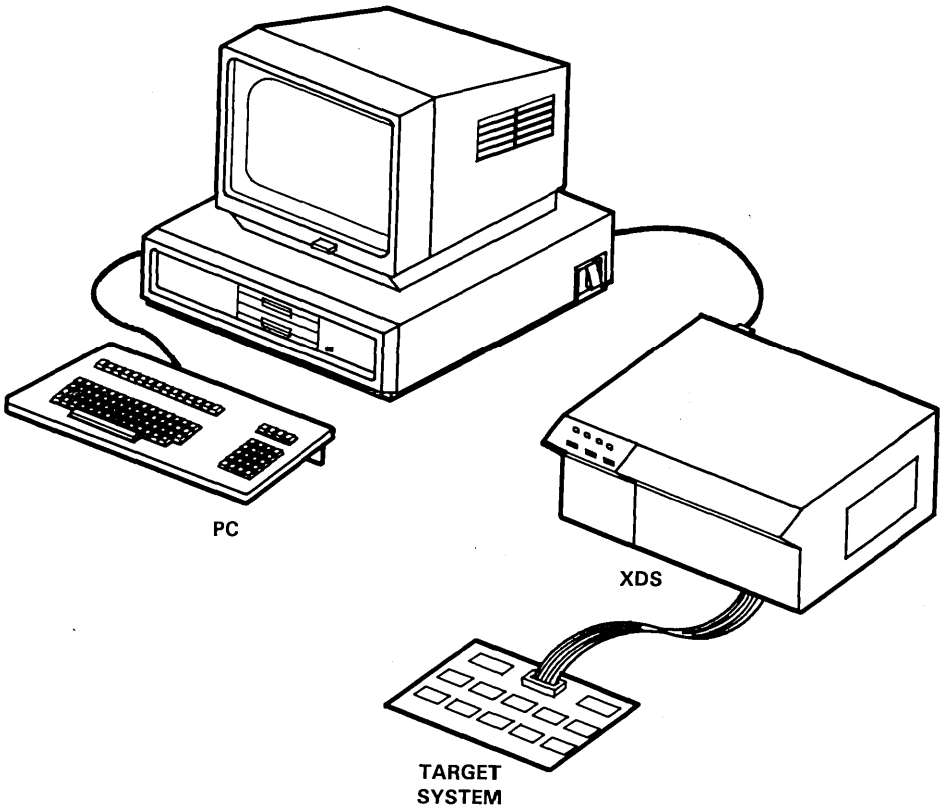


Figure 14-3. Typical XDS System Configuration

14.2.2 The Debugger Function

The XDS debugger function is provided by software that runs on a PC. This program provides window oriented, interactive programming that facilitates the development of applications for TMS370 family devices. The user develops an executable COFF object file using a text editor and the TMS370 assembly language tools. The debugger function allows the object file to be downloaded into the target device or the emulator memory of the XDS System. The debugger and emulator functions then provide evaulation of microprocessor/program operation.

A user debugging a system needs to focus on a number of different areas such as the code being executed, the registers of the target machine, and the variables in the program. The debugger aids this by using a menu-oriented command language that allows control of the debugging process. The command language is designed to be both simple for the inexperienced user and efficient for the expert. This is accomplished by limiting command menus to just one or two levels, so that nearly everything in the debugger can be controlled by a simple two-letter command without wasting keystrokes. Commands requiring additional input or qualification provide a prompt for that information or a menu of subcommands.

The top level screen (Figure 14-4) of the debugger consists of the following elements:

- Available command menu
- Status line
- Information windows
- Function key reminder line

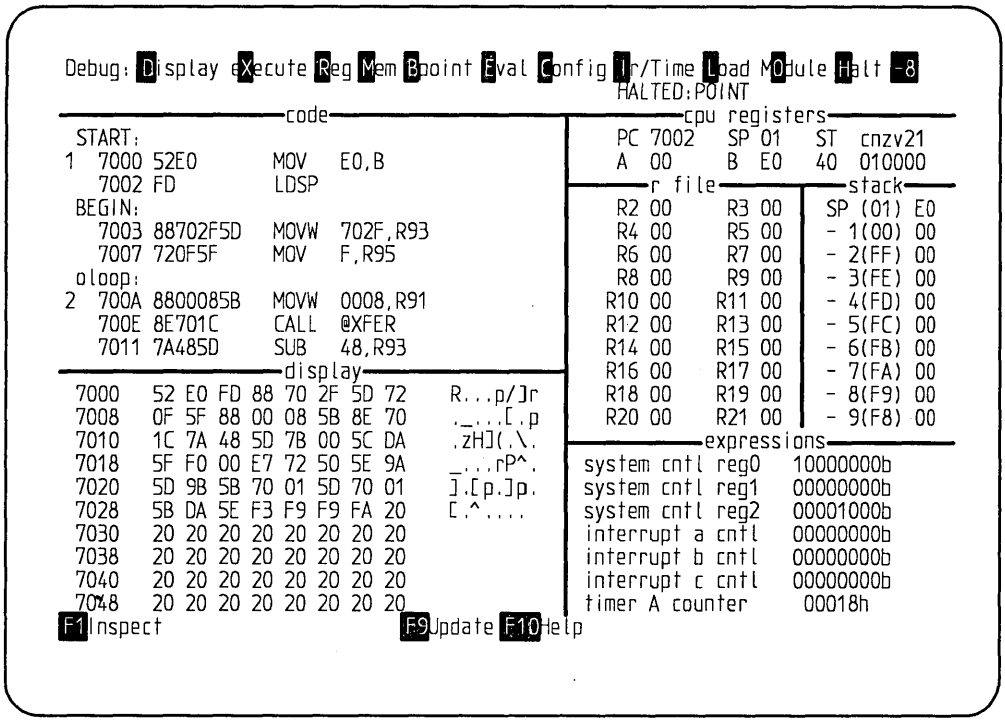


Figure 14-4. XDS Debugger Top Level Screen

The available command menu displays the single keystroke commands that can be used from that menu. Pressing the indicated key either performs the associated action or calls up a prompt or menu of subcommands. The status line indicates the current status of the system. The function key reminder line displays several functions provided by the function [F] keys.

The following paragraphs describe the information windows, which take up the area under the status line and above the function key reminder line. Some windows are based on virtual buffers in the XDS System, which means that the debugger keeps track of more information than can be displayed at one time. To view all of the information, the user simply scrolls through the window. The debugger automatically accesses the emulator whenever scrolling passes beyond the buffer's current contents. The user can easily move from window to window to make specific changes and view data. Information displayed in the windows is updated automatically whenever the microcontroller stops running, and manually with a function key. Updates or new values are highlighted for easy recognition.

14.2.2.1 Code Window

The code window (located in the upper left corner) displays disassembled object code being debugged. The current instruction at the PC is identified with a highlighted address. Also, instructions at which simple breakpoints have been set are marked with a numeric "breakpoint ID" to the left of the address. Immediate values in instructions are displayed in hexidecimal.

This window can be scrolled downward, and simple breakpoints can be added or removed. Upward scrolling is possible only up to the top of the virtual buffer, since backward disassembly is impossible.

14.2.2.2 Display Window

The display window is located in the lower left corner of the window area. This window displays miscellaneous debugger information such as:

- Memory, dumped in hexadecimal format
- Peripheral file register contents
- Symbols in the symbol table
- Object module names (with current module highlighted)
- PC text file with available control keys for "find" functions

This window can be scrolled up and down.

14.2.2.3 CPU Registers Window

The contents of five registers (A, B, PC, SP, and ST) are displayed in the CPU registers window in the upper right corner. The contents of these registers can be modified from this window. Scrolling is not needed since all available information is shown.

14.2.2.4 Register File Window

The register file window (located to the right of center) shows the contents of the register file, 20 registers at a time. The data can be scrolled up or down, and changed at will.

14.2.2.5 Stack Window

The stack window, located to the far right of center, dislays the contents of the current program stack within the register file. The stack window differs from the regsiter file window in that a) when updated, the window automatically changes the display to reflect the offset of each register from the current top of stack and b) the registers are displayed in reverse order, so that "higher" on the stack corresponds to "higher" in the window.

14.2.2.6 Expression Window

The expression window (located in the lower right corner) is used to display aribitrary expressions specified by the user. When prompted by the debugger for an address or a value, an arbitrary complex expression may be entered. The debugger evaluates expressions using the symbol table and the emulator. Expressions can consist of numeric constants, symbols, and register names, seperated by operators. All expressions are evaluated displayed as a 16-bit value in both hexadecimal and decimal. For example, if the expression "PC + 23h" is entered and the current value in the PC is 7000h, the debugger displays "PC+23h = 07023h = 28707". The debugger then prompts for a "save" upon which the expression is displayed in the window.

14.2.3 Breakpoint/Trace/Timing Functions

The Breakpoint, Trace and Timing (BTT) board of the XDS System monitors various microcontroller activities at the hardware level. The board can be programmed to take certain actions triggered by the occurance of specified qualifiers, depending on what *state* the board is in. The BTT is always in one of four states (Figure 14-5). Up to four actions can be qualified per state, with certain restrictions. A qualified event in one state can cause a transition to another state with a new set of parameters and actions becoming affective. This allows multilevel or sequenced breakpoints to be used for complex debugging problems.

The occurance of specified qualifiers results in an action taken by the BTT board. These qualifiers consist of the following:

ADDRESS The BTT monitors the memory bus during all memory cycles. Two address qualifiers can be used to trigger an action on a particular address or range of addresses. These can be used to define two distinct single point addresses, an inclusive range (any address within the range qualifiers), or an exclusive range (any address outside the range qualifiers). A mask can be specified to selectively ignore some or all of the external qualifier bits.

DATA The BTT also monitors the value on the data bus during each memory cycle. Two data qualifiers can be used with the data bus in exactly the same way as the two address qualifiers are used with the address bus. A mask can be specified to selectively ignore some or all of the external qualifier bits.

CYCLE Memory cycle types can be specified to qualify to trigger an action. Memory cycle types are read, write, and instruction fetch. Any one or any combination of these cycles can be qualified.

EXTERNAL The BTT can monitor the logic level of the eight external probe lines. A qualifier can be used to trigger an action on a particular value from these inputs. A mask can be specified to selectively ignore some or all of the external qualifier bits.

Actions that can be taken by the BTT board on the basis of the above qualifiers consist of the following:

BP/EVENT Triggering a breakpoint/event may cause either a hardware breakpoint, decrement the state counter or transfer control to the next (or beginning) state.

TRACE A cycle which satisfies the TRACE qualifiers will be stored in the TRACE buffer. This provides a history of the program execution for later inspection.

JUMP The BTT has four separate states in which different sets of actions can be specified. The JUMP action forces a transition into a different state when triggered.

POINT TIMER The BTT has two timers that can be started or stopped by qualified actions. The POINT TIMER action uses the two address qualifiers to control one timer. The timer is started when the first address is qualified and stopped when the other address is qualified.

RANGE TIMER The RANGE TIMER actions also control the BTT timers but differ from the point timer action in that one action starts a timer and a separate action stops it. Thus, there are actually two actions, "range timer start" and "range timer stop."

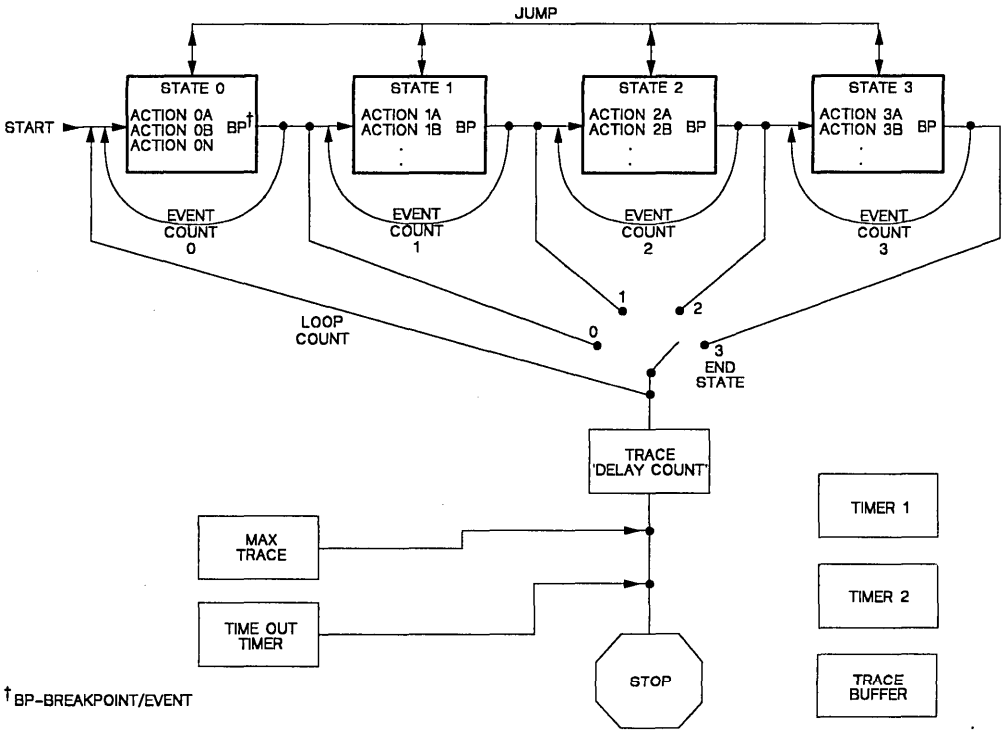


Figure 14-5. BTT Operation

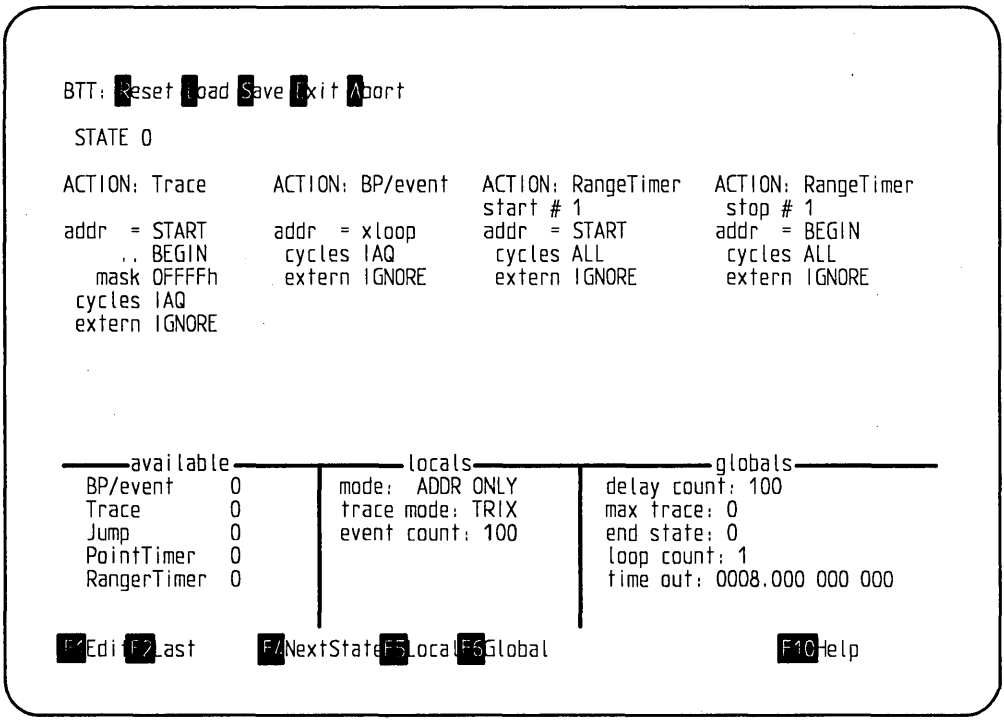


Figure 14-6. BTT Screen

The trace sample function of the BTT board provides "snapshot" storage of bus cycle activity. Up to 2047 samples, each 104 bits wide, can be stored by the circular trace buffer. As more samples are stored, the buffer wraps around, replacing the oldest samples with the newest ones. Each sample contains the following information:

- Address and data bus values
- Bus-cycle access type (read, write, instruction fetch)
- External logic-probe values
- BTT state and breakpoint/event indicators
- Time stamp from free-running timer (hours though nanoseconds)

The trace buffer screen provides a chronological display of the trace samples. Figure 14-7 shows the screen display for the trace functions.

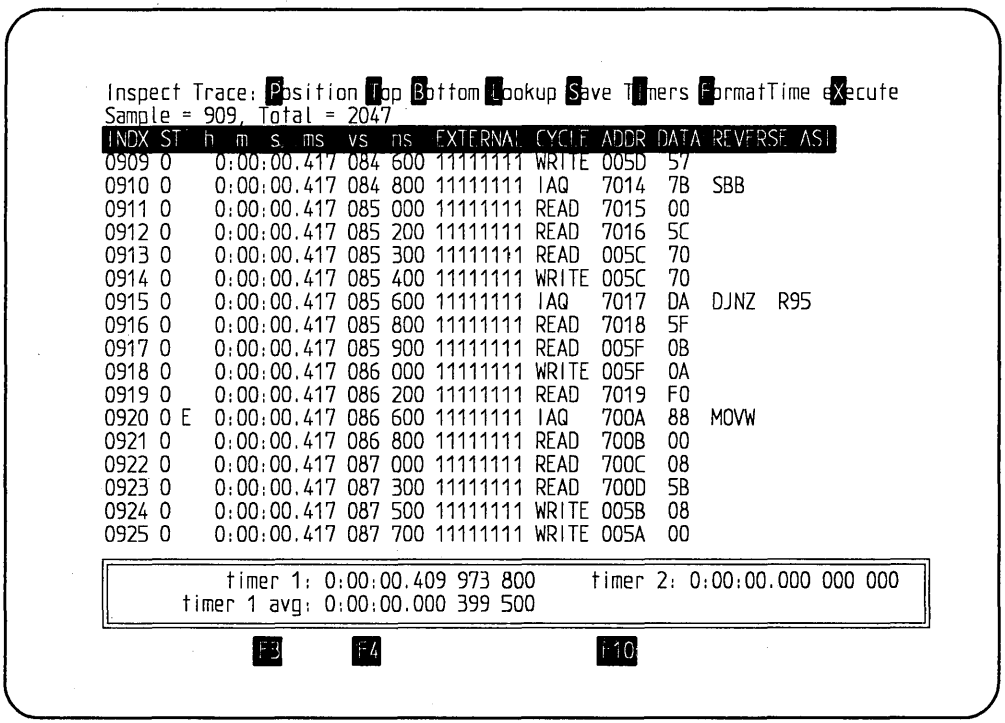


Figure 14-7. Trace Sample Screen

The BTT board has three timers. Two timers are controlled by event qualification, while the third is free running. The timers allow timing statistics to be taken, such as the time the microcontroller spends executing a particular routine. This aids the programmer in developing efficient code and evaluating system performance. The display format of the trace buffer screen can be altered by the user to show one of the following statistics determined by the timers:

- Time stamps referenced from starting time
- Delta or time between trace samples
- Time samples referenced from selected trace sample

14.2.4 XDS System Operating Considerations

The emulation hardware of the XDS System generally exhibits the same characteristics as the actual TMS370 devices. There are, however, a few subtle differences that the designer should be aware of when building a prototype circuit for use with the XDS System.

14.2.4.1 Mode Control Pin

To allow the XDS System to function without being attached to a target system, a 20K ohm pull down resistor should be connected to the mode control line in the XDS unit. This increases the minimum input current needed to drive this line high (I_I) from 100 uA to 300 uA. If a pull up resister is used to put the device in the microcomputer mode, then it's value should be no greater than 1K ohm when using the XDS System.

14.2.4.2 Reset

The XDS System adds an analog switch and a 51K ohm pull up resister to the reset line. This increases the current necessary to pull this line to a logic low from 10 uA to 100 uA.

14.2.4.3 Clock In

The XDS System cannot drive a crystal located on the target system. Therefore, either the crystal must be moved to socket Y1 of the emulator board, or a TTL level external clock be connected.

14.3 The TI EEPROM Programmer

The TI EEPROM Programmer is an interactive, menu driven system that provides a method of programming TMS370 prototyping devices and industry-standard UVEPROMs. The Programmer can interact either directly with a PC or through the XDS for easy programming, modifying, and reading of the target memory device. Sockets are provided for all members of the TMS370 family as well as UVEPROMs such as the 2732, 2764, 27128, and 27256.

The TI EEPROM Programmer system (as shown in Figure 14-8) consists of the TI EEPROM Programmer and an IBM compatible or TI PC running EEPROM programmer software under MS/PC-DOS. The TI EEPROM Programmer comes complete with power and interface cables, EEPROM programmer software for the PC, and a user guide.

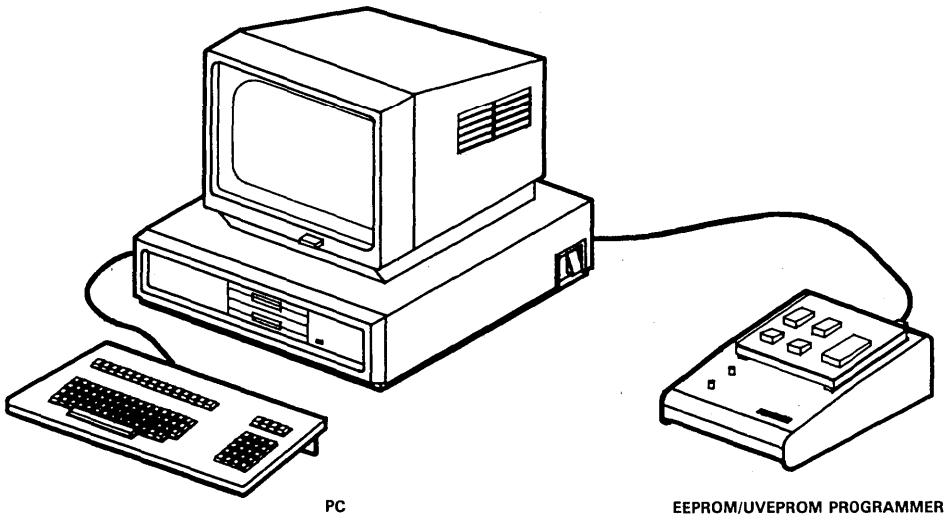


Figure 14-8. Typical EEPROM/UVEPROM Programmer Configuration

The programmer software provides both interactive and limited batch control with the following features:

- Window oriented screens with a menu-driven command structure
- Block erase for TMS370 family devices only
- Programming mode bit selection for TMS370 family devices only
- Relocatable programming capability which allows source data bytes within certain address range to be programmed at specified address
- Reverse assembly code display
- Intermediate PC memory which provides a storage area for downloading a COFF file or uploading from devices
- Ability to inspect and patch loaded data in PC memory
- Ability to generate a COFF file from PC memory content
- Ability to save or load Programmer Configuration to or from Configuration/Batch file

The TI EEPROM Programmer, unlike most other EPROM programmers, can use COFF object files developed by the assembler/linker as input for programming the TMS370 devices. Other EPROM programmers will require that the object files be converted into the Intel hex object format before programming.

14.4 Prototyping/Preproduction Devices

The TMS370C850 and TMS370C810 devices can be used as prototyping devices for the TMS370C050 and TMS370C010 devices respectively. The TMS370C8x0 devices replace the mask ROM with 4 kilobytes of EEPROM. These devices are assembled in the same package types as the masked ROM parts so they can plug into the same target application as the final masked device. The TMS370C850 and TMS370C810 provide form factor preproduction parts with zero leadtime for field testing and production qualifications, thereby reducing the overall time to market. Both TMS370C8x0 devices can be programmed directly from the assembler or linker output file with the TI EEPROM Programmer.

15. Electrical Specifications

This section contains electrical and timing information for the TMS370 family devices. Specifications that apply to the TMS370Cx10 devices are presented first, followed by specifications that apply to the TMS370Cx50 devices. Specifications for the TMS370Cx10 devices generally apply also to the TMS370Cx50 devices, since a TMS370Cx50 device essentially consists of a related TMS370Cx10 plus additional circuitry.

Section **Page**

Caution:

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute maximum rated conditions for extended periods may affect device reliability.

15.1 TMS370Cx10 Specifications

The specifications given in the following tables apply to the TMS370C010 and TMS370C810.

Table 15-1. Absolute Maximum Ratings over Operating Free-Air Temperature Range (unless otherwise noted)

Supply voltage range, V_{CC}† ..... -0.3 V to 7 V
Input voltage range: All pins except MC ..... -0.3 V to V_{CC}+0.3V
Input voltage range: MC ..... -0.3 V to 14 V
Input buffer current ..... ±10 mA
Maximum source current, I_{CC} ..... 170 mA
Maximum drain current, I_{SS} ..... 170 mA
Continuous power dissipation ..... 1 W
Storage temperature range ..... -65°C to 150°C

† Unless otherwise noted, all voltages are with respect to V_{SS}.

Table 15-2. Recommended Operating Conditions

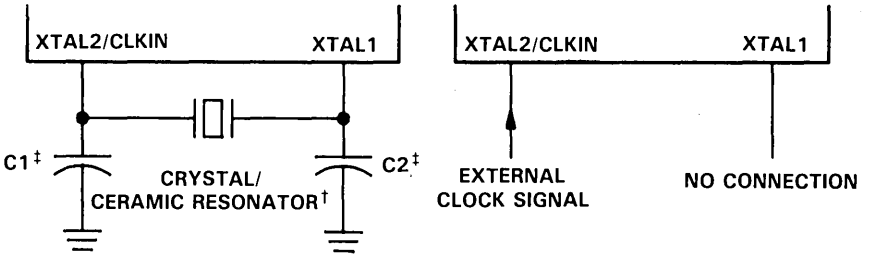
			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage (see Note 1)		4.5	5	5.5	V
V_{CC}	RAM data retention supply voltage		3		5.5	V
V_{IL}	Low-level input voltage	All pins except MC and XTAL2/CLKIN	V_{SS}		0.8	V
		MC,normal operation	V_{SS}		0.3	V
		XTAL2/CLKIN	V_{SS}		0.8	V
V_{IH}	High-level input voltage	All pins except MC and XTAL2/CLKIN	2		V_{CC}	V
		MC/Write Protect Override (WPO)	11.7	12		V
		XTAL2/CLKIN	0.8V_{CC}		V_{CC}	V
		$\overline{RESET}$	0.7V_{CC}		V_{CC}	V
T_A	Operating free-air temperature	A version	−40		85	°C
		L version	0		70	°C

NOTES: 1. All voltage values are with respect to V_{SS}.
2. $\overline{RESET}$ is externally released while V_{CC} is within the recommended operating range of 4.5 V-5.5 V and externally activated when V_{CC}<4.5 V or V_{CC}>5.5 V. RAM data retention is valid throughout the 2 MHz-20 MHz frequency range. An active $\overline{RESET}$ initializes (clears) RAM locations 0000h and 0001h.

Table 15-3. Electrical Characteristics over Full Range of Operating Conditions

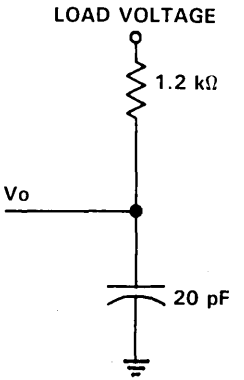
PARAMETER			TEST CONDITIONS	MIN	TYP	MAX	UNIT
V_{OL}	Low-level output voltage		I_{OL} = 1.4 mA			0.4	V
V_{OH}	High-level output voltage		I_{OH} = −50 µA	0.9V_{CC}			V
			I_{OH} = −2 mA	2.4			V
I_I	Input current	MC	0 V ≤ V_I ≤ 12 V			400	µA
		I/O pins	0 V ≤ V_I ≤ V_{CC}			±10	µA
I_{OL}	Low-level output current		V_{OL} = 0.4 V	1.4			mA
I_{OH}	High-level output current		V_{OH} = 0.9V_{CC}	−50			µA
			V_{OH} = 2.4 V	−2			mA
I_{CC}	Supply current (Operating mode) Osc Power bit = 0 (see Note 3)	TMS370C010	Operating, Notes 1 and 2 CLKIN frequency = 20 MHz			48	mA
		TMS370C810				80	
		TMS370C010	Operating, Notes 1 and 2 CLKIN frequency = 12 MHz			33	
		TMS370C810				55.6	
		TMS370C010	Operating, Notes 1 and 2 CLKIN frequency = 2 MHz			14	
		TMS370C810				25	
I_{CC}	Supply current (Standby mode) Osc Power bit = 0 (see Note 4)	TMS370C010	Standby, Notes 1 and 2 CLKIN frequency = 20 MHz			20.8	mA
		TMS370C810				28	
		TMS370C010	Standby, Notes 1 and 2 CLKIN frequency = 12 MHz ns			13.6	
		TMS370C810				18.2	
		TMS370C010	Standby, Notes 1 and 2 CLKIN frequency = 2 MHz			4.6	
		TMS370C810				6	
I_{CC}	Supply current (Standby mode) Osc Power bit = 1 (See Note 5)	TMS370C010	Standby, Notes 1 and 2 CLKIN frequency = 12 MHz			10.8	mA
			Standby, Notes 1 and 2 CLKIN frequency= 20 MHz			3.8	
I_{CC}	Supply current (Halt mode)	TSM370C010	Halt mode, Note 2 XTAL2/CLKIN < 0.2 V			50	µA
		TMS370C810				100	

NOTES: 1. Single chip mode, ports configured as inputs, or outputs with no load. All inputs ≤ 0.2 V or ≥ V_{CC} − 0.2 V.
2. All ports configured as inputs, or outputs with no load. All inputs ≤ 0.2 V or ≥ V_{CC} − 0.2 V.
3. Maximum operating current for TMS370C010 = 1.9(f_X) + 10.2 mA.
   Maximum operating current for TMS370C810 = 3.06(f_X) + 18.9 mA.
4. Maximum standby current for TMS370C010 = 0.9(f_X) + 2.8 mA.
   Maximum standby current for TMS370C810 = 1.2(f_X) + 3.56 mA.
5. Maximum standby current for TMS370C010 = 0.7(f_X) + 2.4 mA.



† The crystal/ceramic resonator frequency is four times the reciprocal of the system clock period.
‡ The values of C1 and C2 should be the values recommended by the crystal/ceramic resonator manufacturer.

Figure 15-1. Recommended Crystal/Clock Connections



CASE 1: $V_O = V_{OH} = 2.4$ V; LOAD VOLTAGE = 0 V
CASE 2: $V_O = V_{OL} = 0.4$ V; LOAD VOLTAGE = 2.1 V

Figure 15-2. Output Loading Circuit for Test

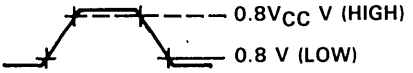


Figure 15-3. XTAL2/CLKIN Measurement Points

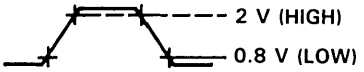


Figure 15-4. General Measurement Points

Table 15-4. External Clocking Requirements

NO.			MIN	NOM	MAX	UNIT
1	$t_{w(CI)}$	XTAL2/CLKIN pulse duration (Note 1)	20			ns
2	$t_{r(CI)}$	XTAL2/CLKIN rise time			30	ns
3	$t_{f(CI)}$	XTAL2/CLKIN fall time			30	ns
4	$t_{d(CIH-COL)}$	Delay time, XTAL2/CLKIN rise to CLKOUT fall			100	ns
	f_x	Crystal operating frequency	2		20	MHz

† For V_{IL} and V_{IH}, refer to "Recommended Operating Conditions".

NOTE 1. 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.

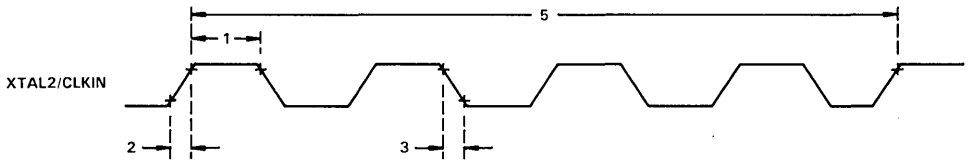


Figure 15-5. External Clock Timing

Table 15-5. General Purpose Output Switching Time Requirements

			MIN	NOM	MAX	UNIT
t_r	Rise time	INT2, INT3, SPISOMI, SPISIMO, SPICLK, T1IC/CR, T1PWM, T1EVT, T2IC1/CR, T2IC2/PWM, T2EVT, SCITXD, SCIRXD, SCICLK			45	ns
t_f	Fall time	INT2, INT3, SPISOMI, SPISIMO, SPICLK, T1IC/CR, T1PWM, T1EVT, T2IC1/CR, T2IC2/PWM, T2EVT, SCITXD, SCIRXD, SCICLK			45	ns

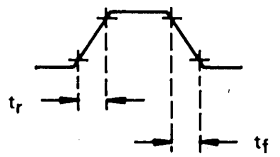


Figure 15-6. Switching Time Measurement Points

Table 15-6. Recommended EEPROM Timing Requirements For Programming

		MIN	NOM	MAX	UNIT
$t_{w(PGM)B}$	Programming signal pulse duration to insure valid data is stored (byte mode)	10			ms
$t_{w(PGM)AR}$	Programming signal pulse duration to insure valid data is stored (array mode)	20			ms

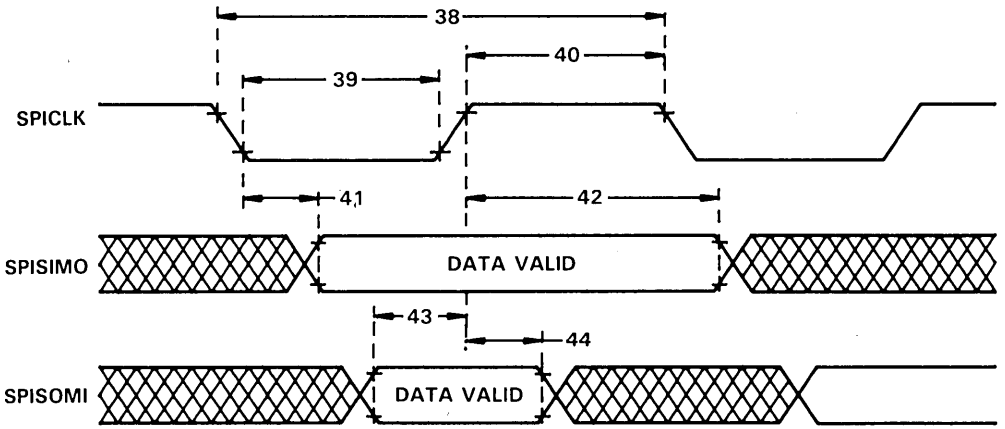
Table 15-7. SPI Master External Timing Characteristics

NO.	PARAMETER		MIN	MAX	UNIT
38	$t_{c(SPC)}$	SPICLK cycle time	$2t_c$	$256t_c$	ns
39	$t_{w(SPCL)}$	SPICLK low pulse duration	t_c-45	$128t_c$	ns
40	$t_{w(SPCH)}$	SPICLK high pulse duration	t_c-45	$128t_c$	ns
41	$t_{d(SPCL-SIMOV)}$	Delay time, SPISIMO valid after SPICLK low (Polarity = 1)	−50	50	ns
42	$t_{v(SPCH-SIMO)}$	SPISIMO data valid after SPICLK high (Polarity = 1)	$t_{w(SPCH)}-50$		ns

Table 15-8. SPI Master External Timing Requirements

NO.			MIN	MAX	UNIT
43	$t_{su(SOMI-SPCH)}$	SPISOMI setup time to SPICLK high (Polarity = 1)	$.25t_c+150$		ns
44	$t_{v(SPCH-SOMI)}$	SPISOMI data valid after SPICLK high (Polarity = 1)	0		ns

NOTE 1. t_c = system clock cycle time = $4/f_x$.



NOTE 12. The diagram above is for Polarity=1. SPICLK is inverted from above diagram when Polarity=0.

15

Figure 15-7. SPI Master External Timing

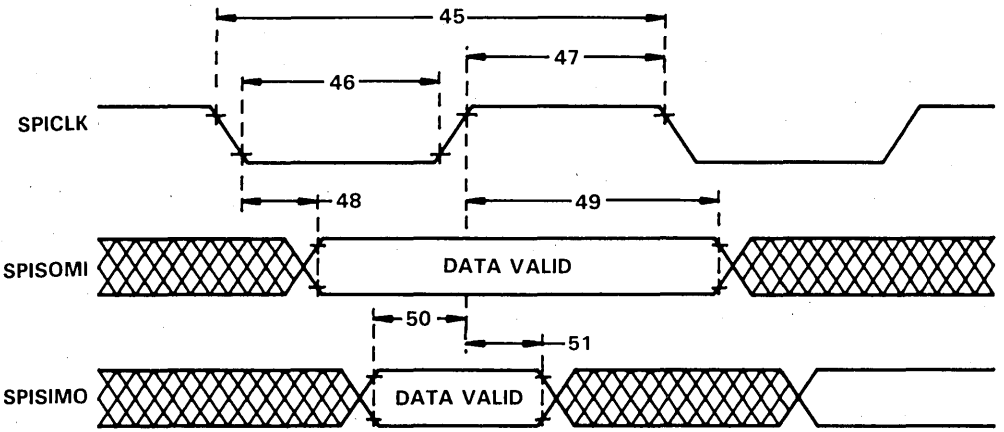
Table 15-9. SPI Slave External Timing Characteristics

NO.	PARAMETER		MIN	MAX	UNIT
48	$t_{d(SPCL-SOMIV)S}$	Delay time, SPISOMI valid after SPICLK low (Polarity = 1)		$3.25t_c+100$	ns
49	$t_{v(SPCH-SOMI)S}$	SPISOMI data valid after SPICLK high (Polarity = 1)	$t_{w(SPCH)S}$		ns

Table 15-10. SPI Slave External Timing Requirements

NO.			MIN	MAX	UNIT
45	$t_{c(SPC)S}$	SPICLK cycle time	$8t_c$		ns
46	$t_{w(SPCL)S}$	SPICLK low pulse duration	$4t_c-45$		ns
47	$t_{w(SPCH)S}$	SPICLK high pulse duration	$4t_c-45$		ns
50	$t_{su(SIMO-SPCH)S}$	SPISIMO setup time to SPICLK high (Polarity = 1)	0		ns
51	$t_{v(SPCH-SIMO)S}$	SPISIMO data valid after SPICLK high (Polarity = 1)	$3t_c+100$		ns

NOTE 1. t_c = system clock cycle time = $4/f_x$.



NOTE 12. The diagram above is for Polarity=1. SPICLK is inverted from above diagram when Polarity=0.
NOTE 13. As a slave, the SPICLK pin is used as the input for the serial clock, which is supplied from the network master.

15

Figure 15-8. SPI Slave External Timing

15.2 TMS370Cx50 Specifications

The specifications given in the following tables apply to the devices in the TMS370Cx50 catagory.

Table 15-11. Absolute Maximum Ratings over Operating Free-Air Temperature Range (unless otherwise noted)

Supply voltage range, V_{CC}† ..... -0.3 V to 7 V
Supply voltage range for digital I/O, V_{CC2}† ..... -0.3 V to 7 V
Supply voltage range for analog, V_{CC3}† ..... -0.3 V to 7 V
Reference voltage range, V_{ref} (non-V_{CC3} reference for A/D) ..... V_{SS3}-0.1 V to V_{CC3}+0.1 V
Input voltage range: All pins except MC ..... -0.3 V to V_{CC}+0.3V
Input voltage range: MC ..... -0.3 V to 14 V
Input buffer current ..... ±10 mA
Maximum source current, I_{CC} ..... 170 mA
Maximum drain current, I_{SS} ..... 170 mA
Continuous power dissipation ..... 1 W
Storage temperature range ..... -65°C to 150°C

† Unless otherwise noted, all voltages are with respect to V_{SS}.

Table 15-12. Recommended Operating Conditions

			MIN	NOM	MAX	UNIT
V_{CC1}	Digital logic supply voltage (Note 1)		4.5	5	5.5	V
V_{CC1}	RAM data retention supply voltage		3		5.5	V
V_{CC2}	Digital I/O supply voltage (Note 1)		4.5	5	5.5	V
V_{CC3}	Analog supply voltage (Note 1)		4.5	5	5.5	V
V_{IL}	Low-level input voltage	All pins except MC and XTAL2/CLKIN	V_{SS}		0.8	V
		MC	V_{SS}		0.3	V
		XTAL2/CLKIN	V_{SS}		0.8	V
V_{IH}	High-level input voltage	All pins except MC and XTAL2/CLKIN	2		V_{CC}	V
		MC	V_{CC}-0.3		V_{CC}	V
		XTAL2/CLKIN	0.8V_{CC}		V_{CC}	V
		$\overline{RESET}$	0.7V_{CC}		V_{CC}	V
	MC (mode control) voltage (Note 2)	EEPROM write protect override	11.7	12		V
		Microprocessor	V_{CC}-0.3			V
		Microcomputer	V_{CC}-0.3			V
T_A	Operating free-air temperature	A version	−40		85	°C
		L version	0		70	°C

NOTES: 1. All voltage values are with respect to V_{SS}.
2. The hardware protect ovcerride, microprocessor, or microcomputer mode can be selected only while $\overline{RESET}$ is high (active).
3. $\overline{RESET}$ is externally released while V_{CC} is within the recommended operating range of 4.5 V-5.5 V and externally activated when V_{CC}<4.5 V or V_{CC}>5.5 V. RAM data retention is valid throughout the 2 MHz-20 MHz frequency range. An active $\overline{RESET}$ initializes (clears) RAM locations 0000h and 0001h.

15

Table 15-13. Electrical Characteristics over Full Range of Operating Conditions

PARAMETER			TEST CONDITIONS	MIN	TYP	MAX	UNIT
V_{OL}	Low-level output voltage	Ports A, B, C, and D, and $\overline{RESET}$	I_{OL} = 2 mA			0.4	V
		Other outputs	I_{OL} = 1.4 mA			0.4	V
V_{OH}	High-level output voltage		I_{OH} = −50 µA	0.9V_{CC}			V
			I_{OH} = −2 mA	2.4			V
I_I	Input current	MC	0 V ≤ V_I ≤ 12 V			400	µA
		I/O pins	0 V ≤ V_I ≤ V_{CC}			±10	µA
I_{OL}	Low-level output current	Ports A, B, C, and D, and $\overline{RESET}$	V_{OL} = 0.4 V	2			mA
		Other outputs	V_{OL} = 0.4 V	1.4			mA
I_{OH}	High-level output current		V_{OH} = 0.9V_{CC}	−50			µA
			V_{OH} = 2.4 V	−2			mA
I_{CC}	Supply current (Operating mode) Osc Power bit = 0 (see Note 4)	TMS370C050	Operating, Notes 1 and 3 CLKIN frequency = 20 MHz			67	mA
		TMS370C850				80	
		TMS370C050	Operating, Notes 1 and 2 CLKIN frequency = 12 MHz			46.2	
		TMS370C850				55.6	
		TMS370C050	Operating, Notes 1 and 3 CLKIN frequency = 2 MHz			20	
		TMS370C850				25	
I_{CC}	Supply current (Standby mode) Osc Power bit = 0 (see Note 5)	TMS370C050	Standby, Notes 2 and 3 CLKIN frequency = 20 MHz			20.8	mA
		TMS370C850				28	
		TMS370C050	Standby, Notes 2 and 3 CLKIN frequency = 12 MHz			13.6	
		TMS370C850				18.2	
		TMS370C050	Standby, Notes 2 and 3 CLKIN frequency = 2 MHz			4.6	
		TMS370C850				6	
I_{CC}	Supply current (Standby mode) Osc Power bit = 1 (See Note 6)	TMS370C050	Standby, Notes 2 and 3 CLKIN frequency = 12 MHz			10.8	mA
			Standby, Notes 2 and 3 CLKIN frequency = 2 MHz			3.8	
I_{CC}	Supply current (Halt mode)	TSM370C050	Halt mode, Note 2 XTAL2/CLKIN < 0.2 V			50	µA
		TMS370C850				100	

NOTES: 1. Single chip mode, ports configured as inputs, or outputs with no load. All inputs ≤ 0.2 V or ≥ V_{CC} − 0.2 V.
2. All ports configured as inputs, or outputs with no load. All inputs ≤ 0.2 V or ≥ V_{CC} − 0.2 V.
3. XTAL2/CLKIN is driven with an external square wave signal with 50% duty cycle and rise and fall times less than 10 ns. Currents may be higher with a crystal oscillator. At 20 MHz this extra current = 0.1 mA × (total load capacitance + crystal capacitance in pF).
4. Maximum operating current for TMS370C050 = 2.6(f_X) + 15 mA.
   Maximum operating current for TMS370C850 = 3.06(f_X) + 18.9 mA.
5. Maximum standby current for TMS370C050 = 0.9(f_X) + 2.8 mA.
   Maximum standby current for TMS370C850 = 1.2(f_X) + 3.56 mA.
6. Maximum standby current for TMS370C050 = 0.7(f_X) + 2.4 mA. (Osc Power bit valid only from 2 MHz to 12 MHz.)

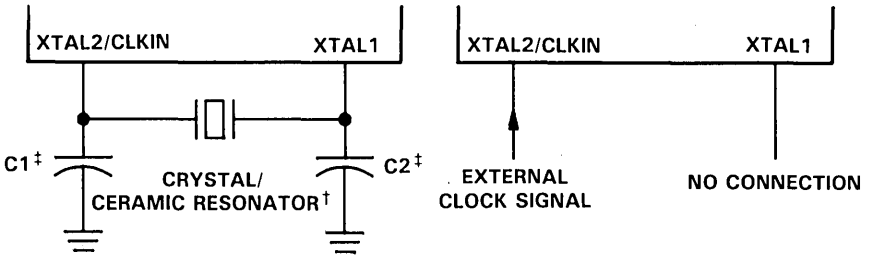
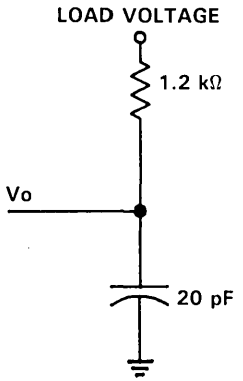


Figure 15-9. Recommended Crystal/Clock Connections



CASE 1: $V_O = V_{OH} = 2.4$ V; LOAD VOLTAGE = 0 V
CASE 2: $V_O = V_{OL} = 0.4$ V; LOAD VOLTAGE = 2.8 V FOR PORTS A, B, C, and D, and $\overline{\text{RESET}}$
LOAD VOLTAGE = 2.1 V FOR OTHER OUTPUTS

Figure 15-10. Output Loading Circuit for Test

15

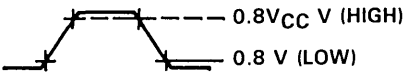


Figure 15-11. XTAL2/CLKIN Measurement Points

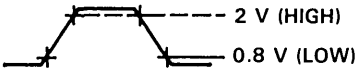


Figure 15-12. General Measurement Points

Table 15-14. External Clocking Requirements

NO.			MIN	NOM	MAX	UNIT
1	$t_{w(CI)}$	XTAL2/CLKIN pulse duration (Note 1)	20			ns
2	$t_{r(CI)}$	XTAL2/CLKIN rise time			30	ns
3	$t_{f(CI)}$	XTAL2/CLKIN fall time			30	ns
4	$t_{d(CIH-COL)}$	Delay time, XTAL2/CLKIN rise to CLKOUT fall			100	ns
	f_x	Crystal operating frequency	2		20	MHz

† For V_{IL} and V_{IH}, refer to "Recommended Operating Conditions".

NOTE 1. 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.

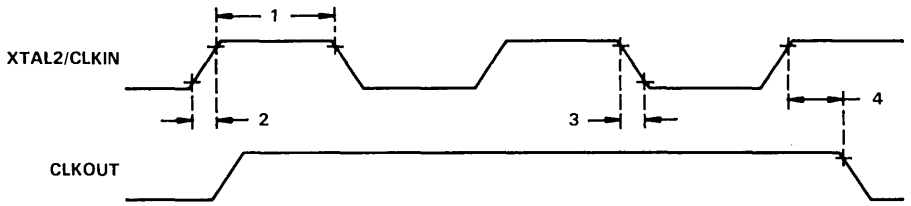


Figure 15-13. External Clock Timing

Table 15-15. Peripheral Module and General Purpose Output Switching Times

			MIN	NOM	MAX	UNIT
t_r	Rise time	INT2, INT3, SPISOMI, SPISIMO, SPICLK, T1IC/CR, T1PWM, T1EVT, T2IC1/CR, T2IC2/PWM, T2EVT, SCITXD, SCIRXD, SCICLK			45	ns
t_f	Fall time	INT2, INT3, SPISOMI, SPISIMO, SPICLK, T1IC/CR, T1PWM, T1EVT, T2IC1/CR, T2IC2/PWM, T2EVT, SCITXD, SCIRXD, SCICLK			45	ns

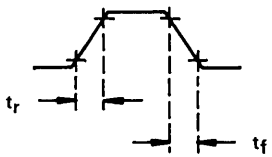


Figure 15-14. Switching Time Measurement Points

Table 15-16. Recommended EEPROM Timing Requirements For Programming

		MIN	NOM	MAX	UNIT
$t_{w(PGM)B}$	Programming signal pulse duration to insure valid data is stored (byte mode)	10			ms
$t_{w(PGM)AR}$	Programming signal pulse duration to insure valid data is stored (array mode)	20			ms

Table 15-17. Switching Characteristics and Timing Requirements

NO.			MIN	MAX	UNIT
5	t_c	CLKOUT (system clock) cycle time	200	2000	ns
6	$t_{w(COL)}$	CLKOUT low pulse duration	$0.5t_c-20$	$0.5t_c$	ns
7	$t_{w(COH)}$	CLKOUT high pulse duration	$0.5t_c$	$0.5t_c+20$	ns
8	$t_{d(COL-A)}$	Delay time, CLKOUT low to address, R/$\overline{W}$, and $\overline{OCF}$ valid		$0.25t_c+40$	ns
9	$t_{v(A)}$	Address valid to $\overline{EDS}$, $\overline{CSE1}$, $\overline{CSE2}$, $\overline{CSH1}$, $\overline{CSH2}$, $\overline{CSH3}$, and $\overline{CSPF}$ low	$0.5t_c-50$		ns
10	$t_{su(D)}$	Write data setup time to $\overline{EDS}$ high	$0.75t_c-40$†		ns
11	$t_{h(EH-A)}$	Address, R/$\overline{W}$, and $\overline{OCF}$ hold time from $\overline{EDS}$, $\overline{CSE1}$, $\overline{CSE2}$, $\overline{CSH1}$, $\overline{CSH2}$, $\overline{CSH3}$, and $\overline{CSPF}$ high	$0.5t_c-40$		ns
12	$t_{h(EH-D)W}$	Write data hold time from $\overline{EDS}$ high	$0.75t_c+15$		ns
13	$t_{d(DZ-EL)}$	Delay time, data bus high impedance to $\overline{EDS}$ low (read cycle)	$0.25t_c-30$		ns
14	$t_{d(EH-D)}$	Delay time, $\overline{EDS}$ high to data bus enable (read cycle)	$1.25t_c-40$		ns
15	$t_{d(EL-DV)}$	Delay time, $\overline{EDS}$ low to read data valid		t_c-65†	ns
16	$t_{h(EH-D)R}$	Read data hold time from $\overline{EDS}$ high	0		ns
17	$t_{su(WT-COH)}$	$\overline{WAIT}$ setup time to CLKOUT high	$0.25t_c+75$‡		ns
18	$t_{h(COH-WT)}$	$\overline{WAIT}$ hold time from CLKOUT high	0		ns
19	$t_{d(EL-WTV)}$	Delay time, $\overline{EDS}$ low to $\overline{WAIT}$ valid		$0.5t_c-70$	ns
20	t_w	Pulse duration; $\overline{EDS}$, $\overline{CSE1}$, $\overline{CSE2}$, $\overline{CSH1}$, $\overline{CSH2}$, $\overline{CSH3}$, and $\overline{CSPF}$ low	t_c-40†	t_c+40†	ns
21	$t_{d(AV-DV)R}$	Delay time, address valid to read data valid		$1.5t_c-75$†	ns
22	$t_{d(AV-WTV)}$	Delay time, address valid to $\overline{WAIT}$ valid		t_c-85	ns
23	$t_{d(AV-EH)}$	Delay time, address valid to $\overline{EDS}$ high (end of write)	$1.5t_c-40$†		ns

† If wait states, PFWait, or the Auto-Wait feature are used, add t_c to this value for each wait state invoked.
‡ If the Auto-Wait feature is enabled, the $\overline{WAIT}$ input may assume a "Don't Care" condition until the third cycle of the access.

NOTE 1. t_c = system clock cycle time = $4/f_x$.

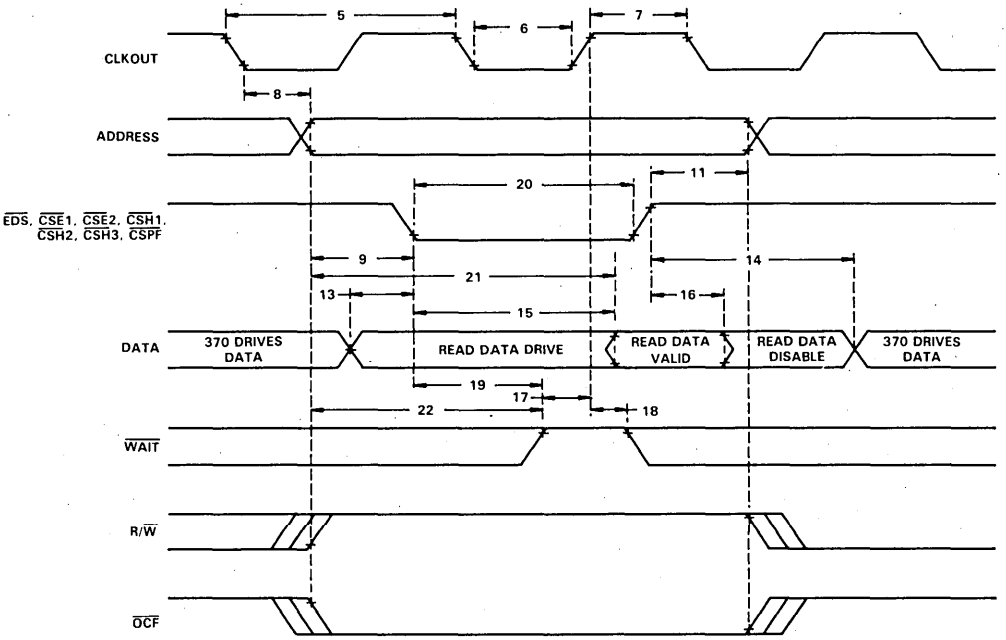


Figure 15-15. External Read Timing

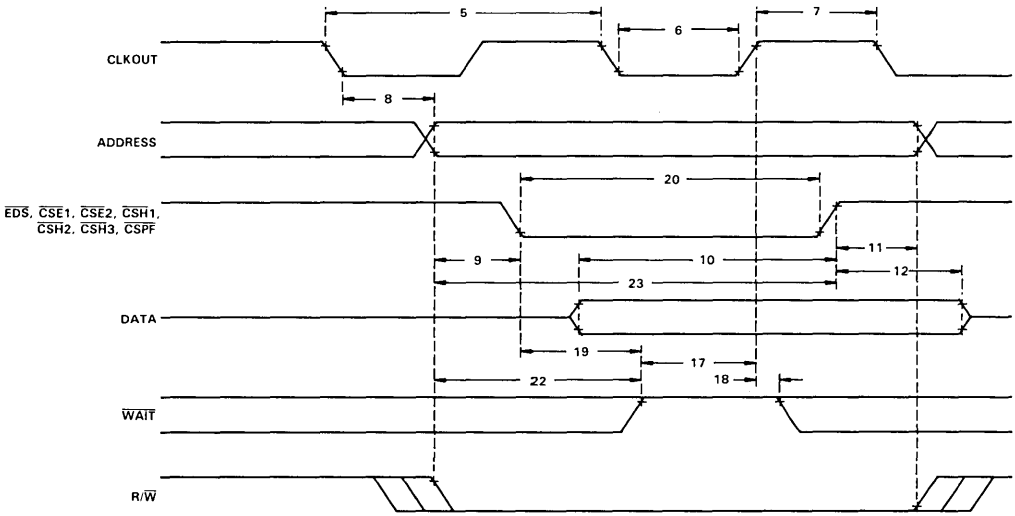


Figure 15-16. External Write Timing

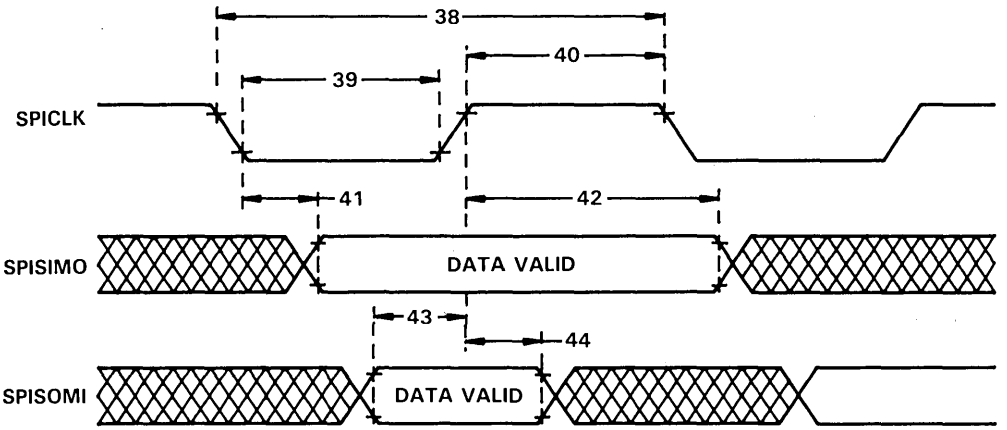
Table 15-18. SPI Master External Timing Characteristics

NO.	PARAMETER		MIN	MAX	UNIT
38	$t_{c(SPC)}$	SPICLK cycle time	$2t_c$	$256t_c$	ns
39	$t_{w(SPCL)}$	SPICLK low pulse duration	t_c-45	$128t_c$	ns
40	$t_{w(SPCH)}$	SPICLK high pulse duration	t_c-45	$128t_c$	ns
41	$t_{d(SPCL-SIMOV)}$	Delay time, SPISIMO valid after SPICLK low (Polarity = 1)	−50	50	ns
42	$t_{v(SPCH-SIMO)}$	SPISIMO data valid after SPICLK high (Polarity = 1)	$t_{w(SPCH)}-50$		ns

Table 15-19. SPI Master External Timing Requirements

NO.			MIN	MAX	UNIT
43	$t_{su(SOMI-SPCH)}$	SPISOMI setup time to SPICLK high (Polarity = 1)	$.25t_c+150$		ns
44	$t_{v(SPCH-SOMI)}$	SPISOMI data valid after SPICLK high (Polarity = 1)	0		ns

NOTE 1. t_c = system clock cycle time = $4/f_x$.



NOTE 15. The diagram above is for Polarity=1. SPICLK is inverted from above diagram when Polarity=0.

15

Figure 15-17. SPI Master External Timing

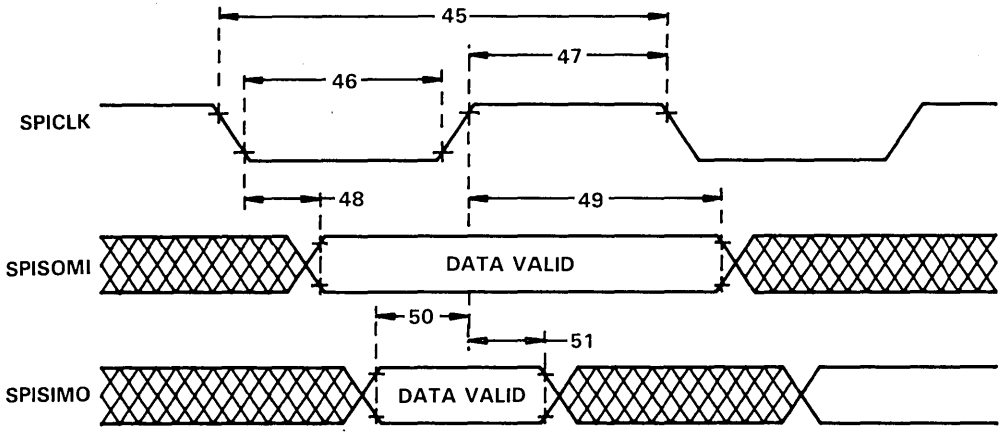
Table 15-20. SPI Slave External Timing Characteristics

NO.	PARAMETER		MIN	MAX	UNIT
48	$t_{d(SPCL-SOMIV)S}$	*Delay time, SPISOMI valid after SPICLK low (Polarity = 1)*		$3.25t_c+100$	ns
49	$t_{v(SPCH-SOMI)S}$	SPISOMI data valid after SPICLK high (Polarity = 1)	$t_{w(SPCH)S}$		ns

Table 15-21. SPI Slave External Timing Requirements

NO.			MIN	MAX	UNIT
45	$t_{c(SPC)S}$	SPICLK cycle time	$8t_c$		ns
46	$t_{w(SPCL)S}$	SPICLK low pulse duration	$4t_c-45$		ns
47	$t_{w(SPCH)S}$	SPICLK high pulse duration	$4t_c-45$		ns
50	$t_{su(SIMO-SPCH)S}$	SPISIMO setup time to SPICLK high (Polarity = 1)	0		ns
51	$t_{v(SPCH-SIMO)S}$	SPISIMO data valid after SPICLK high (Polarity = 1)	$3t_c+100$		ns

NOTE 1. t_c = system clock cycle time = $4/f_x$.



NOTE 15. The diagram above is for Polarity=1. SPICLK is inverted from above diagram when Polarity=0.
NOTE 16. As a slave, the SPICLK pin is used as the input for the serial clock, which is supplied from the network master.

15

Figure 15-18. SPI Slave External Timing

Table 15-22. SCI Isosynchronous Mode Timing Charateristics For Internal Clock

NO.	PARAMETER		MIN	MAX	UNIT
24	$t_{c(SCC)}$	SCICLK cycle time	$2t_c$	$131{,}072t_c$	ns
25	$t_{w(SCCL)}$	SCICLK low pulse duration	t_c-45	$65{,}536t_c$	ns
26	$t_{w(SCCH)}$	SCICLK high pulse duration	t_c-45	$65{,}536t_c$	ns
27	$t_{d(SCCL-TXDV)}$	Delay time, SCITXD valid after SCICLK low	−50	50	ns
28	$t_{v(SCCH-TXD)}$	SCITXD data valid after SCICLK high	$t_{w(SCCH)}-50$		ns

Table 15-23. SCI Isosynchronous Mode Timing Requirements For Internal Clock

NO.			MIN	MAX	UNIT
29	$t_{su(RXD-SCCH)}$	SCIRXD setup time to SCICLK high	$0.25t_c+145$		ns
30	$t_{v(SCCH-RXD)}$	SCIRXD data valid after SCICLK high	0		ns

NOTE 1. t_c = system clock cycle time = $4/f_x$.

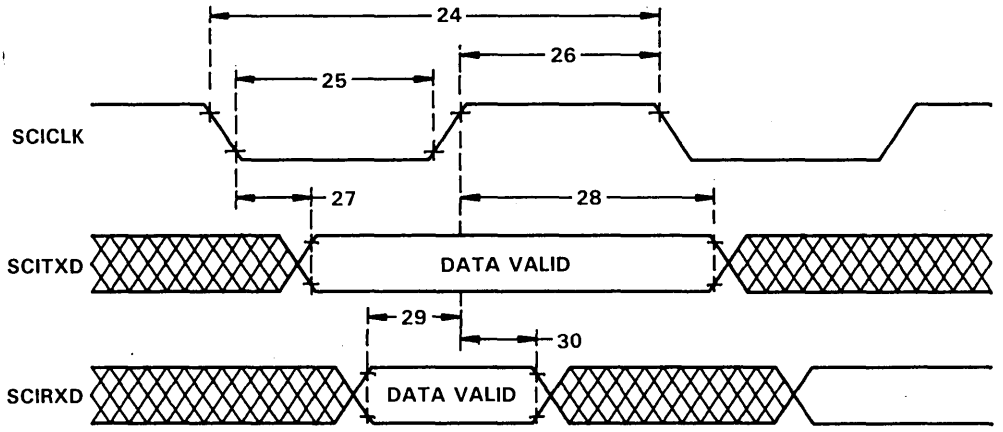


Figure 15-19. SCI Isosynchronous Mode Timing For Internal Clock

Table 15-24. SCI Isosynchronous Mode Timing Charateristics For External Clock

NO.	PARAMETER		MIN	MAX	UNIT
34	$t_{d(SCCL-TXDV)}$	Delay time, SCITXD valid after SCICLK low		$4.25t_C+145$	ns
35	$t_{v(SCCH-TXD)}$	SCITXD data valid after SCICLK high	$t_{w(SCCH)}$		ns

Table 15-25. SCI Isosynchronous Mode Timing Requirements For External Clock

NO.			MIN	MAX	UNIT
31	$t_{c(SCC)}$	SCICLK cycle time	$10t_C$		ns
32	$t_{w(SCCL)}$	SCICLK low pulse duration	$4.25t_C+120$		ns
33	$t_{w(SCCH)}$	SCICLK high pulse duration	t_C+120		ns
36	$t_{su(RXD-SCCH)}$	SCIRXD setup time to SCICLK high	40		ns
37	$t_{v(SCCH-RXD)}$	SCIRXD data valid after SCICLK high	$2t_C$		ns

NOTE 1. t_C = system clock cycle time = $4/f_X$.

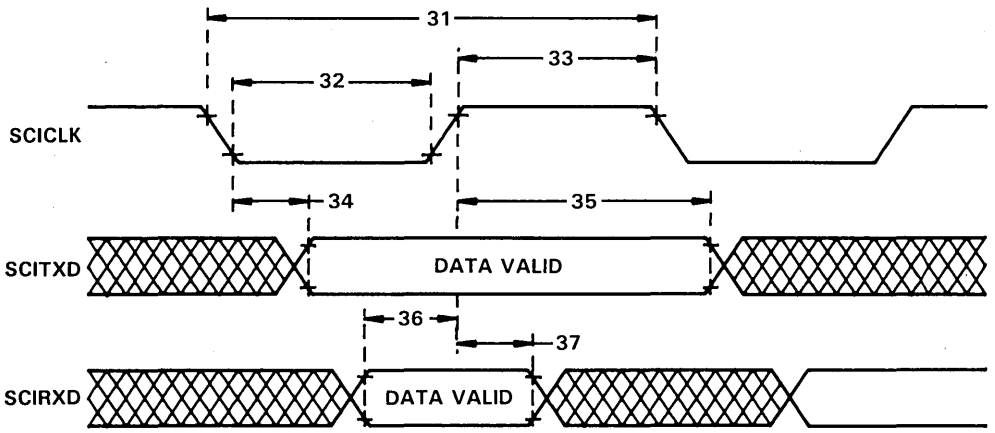


Figure 15-20. SCI Isosynchronous Mode Timing For External Clock

Table 15-26. A/D Converter Recommended Operating Conditions

		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 (Note 1)	2.5	V_{CC3}	$V_{CC3}+0.1$	V
	Analog input for conversion	V_{SS3}		V_{ref}	V

NOTE 1. V_{ref} must be stable, within ±1/2 LSB of the required resolution, during the entire conversion time.

Table 15-27. A/D Converter Operating Characteristics Over Full Range Of Operating Conditions

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
	Absolute accuracy (Note 1)	V_{CC3} = 5.5 V, V_{ref} = 5.1 V			±1	LSB
	Differential/integral linearity error (Note 2)	2.5 V < V_{ref} < 5.8 V			±0.5	LSB
I_{CC3}	Analog supply current	Converting			2	mA
		Not converting			5	µA
I_I	Input current, AN0-AN7	0 V ≤ V_I ≤ 5.5 V			2	µA
	V_{ref} input charge current				1	mA
Z_{ref}	Input impedance of V_{ref}	XTAL2/CLKIN ≤ 12 MHz	24			kΩ
		12 MHz<XTAL2/CLKIN≤20 MHz	10			kΩ
	Conversion time (excluding sample time)		$164t_c$			ns

NOTES: 1. Absolute resolution = 20 mV. At V_{ref} = 5.1 V, this is 1 LSB. As V_{ref} decreases, LSB size decreases and thus absolute error in terms of LSBs increases.
2. Excluding quantization error of 1/2 LSB.

Table 15-28. Analog Timing Requirements

		MIN	NOM	MAX	UNIT
$t_{su(S)}$	Analog input setup to sample command	0			ns
$t_{h(AN)}$	Analog input hold from start of conversion	$18t_c$			ns
$t_{w(S)}$	Duration of sample time per kilohm of source impedance (Note 1)	1			µs/kΩ

NOTE 1. The value given is valid for a signal with a source impedance greater than 1 kΩ. If the source impedance is less than 1 kΩ, use a minimum sampling time of 1 µs.

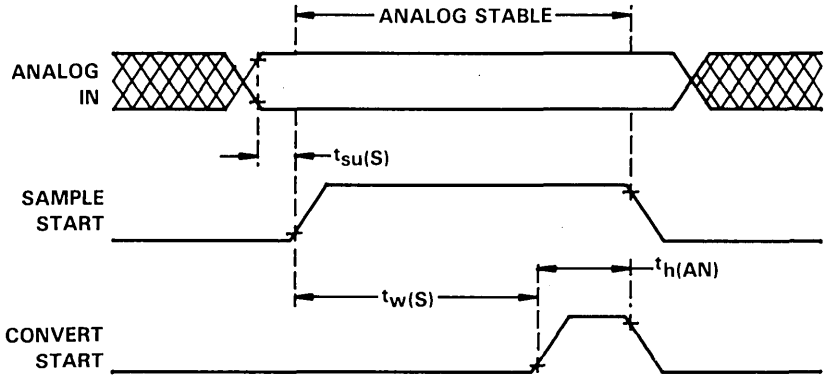


Figure 15-21. Analog Timing

16. Customer Information

This section includes general information on mask-ROM prototyping, TMS370 physical characteristics, and parts ordering. Topics covered in this section include:

16.1 Mask ROM Prototype and Production Flow

The TMS370 family includes two mask-ROM microcontrollers; the TMS370C010 and the TMS370C050. The ROM is manufactured containing customer's application code. The custom-programmed nature of these devices requires a standard, defined interface between the customer and the factory during production. Figure 16-1 shows this standard of prototype/production flow for customer ROM receipt.

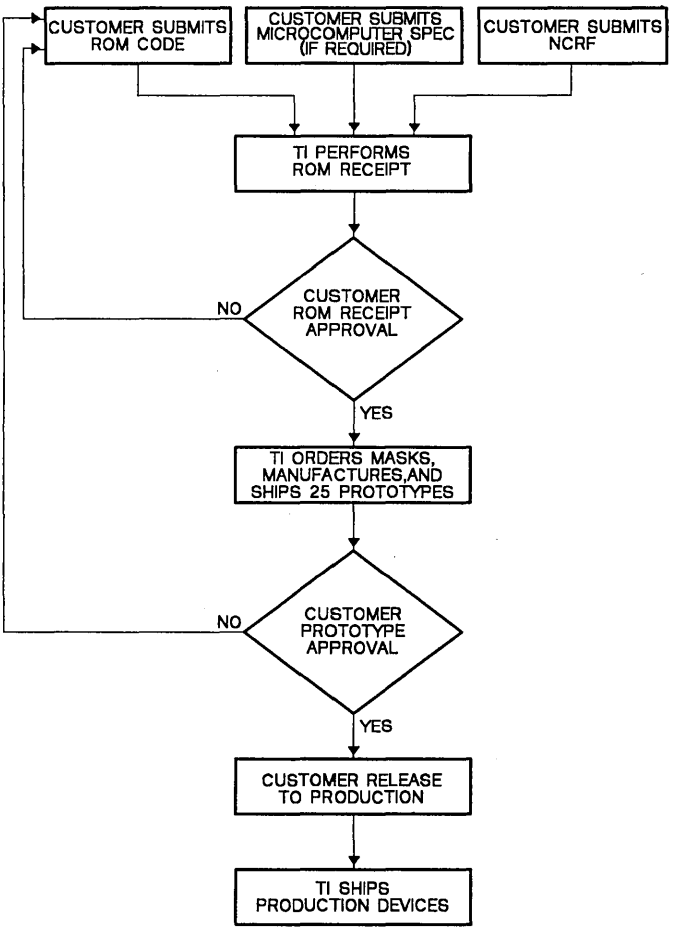


Figure 16-1. Prototype and Production Flow

1) Customer Required Information

For TI to accept the receipt of a customer ROM algorithm, each of the following three items must be received by the TI factory:

a) The customer completes and submits a New Code Release Form (NCRF – available from TI Field Sales Office) describing the custom features of the device (e.g., customer information, prototype and production quantities and dates, any exceptions to standard electrical specifications, customer part numbers and symbols, package type, etc.).

b) If non-standard specifications are requested on the NCRF then the customer submits a copy of the discription of the microcomputer, including the functional description and electrical specification (including absolute maximum ratings, recommended operating conditions, and timing values). TI will then respond to the requested specification changes.

c) When the customer has completed code development and has verified the code with the development system, the object file is submitted in Intel hex object format to the TI factory using an acceptable transfer media. Acceptable media include the following:

- Modem transfer: PC-to-PC via Xmodem, Ymodem, or Zmodem protocol or Microstuf's Crosstalk XVI protocol.
- MS-DOS formatted 5 1/4" floppy disk compatible with IBM or TI PC
- EPROM devices (currently supported: TMS2764, TMS27C64, TMS27128, TMS27C128).
- TMS370C8x0 EEPROM devices.

The completed NCRF, customer specification (if required), and ROM code should be given to the local representive or sent to the nearest Field Sales Office.

2) TI Performs ROM Receipt

Code review and ROM receipt is performed on the customer's code and a unique manufacturing ROM code number (such as R150x123FN) is assigned to the customer's algorithm. All future correspondence should indicate this number. The ROM receipt procedure reads the ROM code information, processes it, reproduces the customer's ROM object code on the media requested on the NCRF, and returns the processed and the original code to the customer for verification of correct ROM receipt. (Note: The customer must provide the EPROM/EEPROM device if that type of media has been requested on the NCRF). All TMS370 mask ROM devices contain ROM space that is reserved for TI use only. The contents of this reserved space is changed when TI processes the mask ROM with the customer's object code. Therefore, the customer should not use locations 7FE0h through 7FEBh in their algorithm or checksum routine.

3) Customer ROM Receipt Approval

The customer then verifies that the ROM code received and processed by TI is correct and that no information was misinterpreted in the transfer. The customer must then return an algorithm approval form (available from the field sales office) for correct ROM receipt verification or resubmit the code for processing. This written confirmation of verification constitutes the contractual agreement for creation of the custom mask and manufacture of ROM verification prototype units.

4) TI Orders Masks, Manufacturing, and Ships 25 Prototypes

TI generates the prototype photomasks, processes, manufactures, and tests 25 microcomputer prototypes containing the customer's ROM pattern for shipment to the customer for ROM code verification. These microcomputer devices have been made using the custom mask but are for the purposes of ROM verification only. Prototype devices are symbolized with a **P** preceding the manufacturing ROM code number (eg., PR150x123FN) to differentiate them from production devices.

5) Customer Prototype Approval

The customer verifies the operation of these prototypes in the system and responds with written customer prototype approval or disapproval. This written customer prototype approval constitutes the contractual agreement to initiate volume microcomputer production using the verified prototype ROM code.

6) Customer Release to Production

With customer algorithm approval, the ROM code is released to production and TI will begin shipment of production devices according to customer's final specification and order requirements.

Two lead times are quoted in reference to the preceding flow:

- Prototype lead time – elapsed time from the receipt of written ROM receipt verification to the delivery of 25 prototype devices.
- Production lead time – elapsed time from the receipt of written customer prototype approval to delivery of production devices.

For the latest TMS370 family lead times, contact the nearest TI field sales office.

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 therefore not be used in the customer's software algorithm, nor should it be used during mask ROM/firmware development. **The reserve location contents are changed by TI.**

16.2 Mechanical Package Information

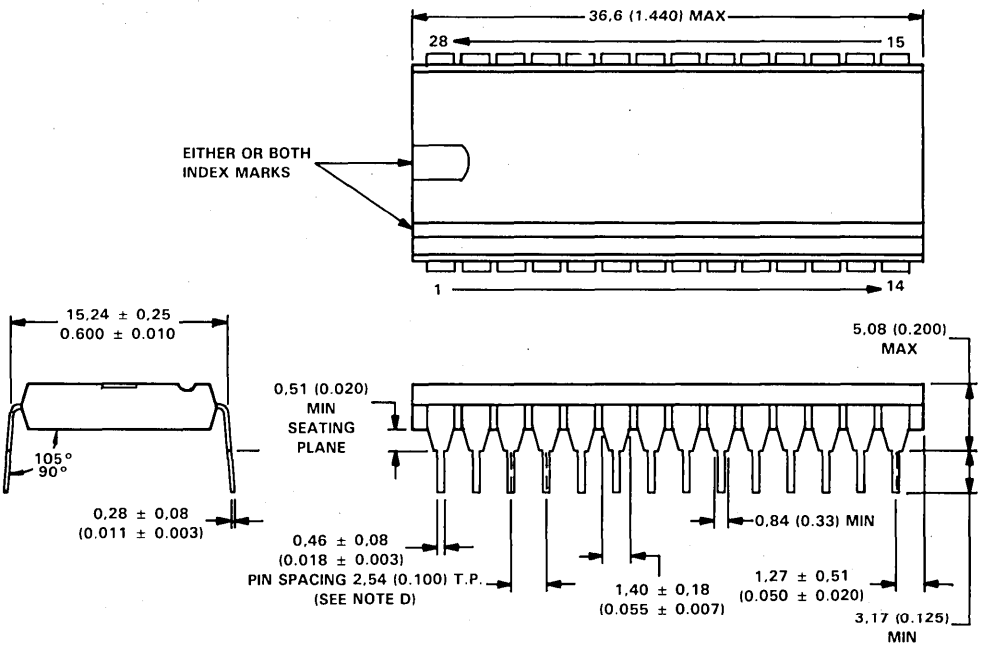
The TMS370 microcomputer family devices are assembled in two package types according to the type of material and outline used for the package. These package types are:

- Plastic dual-inline package (DIP)
- Plastic leaded chip carrier (PLCC)

Package types are designated in the device symbol by the suffix on the customer's ROM code number for devices manufactured with customer ROM code (eg., R150x123FN) and by the suffix of the standard device number for devices with EEPROM. Table 16-1 indicates the package type, suffix indicator, and family members supported on that package type.

Table 16-1. Package Types

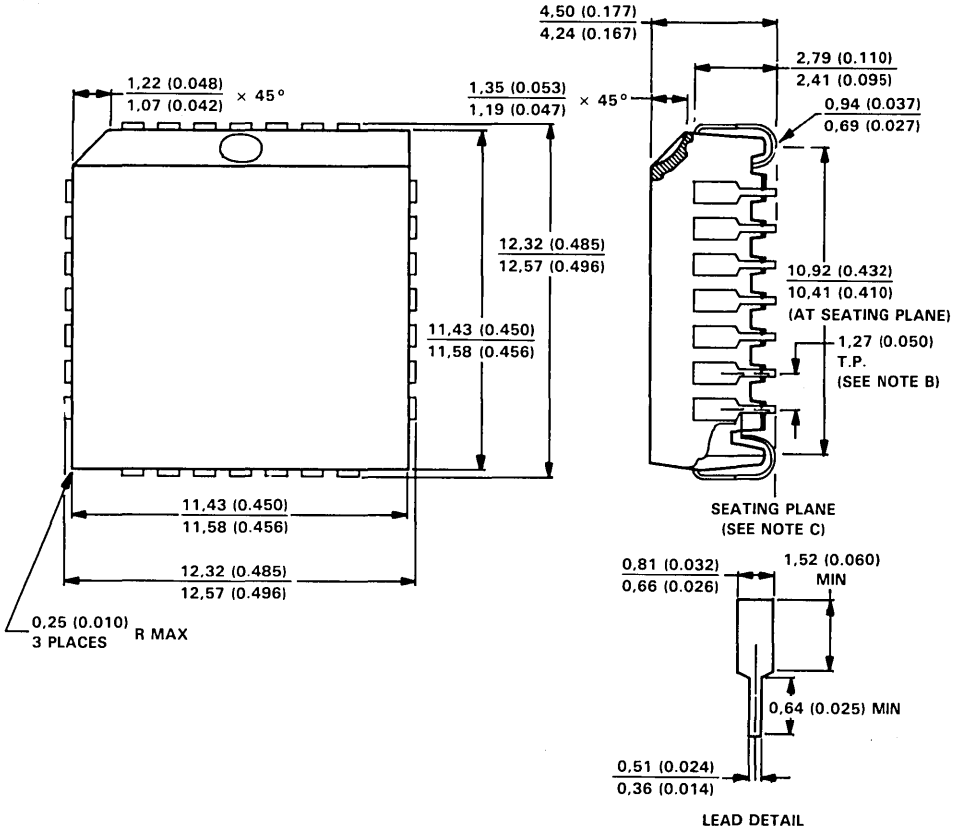
PACKAGE TYPE	SUFFIX INDICATOR	FAMILY MEMBERS
28-pin plastic DIP (100-mil pin spacing)	N	TMS370C010, TMS370C810
28-pin PLCC (50-mil pin spacing)	FN	TMS370C010, TMS370C810
68-pin PLCC (50-mil pin spacing)	FN	TMS370C050, TMS370C850



ALL LINEAR DIMENSIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES

NOTE D. Each pin centerline is located within 0,25 (0.010) of its true longitudinal position.

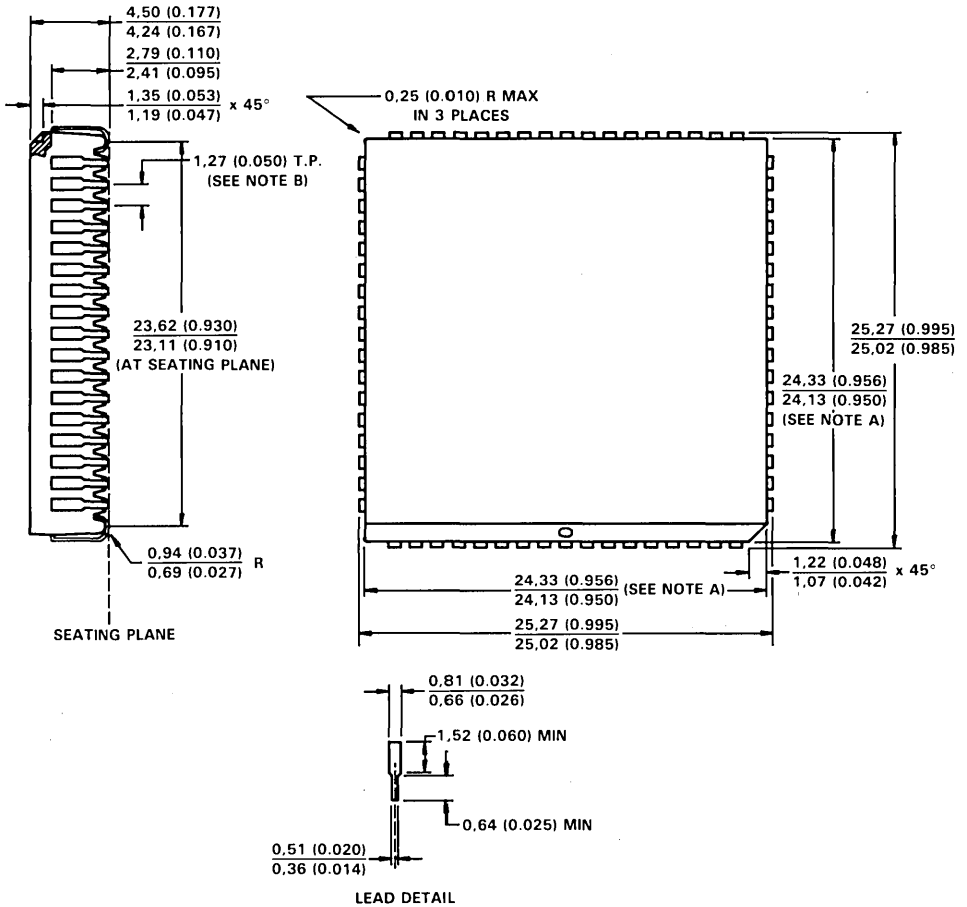
Figure 16-2. 28-pin Plastic Dual-Inline Package, 100-MIL Pin Spacing (Type N Package Suffix)



ALL LINEAR DIMENSIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES

NOTES: A. Center line of center pin each side is within 0,10 (0.004) of package centerline as determined from this dimension.
B. Location of each pin is within 0,127 (0.005) of time position with respect to center pin on each side.
C. The lead contact points are planar within 0,10 (0.004).

Figure 16-3. 28-Pin Plastic-Leaded Chip Carrier Package (Type FN Package Suffix)



NOTES: A. Centerline of center pin each side is within 0,10 (0.004) of package centerline as determined by this dimension.
B. Location of each pin is within 0,127 (0.005) of true position with respect to center pin on each side.

ALL LINEAR DIMENSIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES.

Figure 16-4. 68-Pin Plastic Leaded Chip Carrier Package (Type FN Package Suffix)

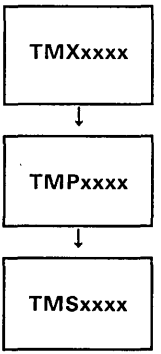
16.3 TMS370 Family Numbering and Symbol Conventions

All TMS370 devices are marked with information as to the type, package, copyright date(s), place of manufacture, and manufacturing data.

16.3.1 Device Prefix Designators

To provide expeditious system evaluations by customers during the product development cycle, Texas Instruments assigns a prefix designator with three options: TMX, TMP, and TMS.

TMX, TMP, and TMS are representative of the evolutionary stages of product development from engineering prototypes through fully qualified production devices. Figure 16-5 depicts this evolutionary development flowchart. Production devices shipped by Texas Instruments have the TMS designator signifying that they have demonstrated the high standards of Texas Instruments quality and reliability.



Experimental devices that may not be representative of the final device's electrical specifications and have not completed reliability verification.

Devices that conform to the electrical specifications but have not completed quality and reliability verification.

Fully qualified production devices.

Figure 16-5. Development Flowchart

TMX devices are shipped against the following disclaimer:

1) Experimental product and its reliability has not been characterized.
2) Product is sold "as is".
3) Product is not warranted to be exemplary of final production version if or when released by Texas Instruments.

TMP devices are shipped against the following disclaimer:

1) Customer understands that the product purchased hereunder has not been fully characterized and the expectation of reliability cannot be defined; therefore, Texas Instruments standard warranty refers only to the device's specifications.
2) No warranty of merchantability or fitness is expressed or implied.

TMS devices have been fully characterized and the quality and reliability of the device has been fully demonstrated. Texas Instruments' standard warranty applies.

16.3.2 Device Numbering Convention

Figure 16-6 illustrates the numbering and symbol nomenclature for the TMS370 family.

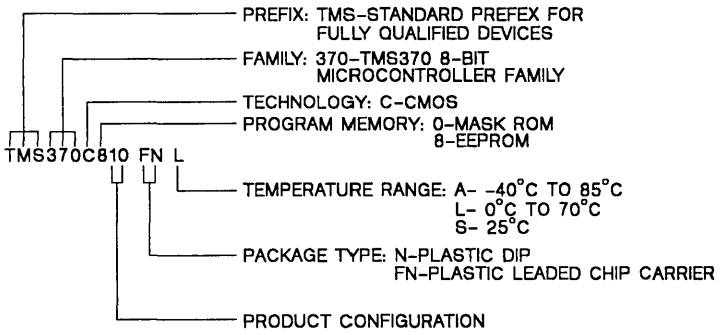


Figure 16-6. TMS370 Family Nomenclature

16.3.3 Device Symbols

The device symbolization of the TMS370 family members can be divided into two catagories: those with factory programmed mask ROM, and those with user programmed memory.

16.3.3.1 TMS370 Family Members with Mask-ROM

TMS370 family members with mask-ROM are custom-programmed devices where the ROM is mask programmed according to the customer's application code. These devices follow the prototyping and production flow outlined in Section 16.3. Since they are semi-custom devices, they receive a unique ROM code identification number.

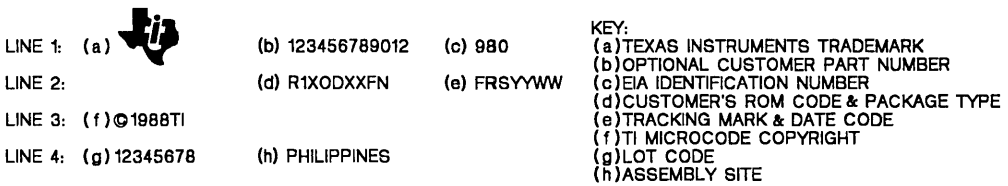


Figure 16-7. TI Standard Symbolization for Mask ROM Device in 28-Pin N-Type Package

16

LINE 1: (a) 123456789012
LINE 2: (b) R1XODXXFN
LINE 3: (c) 980 (d) FRSYYWW
LINE 4: (e) 12345678
LINE 5: (f) ©1986TI
(BACKSIDE) (g) PHILIPPINES

KEY:
(a)OPTIONAL CUSTOMER PART NUMBER
(b)CUSTOMER'S ROM CODE & PACKAGE TYPE
(c)EIA IDENTIFICATION NUMBER
(d)TRACKING MARK & DATE CODE
(e)LOT CODE
(f)TI MICROCODE COPYRIGHT
(g)ASSEMBLY SITE (BOTTOM OF PACKAGE)

Figure 16-8. TI Standard Symbolization for Mask ROM Device in 28-Pin FN Type Package

LINE 1: (a) 123456789012 (b)980
LINE 2: (c) R1XODXXFN
LINE 3: (d) (e)FRSYYWW
LINE 4: (f) 12345678
LINE 5: (g) ©1986TI
(BACKSIDE) (h)PHILIPPINES

KEY:
(a)OPTIONAL CUSTOMER PART NUMBER
(b)EIA IDENTIFICATION NUMBER
(c)CUSTOMER'S ROM CODE & PACKAGE TYPE
(d)TEXAS INSTRUMENTS TRADEMARK
(e)TRACKING MARK & DATE CODE
(f)LOT CODE
(g)TI MICROCODE COPYRIGHT
(h)ASSEMBLY SITE

Figure 16-9. TI Standard Symbolization for Mask ROM Device in 68-Pin FN Type Package

16.3.3.2 TMS370 Family Members with Program EEPROM

TMS370 family members with on-chip program EEPROM are standard device types, and therefore have a standard identification. The TMS370 family members with program EEPROM include the TMS370C810 and the TMS370C850.

LINE 1:
LINE 2: (a) (b) TMS370C810N
LINE 3: (d) ©1986TI (c) FRSYYWW
LINE 4: (f) PHILIPPINES (e) 12345678

KEY:
(a)TEXAS INSTRUMENTS TRADEMARK
(b)STANDARD DEVICE PART NUMBER
(c)TRACKING MARK & DATE CODE
(d)TI MICROCODE COPYRIGHT
(e)LOT CODE
(f)ASSEMBLY SITE

Figure 16-10. TI Standard Symbolization for Program EEPROM Device in N-Type Package

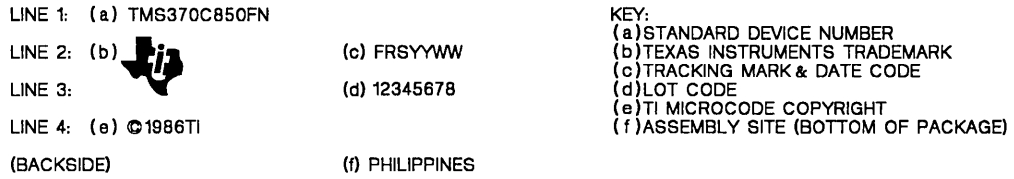


Figure 16-11. TI Standard Symbolization for EEPROM Device in FN-Type Package

16.4 Development Support Tools Ordering Information

All the necessary development support tools (excluding a PC) for the TMS370 family are available from TI separately or as a complete package. The develoopement tools are designed to work with an IBM, IBM compatible or TI PC with a minimum of 512K bytes of memory and a 5 1/4 inch floppy disk drive.

16.4.1 TMS370 Macro Assembler, Linker, and Utilities

This software package includes all the utilities required for developing object code for the TMS370 devices.

PART NUMBER	DESCRIPTION
TMDS3740810-02	Assembler/Linker

16.4.2 TMS370 EEPROM Programmer

The TMS370 EEPROM Programmer provides the physical means to program the TMS370 prototype devices. The programmer comes with the necessary cables and control software for interfacing with an IBM compatible or TI PC.

PART NUMBER	DESCRIPTION
TMDS3760510	EEPROM Programmer

16.4.3 TMS370 XDS System

The XDS System provides software debugging and overall evaluation of a TMS370-based system. The XDS comes complete with necessary cables and debugging program.

PART NUMBER	DESCRIPTION
TMDS3762210	XDS System

16.4.4 Complete TMS370 Development System

The components above (Assembler/Linker, EEPROM Programmer, and XDS System) are available as a single package providing full support of the TMS370 family devices.

PART NUMBER	DESCRIPTION
TMDS3792210	TMS370 Development

A. Peripheral File Map

This appendix summarizes the Peripheral File (PF) and control bit information into a single location for reference.

Each PF register is presented as a row of boxes containing the control or status bits belonging to the register. The register symbol (e.g., SCCR0) and the PF hex address (i.e., P010) are to the left of each register.

The read/write accessibility of each bit is indicated in parentheses below each bit symbol, with the following definitions:

- R – read
- W – write
- P – write in the privilege mode only
- C – clear only
- S – set only
- -0 – cleared by RESET
- -1 – set by RESET
- -† – this bit exhibits special behavior during or after RESET; see the description for this bit in the appropriate section (both bit and register are index entries).

The register summary is followed by block diagrams of the major circuits. The control bits are shown in these diagrams in the following format:

(xx.n) 4A.0 Bit location convention used in figures, where 'xx' is the hexadecimal address of the peripheral register containing the bit and 'n' is the bit number (7 = msb, 0 = lsb).

Bit # -	7	6	5	4	3	2	1	0
SCCR0 P010	COLD START (RC-†)	OSC POWER (RP-0)	PF AUTO WAIT (RW-0)	OSC FLT FLAG (RW-†)	MODE PIN WPO (R-†)	MC PIN DATA (R-†)	---	µP/µC Mode (R)
SCCR1 P011	---	---	---	AUTOWAIT DISABLE (RP-0)	---	MEMORY DISABLE (RP-†)	---	---
SCCR2 P012	HALT/ STANDBY (RP-0)	PWR-DWN/ IDLE (RP-0)	OSC FLT RST ENA (RP-0)	BUS STEST (RP-0)	CPU STEST (RP-1)	OSC FLT DISABLE (RP-0)	INT1 NMI (RP-0)	PRIV-ILEGE DISABLE (RS-0)
INT1 P017	INT1 FLAG (RC-0)	INT1 Pin DATA (R-0)	---	---	---	INT1 POLARITY (RW-0)	INT1 PRIORITY (RW-0)	INT1 ENABLE (RW-0)
INT2 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 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 P01A	BUSY (R-†)	---	---	---	---	AP (RW-0)	W1W0 (RW-0)	EXE (RW-0)
PEECTL P01C	BUSY (R-†)	---	---	---	---	AP (RW-0)	W1W0 (RW-0)	EXE (RW-0)
APORT2 P021	PORT A CONTROL REGISTER 2							
ADATA P022	PORT A DATA							
ADIR P023	PORT A DIRECTION							
BPORT2 P025	PORT B CONTROL REGISTER 2							
BDATA P026	PORT B DATA							
BDIR P027	PORT B DIRECTION							
CPORT2 P029	PORT C CONTROL REGISTER 2							
CDATA P02A	PORT C DATA							
CDIR P02B	PORT C DIRECTION							
DPORT1 P02C	PORT D CONTROL REGISTER 1							
DPORT2 P02D	PORT D CONTROL REGISTER 2							
DDATA P02E	PORT D DATA							
DDIR P02F	PORT D DIRECTION							

SPICCR 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 P031	RECEIVER OVERRUN (R-0)	SPI INT FLAG (R-0)	---	---	---	MASTER/ SLAVE (RW-0)	TALK (RW-0)	SPI INT ENA (RW-0)
SPIBUF 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)
SPIDAT 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)
SPIPC1 P03D	---	---	---	---	SPICLK DATA IN (R-0)	SPICLK DATA OUT (RW-0)	SPICLK FUNCTION (RW-0)	SPICLK DATA DIR (RW-0)
SPIPC2 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 P03F	SPI STEST (RP-0)	SPI PRIORITY (RP-0)	SPI ESPEN (RP-0)	---	---	---	---	---

T1CNTR MSB P040	Bit 15	T1 COUNTER MSB						Bit 8
T1CNTR LSB P041	Bit 7	T1 COUNTER LSB						Bit 0
T1C MSB P042	Bit 15	COMPARE REGISTER MSB						Bit 8
T1C LSB P043	Bit 7	COMPARE REGISTER LSB						Bit 0
T1CC MSB P044	Bit 15	CAPTURE/COMPARE REGISTER MSB						Bit 8
T1CC LSB P045	Bit 7	CAPTURE/COMPARE REGISTER LSB						Bit 0
WDCNTR MSB P046	Bit 15	WATCHDOG COUNTER MSB						Bit 8
WDCNTR LSB P047	Bit 7	WATCHDOG COUNTER LSB						Bit 0
WDRST P048	Bit 7	WATCHDOG RESET KEY						Bit 0
T1CTL1 P049	WD OVRFL TAP SEL (RP-0)	WD INPUT SELECT 2 (RP-0)	WD INPUT SELECT 1 (RP-0)	WD INPUT SELECT 0 (RP-0)	---	T1 INPUT SELECT 2 (RW-0)	T1 INPUT SELECT 1 (RW-0)	T1 INPUT SELECT 0 (RW-0)
T1CTL2 P04A	WD OVRFL RST ENA (RS-0)	WD OVERFL INT ENA (RW-0)	WD OVERFL INT FLAG (RC-†)	T1 OVRFL INT ENA (RW-0)	T1 OVRFL INT FLAG (RC-0)	---	---	T1 SW RESET (S-0)
T1CTL3 P04B	*Dual Compare Mode*							
	T1EDGE INT FLAG (RC-0)	T1C2 INT FLAG (RC-0)	T1C1 INT FLAG (RC-0)	---	---	T1EDGE INT ENA (RW-0)	T1C2 INT ENA (RW-0)	T1C1 INT ENA (RW-0)
	Capture/Compare Mode							
	T1EDGE INT FLAG (RC-0)	---	T1C1 INT FLAG (RC-0)	---	---	T1EDGE INT ENA (RW-0)	---	T1C1 INT ENA (RW-0)
T1CTL4 P04C	*Dual Compare Mode*							
	T1 MODE =0 (RW-0)	T1C1 OUT ENA (RW-0)	T1C2 OUT ENA (RW-0)	T1C1 RST ENA (RW-0)	T1CR OUT ENA (RW-0)	T1 EDGE 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)	---	T1 EDGE POLARITY (RW-0)	---	T1EDGE DET ENA (RW-0)

Bit # -	7	6	5	4	3	2	1	0
T1PC1 P04D	---	---	---	---	T1EVT DATA IN (R-0)	T1EVT DATA OUT (RW-0)	T1EVT FUNCTION (RW-0)	T1EVT DATA DIR (RW-0)
T1PC2 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 P04F	T1 STEST (RP-0)	T1 PRIORITY (RP-0)	---	---	---	---	---	---
SCICCR P050	STOP BITS (RW-0)	EVEN/ ODD PARITY (RW-0)	PARITY ENABLE (RW-0)	ASYNC/ ISOSYNC (RW-0)	ADDRESS IDLE WUP (RW-0)	SCI CHAR2 (RW-0)	SCI CHAR1 (RW-0)	SCI CHAR0 (RW-0)
SCICTL P051	---	---	SCI SW RESET (RW-0)	CLOCK (RW-0)	TXWAKE (RS-0)	SLEEP (RW-0)	TXENA (RW-0)	RXENA (RW-0)
BAUD MSB P052	BAUDF (msb)	BAUDE	BAUDD	BAUDC	BAUDB	BAUDA	BAUD9	BAUD8
BAUD LSB P053	BAUD7 (RW-0)	BAUD6 (RW-0)	BAUD5 (RW-0)	BAUD4 (RW-0)	BAUD3 (RW-0)	BAUD2 (RW-0)	BAUD1 (RW-0)	BAUD0 (lsb) (RW-0)
TXCTL P054	TXRDY (R-1)	TX EMPTY (R-1)	---	---	---	---	---	SCI TX INT ENA (RW-0)
RXCTL P055	RX ERROR (R-0)	RXRDY (R-0)	BRKDT (R-0)	FE (R-0)	OE (R-0)	PE (R-0)	RXWAKE (R-0)	SCI RX INT ENA (RW-0)
RXBUF P057	RXDT7 (R-0)	RXDT6 (R-0)	RXDT5 (R-0)	RXDT4 (R-0)	RXDT3 (R-0)	RXDT2 (R-0)	RXDT1 (R-0)	RXDT0 (R-0)
TXBUF P059	TXDT7 (RW-0)	TXDT6 (RW-0)	TXDT5 (RW-0)	TXDT4 (RW-0)	TXDT3 (RW-0)	TXDT2 (RW-0)	TXDT1 (RW-0)	TXDT0 (RW-0)
SCIPC1 P05D	---	---	---	---	SCICLK DATA IN (R-0)	SCICLK DATA OUT (RW-0)	SCICLK FUNCTION (RW-0)	SCICLK DATA DIR (RW-0)
SCIPC2 P05E	SCITXD DATA IN (R-0)	SCITXD DATA OUT (RW-0)	SCITXD FUNCTION (RW-0)	SCITXD DATA DIR (RW-0)	SCIRXD DATA IN (R-0)	SCIRXD DATA OUT (RW-0)	SCIRXD FUNCTION (RW-0)	SCIRXD DATA DIR (RW-0)
SCIPRI P05F	SCI STEST (RP-0)	SCITX PRIORITY (RP-0)	SCIRX PRIORITY (RP-0)	SCI ESPEN (RP-0)	---	---	---	---

T2CNTR MSB P060	Bit 15	T2 COUNTER MSB	Bit 8
T2CNTR LSB P061	Bit 7	T2 COUNTER LSB	Bit 0
T2C MSB P062	Bit 15	COMPARE REGISTER MSB	Bit 8
T2C LSB P063	Bit 7	COMPARE REGISTER LSB	Bit 0
T2CC MSB P064	Bit 15	CAPTURE/COMPARE REGISTER MSB	Bit 8
T2CC LSB P065	Bit 7	CAPTURE/COMPARE REGISTER LSB	Bit 0
T2IC MSB P066	Bit 15	T2 CAPTURE REGISTER MSB	Bit 8
T2IC LSB P067	Bit 7	T2 CAPTURE REGISTER LSB	Bit 0

Bit # -	7	6	5	4	3	2	1	0
T2CTL1 P06A	---	---	---	T2 OVRFL INT ENA (RW-0)	T2 OVRFL INT FLAG (RC-0)	T2 INPUT SELECT 1 (RW-0)	T2 INPUT SELECT 0 (RW-0)	T2 SW RESET (S-0)
	Dual Compare Mode							
T2CTL2 P06B	T2EDGE1 INT FLG (RC-0)	T2C2 INT FLG (RC-0)	T2C1 INT FLG (RC-0)	---	---	T2EDGE1 INT ENA (RW-0)	T2C2 INT ENA (RW-0)	T2C1 INT ENA (RW-0)
	Dual Capture Mode							
	T2EDGE1 INT FLG (RC-0)	T2EDGE2 INT FLG (RC-0)	T2C1 INT FLG (RC-0)	---	---	T2EDGE1 INT ENA (RW-0)	T2EDGE2 INT ENA (RW-0)	T2C1 INT ENA (RW-0)
	Dual Compare Mode							
T2CTL3 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 P06D	---	---	---	---	T2EVT DATA IN (R-0)	T2EVT DATA OUT (RW-0)	T2EVT FUNCTION (RW-0)	T2EVT DATA DIR (RW-0)
T2PC2 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 P06F	T2 STEST (RP-0)	T2 PRIORITY (RP-0)	---	---	---	---	---	---
ADCTL P070	CONVERT START (RW-0)	SAMPLE START (RW-0)	REF VOLT SELECT2 (RW-0)	REF VOLT SELECT1 (RW-0)	REF VOLT SELECT0 (RW-0)	AD INPUT SELECT2 (RW-0)	AD INPUT SELECT1 (RW-0)	AD INPUT SELECT0 (RW-0)
ADSTAT P071	---	---	---	---	---	AD READY (R-1)	AD INT FLAG (RC-0)	AD INT ENA (RW-0)
ADDATA P072	DATA7 (R-0)	DATA6 (R-0)	DATA5 (R-0)	DATA4 (R-0)	DATA3 (R-0)	DATA2 (R-0)	DATA1 (R-0)	DATA0 (R-0)
ADIN P07D	PORT E DATA AN 7 (R-0)	PORT E DATA AN 6 (R-0)	PORT E DATA AN 5 (R-0)	PORT E DATA AN 4 (R-0)	PORT E DATA AN 3 (R-0)	PORT E DATA AN 2 (R-0)	PORT E DATA AN 1 (R-0)	PORT E DATA AN 0 (R-0)
ADENA P07E	PORT E INPUT ENA 7 (RW-0)	PORT E INPUT ENA 6 (RW-0)	PORT E INPUT ENA 5 (RW-0)	PORT E INPUT ENA 4 (RW-0)	PORT E INPUT ENA 3 (RW-0)	PORT E INPUT ENA 2 (RW-0)	PORT E INPUT ENA 1 (RW-0)	PORT E INPUT ENA 0 (RW-0)
ADPRI P07F	AD STEST (RP-0)	AD PRIORITY (RP-0)	AD ESPEN (RP-0)	---	---	---	---	---

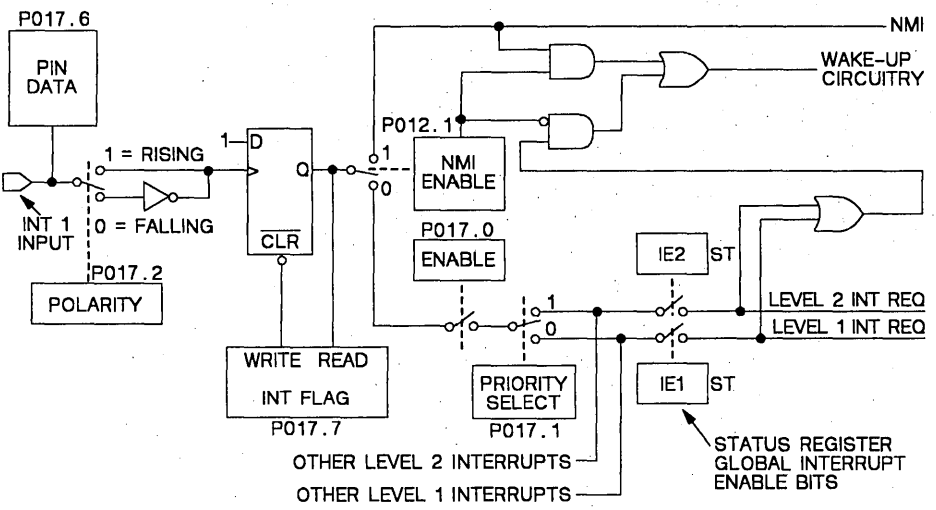


Figure A-1. Interrupt 1 Block Diagram

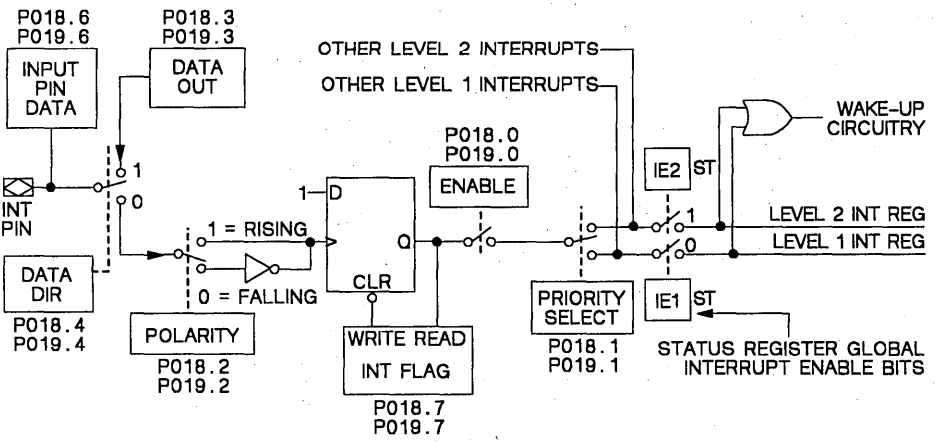


Figure A-2. Interrupts 2 and 3 Block Diagram

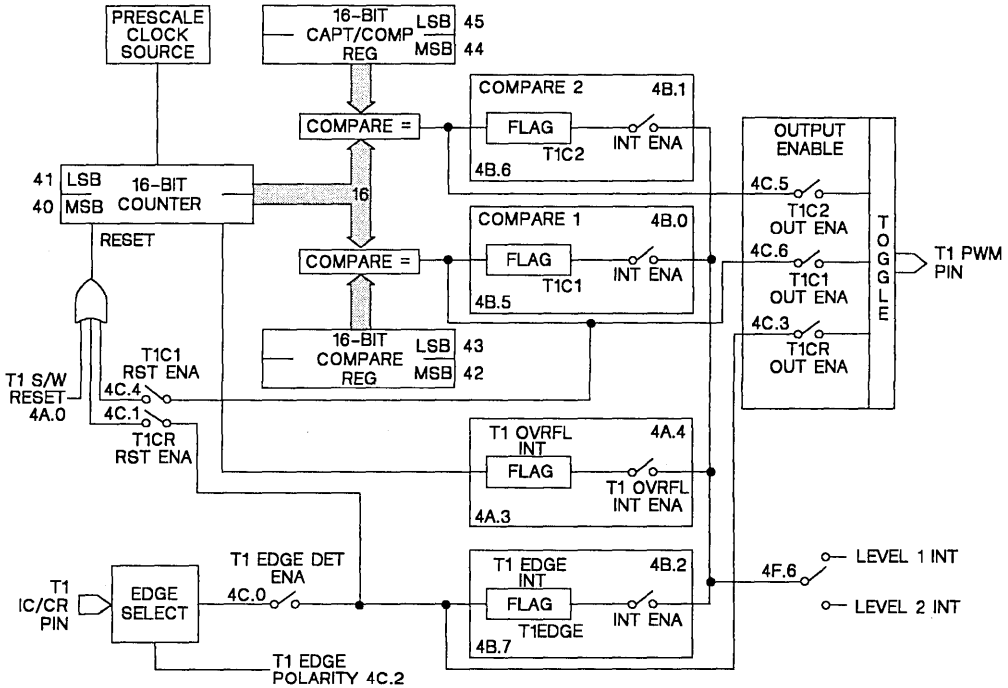


Figure A-3. Timer 1: Dual Compare Mode

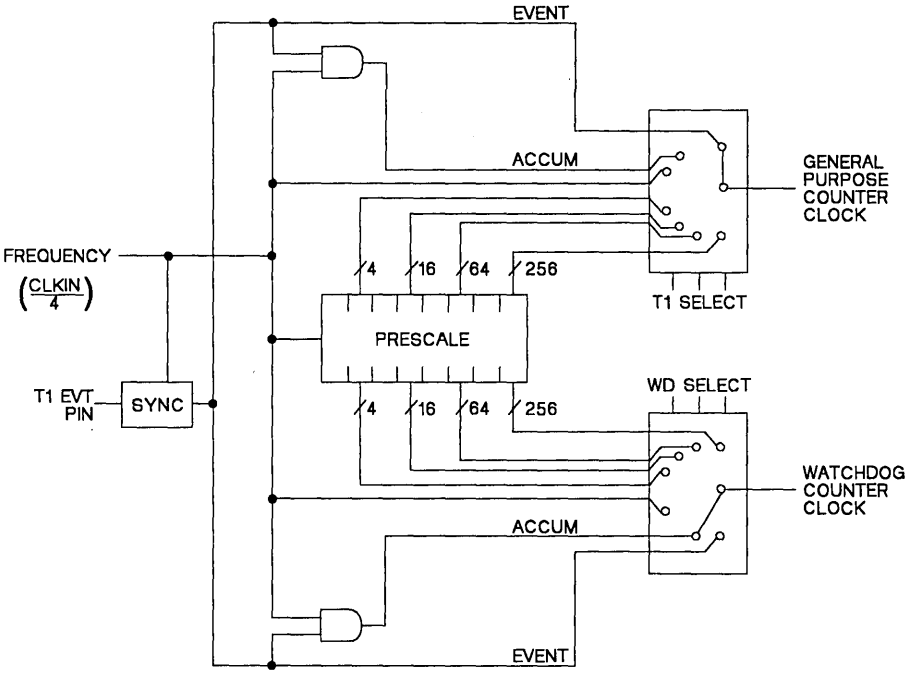


Figure A-4. Timer 1 System Clock Prescaler

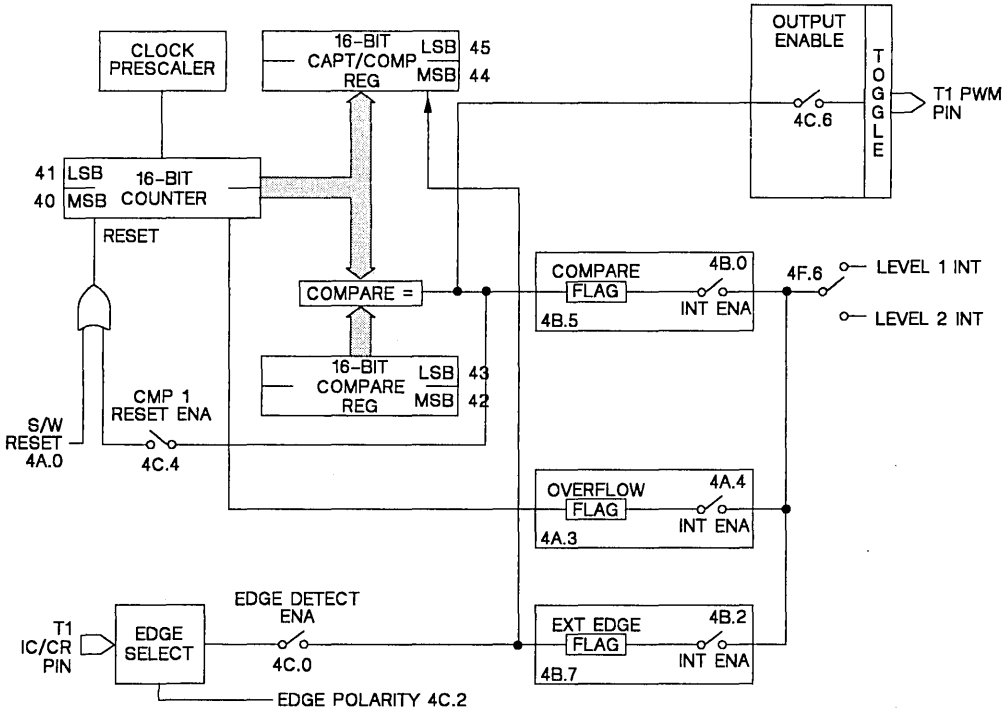


Figure A-5. Timer 1: Capture/Compare Mode

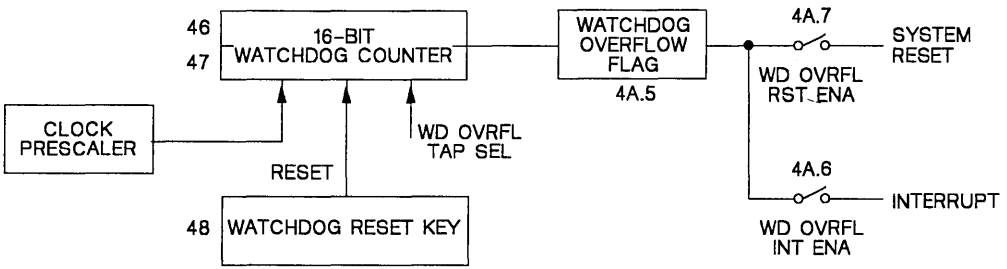


Figure A-6. Watchdog Timer

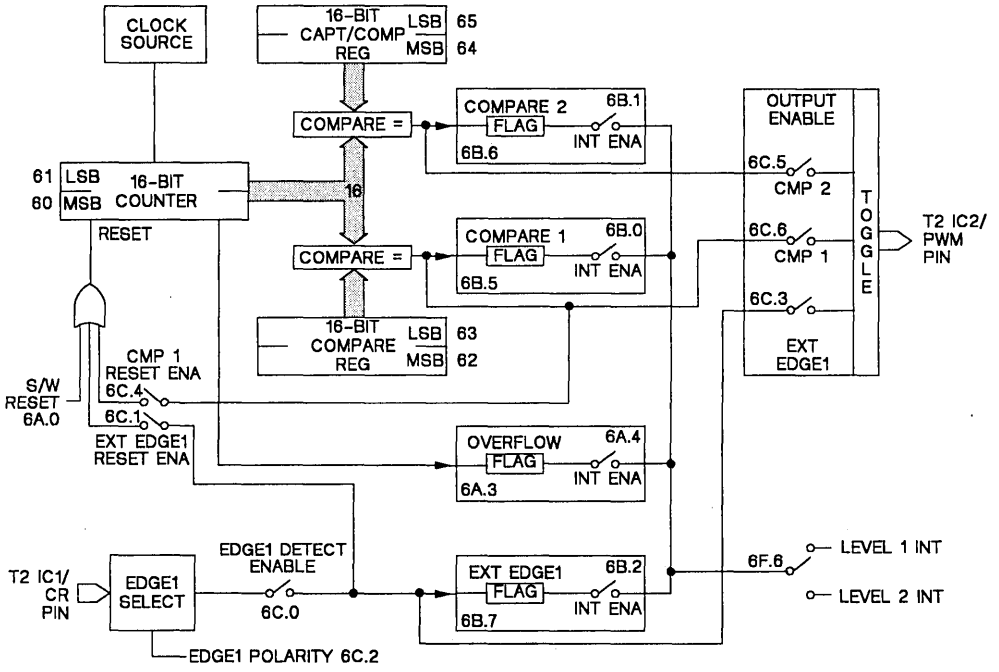


Figure A-7. Timer 2: Dual Compare Mode

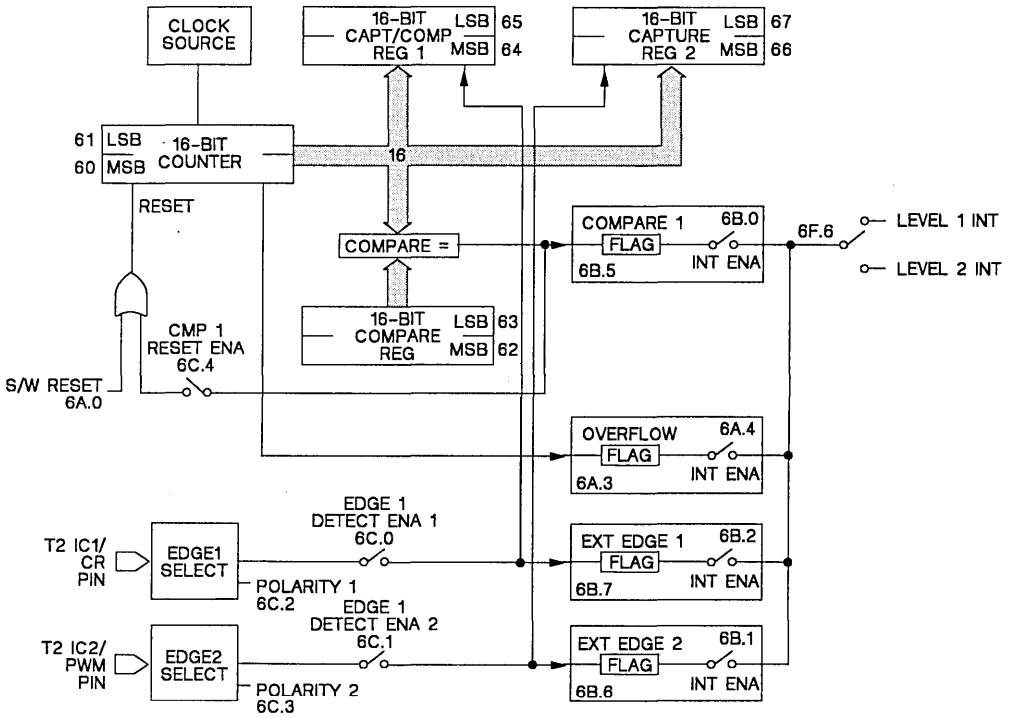


Figure A-8. Timer 2: Dual Capture Mode

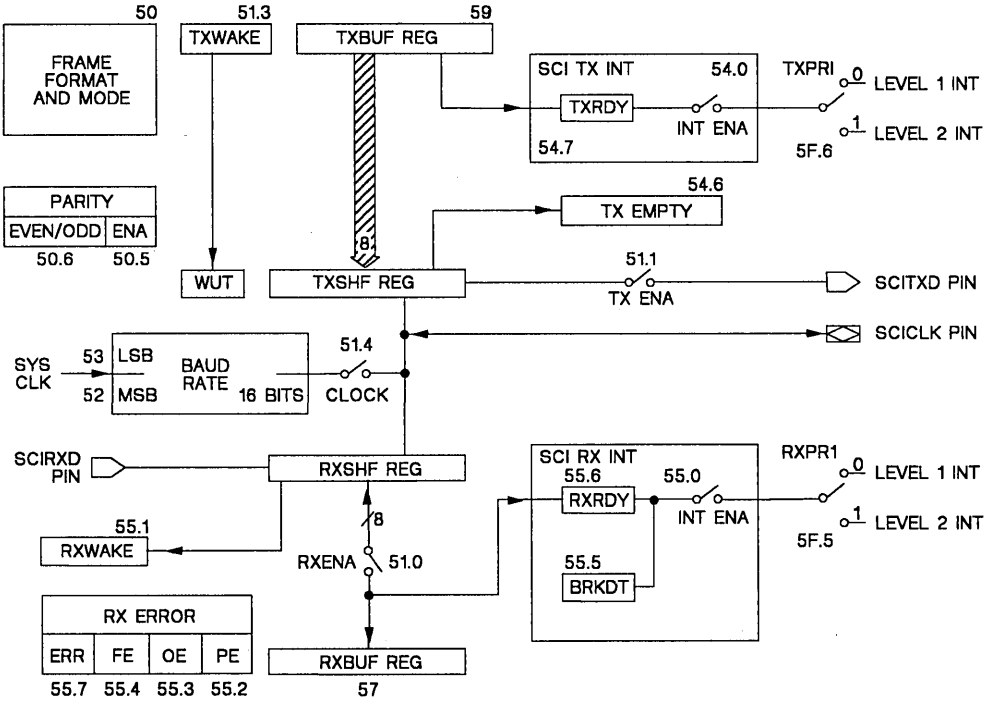


Figure A-9. SCI Block Diagram

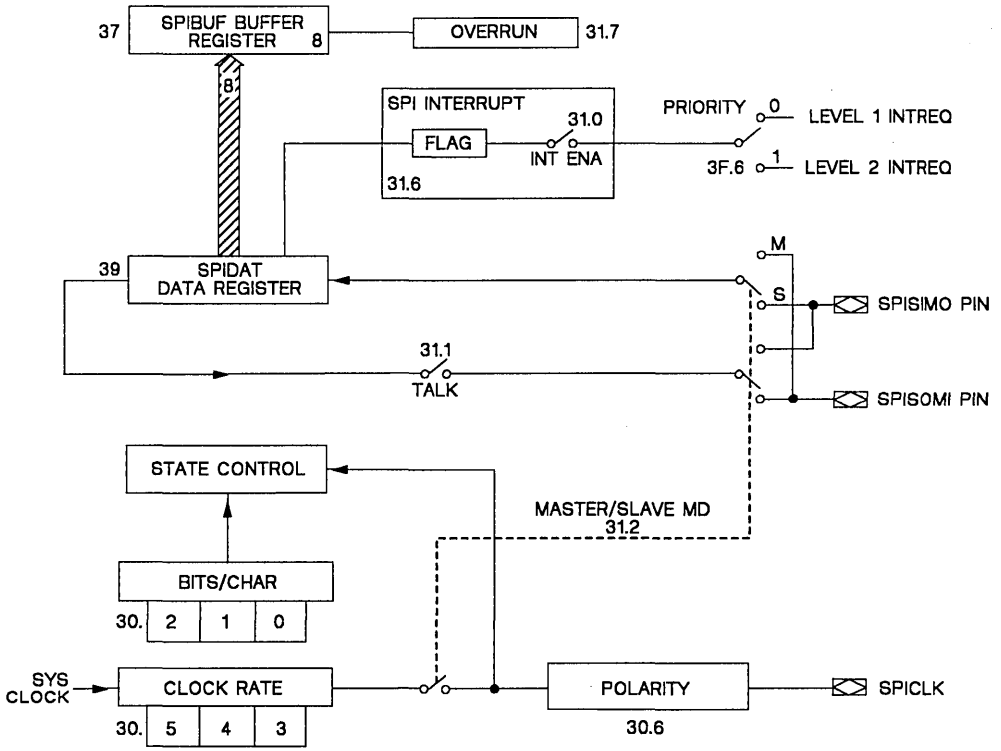


Figure A-10. SPI Block Diagram

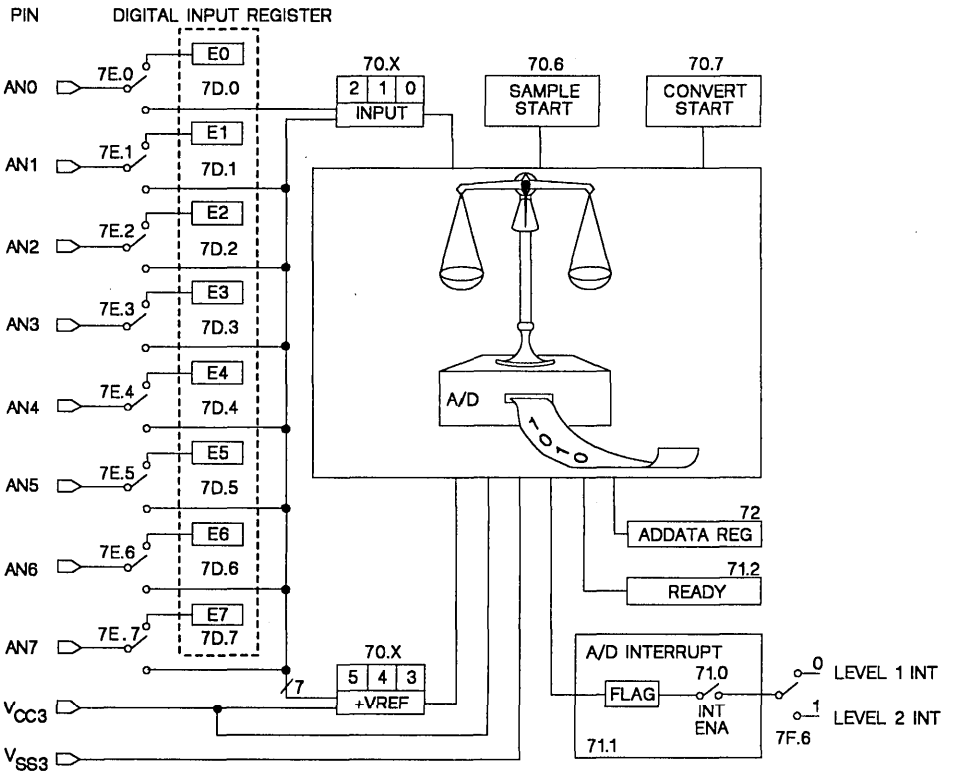


Figure A-11. Analog-to-Digital Converter Block Diagram

B. Character Sets

The TMS370 Assembler recognizes the ASCII character set listed in Table B-1. Table B-2 lists characters that the assembler does not recognize, but may be recognized and acted upon by other programs. The device service routine for the card reader accepts and stores into the calling program's buffer all the characters listed.

Table B-1. ASCII Character Set

HEX (Low nibble)	0-	1-	2-	3-	4-	5-	6-	7- (High nibble)
-0	NUL 0	DLE 16	SP 32	0 48	@ 64	P 80	' 96	p 112
-1	SOH 1	DC1 17	! 33	1 49	A 65	Q 81	a 97	q 113
-2	STX 2	DC2 18	" 34	2 50	B 66	R 82	b 98	r 114
-3	ETX 3	DC3 19	- 35	3 51	C 67	S 83	c 99	s 115
-4	EOT 4	DC4 20	$ 36	4 52	D 68	T 84	d 100	t 116
-5	ENQ 5	NAK 21	% 37	5 53	E 69	U 85	e 101	u 117
-6	ACK 6	SYN 22	& 38	6 54	F 70	V 86	f 102	v 118
-7	BEL 7	ETB 23	' 39	7 55	G 71	W 87	g 103	w 119
-8	BS 8	CAN 24	(40	8 56	H 72	X 88	h 104	x 120
-9	HT 9	EM 25	) 41	9 57	I 73	Y 89	i 105	y 121
-A	LF A	SUB 26	* 42	: 58	J 74	Z 90	j 106	z 122
-B	VT B	ESC 27	+ 43	; 59	K 75	[91	k 107	{ 123
-C	FF C	FS 28	' 44	< 60	L 76	\ 92	l 108	| 124
-D	CR D	GS 29	- 45	= 61	M 77	] 93	m 109	} 125
-E	SO E	RS 30	. 46	> 62	N 78	^ 94	n 110	~ 126
-F	SI F	US 31	/ 47	? 63	O 79	- 95	o 111	DEL 127

Table B-2. Control Characters

HEX VALUE	DECIMAL VALUE	CHARACTER
00	0	NUL
01	1	SOH
02	2	STX
03	3	ETX
04	4	EOT
05	5	ENQ
06	6	ACK
07	7	BEL
08	8	BS
09	9	HT
0A	10	LF
0B	11	VT
0C	12	FF
0D	13	CR
0E	14	SO
0F	15	SI
10	16	DLE
11	17	DC1
12	18	DC2
13	19	DC3
14	20	DC4
15	21	NAK
16	22	SYN
17	23	ETB
18	24	CAN
19	25	EM
1A	26	SUB
1B	27	ESC
1C	28	FS
1D	29	GS
1E	30	RS
1F	31	US
7F	127	DEL

C. Opcode/Instruction Cross Reference

Table C-1 (on the following pages) provides an opcode-to-instruction cross reference of all 73 mnemonics and 245 opcodes of the TMS370 instruction set. To check the instruction of a known opcode, locate the left (high) digit across the top or bottom of the table, then find the right (low) digit along the side of the table. The intersection contains the instruction mnemonic, operands, and byte/cycle particular to that opcode. Some opcodes, such as B0, are shared by two instructions, in which case both mnemonics are shown along with the byte/cycles count.

Table C-1. TMS370 Family Opcode/Instruction Map

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	JMP ra 2/7							INCW #n,Rd 3/11	MOV Ps,A 2/8			CLRC TST A 1/9	MOV A,B 1/9	MOV A,Rd 2/7	TRAP 15 1/14	LDST n 2/6
1	JN ra 2/5		MOV A,Pd 2/8			MOV B,Pd 2/8		MOV Rs,Pd 3/10		MOV Ps,B 2/7				MOV B,Rd 2/7	TRAP 14 1/14	MOV n(SP),A 2/7
2	JZ ra 2/5	MOV Rs,A 2/7	MOV #n,A 2/6	MOV Rs,B 2/7	MOV Rs,Rd 3/9	MOV #n,B 2/6	MOV B,A 1/8	MOV #n,Rd 3/8			MOV Ps,Rd 3/10	DEC A 1/8	DEC B 1/8	DEC Rn 2/6	TRAP 13 1/14	MOV A,n(SP) 2/7
3	JC ra 2/5	AND Rs,A 2/7	AND #n,A 2/6	AND Rs,B 2/7	AND Rs,Rd 3/9	AND #n,B 2/6	AND B,A 1/8	AND #n,Rd 3/8	AND A,Pd 2/9	AND B,Pd 2/9	AND #n,Pd 3/10	INC A 1/8	INC B 1/8	INC Rn 2/6	TRAP 12 1/14	CMP n(SP),A 2/8
4	JP ra 2/5	OR Rs,A 2/7	OR #n,A 2/6	OR Rs,B 2/7	OR Rs,Rd 3/9	OR #n,B 2/6	OR B,A 1/8	OR #n,Rd 3/8	OR A,Pd 2/9	OR B,Pd 2/9	OR #n,Pd 3/10	INV A 1/8	INV B 1/8	INV Rn 2/6	TRAP 11 1/14	extend inst,2 opcodes
5	JPZ ra 2/5	XOR Rs,A 2/7	XOR #n,A 2/6	XOR Rs,B 2/7	XOR Rs,Rd 3/9	XOR #n,B 2/6	XOR B,A 1/8	XOR #n,Rd 3/8	XOR A,Pd 2/9	XOR B,Pd 2/9	XOR #n,Pd 3/10	CLR A 1/8	CLR B 1/8	CLR Rn 2/6	TRAP 10 1/14	
6	JNZ ra 2/5	BTJO Rs,A 3/9	BTJO #n,A 3/8	BTJO B,Rd 3/9	BTJO Rs,Rd 4/11	BTJO #n,B 3/8	BTJO B,A 2/10	BTJO #n,Rd 4/10	BTJO A,Pd 3/11	BTJO B,Pd 3/10	BTJO #n,Pd 4/11	XCHB A 1/10	XCHB TESTB 1/10	XCHB Rn 2/8	TRAP 9 1/14	IDLE 1/6
7	JNC ra 2/5	BTJZ Rs,A 3/9	BTJZ #n,A 3/8	BTJZ Rs,B 3/9	BTJZ Rs,Rd 4/11	BTJZ #n,B 3/8	BTJZ B,A 2/10	BTJZ #n,Rd 4/10	BTJZ A,Pd 3/10	BTJZ B,Pd 3/10	BTJZ #n,Pd 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,Rd 4/13	MOVW Rs,Rd 3/12	MOVW #16(B),Rd 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 @Rd 2/8	JMPL lab(B) 3/10	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 @Rs,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,@Rd 2/9	MOV A,lab(B) 3/12	COMPL A 1/8	COMPL B 1/8	COMPL Rn 2/10	TRAP 12 1/14	PUSH ST 1/8
C	JNV ra 2/5	MPY Rs,A 2/46	MPY #n,A 2/45	MPY Rs,B 2/46	MPY Rs,Rd 3/48	MPY #n,B 2/45	MPY B,A 1/47	MPY #n,Rs 3/47	BR lab 3/9	BR @Rd 2/8	BR lab(B) 3/11	RR A 1/8	RR B 1/8	RR Rn 2/6	TRAP 3 1/14	POP ST 1/8
D	JGE ra 2/5	CMP Rs,A 2/7	CMP #n,A 2/6	CMP Rs,B 2/7	CMP Rs,Rd 3/9	CMP #n,B 2/6	CMP B,A 1/8	CMP #n,Rd 3/8	CMP lab,A 3/11	CMP @Rs,A 2/10	CMP lab(B),A 3/13	RRC A 1/8	RRC B 1/8	RRC Rn 2/6	TRAP 2 1/14	LDSP 1/7
E	JG ra 2/5	DAC Rs,A 2/9	DAC #n,A 2/8	DAC Rs,B 2/9	DAC Rs,Rd 3/11	DAC #n,B 2/8	DAC B,A 1/10	DAC #n,Rd 3/10	CALL lab 3/13	CALL @Rd 2/12	CALL lab(B) 3/15	RL A 1/8	RL B 1/8	RL Rn 2/6	TRAP 1 1/14	STSP 1/8
F	JLO ra 2/5	DSB Rs,A 2/9	DSB #n,A 2/8	DSB Rs,B 2/9	DSB Rs,Rd 3/11	DSB #n,B 2/8	DSB B,A 1/10	DSB #n,Rd 3/10	CALLR lab 3/15	CALLR @Rd 2/14	CALLR lab(B) 3/17	RLC A 1/8	RLC B 1/8	RLC Rn 2/6	TRAP 0 1/14	NOP 1/7
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

NOTE ALL CONDITIONAL JUMPS (OPCODES 01-0F), BTJO, AND BTJZ INSTRUCTIONS USE TWO ADDITIONAL CYCLES IF THE BRANCH IS TAKEN. THE BTJO AND BTJZ INSTRUCTIONS HAVE A RELATIVE ADDRESS AS THE LAST OPERAND.

Second byte of two-byte instructions (F4xx):

	E	F
8	MOVW n(Rn) 4/15	DIV Rn,A 3/14-63
9	JMPL n(Rn) 4/16	
A	MOV n(Rn),A 4/17	
B	MOV A,n(Rn) 4/16	
C	BR n(Rn) 4/16	
D	CMP n(Rn) 4/18	
E	CALL n(Rn) 4/20	
F	CALLER n(Rn) 4/22	
	E	F

ra - relative address
Rn - Register
Rs - Register containing source byte
Rd - Register containing destination byte
Ps - Peripheral register containing source byte
Pd - Peripheral register containing destination byte
Pn - Peripheral register
n - Immediate 8-bit number
#16 - Immediate 16-bit number
lab - 16-bit label

D. Instruction/Opcode Cross Reference

Table D-1 provides an instruction-to-opcode cross reference of all 73 mnemonics and 245 opcodes of the TMS370 instrruction set. The columns are grouped according to addressing modes (General and Extended). The "Other" column contains either the opcode(s) of instructions that do not qualify for the General or Extended catagories, or a notation to be referenced at the bottom of the table for more information on a particular instruction.

Table D-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
JL																							09
JLE																							0A
JLO																							0F
JN																							01
JNC																							07
JNE/JNZ																							06
JNV																							0C
JP																							04
JPZ																							05
JV																							08
LDSP																							FD

† Direct {(label) → (A)}
‡ Indexed {(label + (B)) → (A)}
§ Indirect {(Rn-1: Rn) → (A)}
¶ Offset Indirect (dual opcode instruction, the first of which is F4) {(n + (Rn - 1: Rn)) → (A)}
» Single opcode instructions that do not qualify as a General or Extended addressing mode, and dual opcode instructions that do not qualify as an Offset Indirect addressing mode.

Table D-1. TMS370 Family Instruction/Opcode Set (Concluded)

	GENERAL																	EXTENDED				Other
	A	B	Rn	A,B	B,A	Rn, A	#n, A	Rn, B	#n, B	Rn, Rn	#n, Rn	A, Rn	B, Rn	A, Pn	Pn, A	B, Pn	Pn, B	#n, Pn	†	‡	§	¶	»
LDST																							F0
MOV				C0	62	12	22	32	52	42	72	D0	D1	21	80	51	91	F7	8B	AA	9A	EA	‖
MOVW																			88	A8	98	E8	
MPY					6C	1C	2C	3C	5C	4C	7C												
NOP																							FF
OR					64	14	24	34	54	44	74			84		94		A4					
POP	B9	C9	D9																				FC
PUSH	B8	C8	D8																				FB
RL	BE	CE	DE																				
RLC	BF	CF	DF																				
RR	BC	CC	DC																				
RRC	BD	CD	DD																				
RTI																							FA
RTS																							F9
SBB					6B	1B	2B	3B	5B	4B	7B												
SBIT0																							∫
SBIT1																							∫'
SETC																							F8
STSP																							FE
SUB					6A	1A	2A	3A	5A	4A	7A												
SWAP	B7	C7	D7																				
TRAP																							**
TST	B0	C6																					
XCHB	B6	C6	D6																				
XOR					65	15	25	35	55	45	75			85		95		A5					

† Direct

‡ Indexed

§ Indirect

¶ Offset Indirect (dual opcode instruction, the first of which is F4)

» Unless otherwise indicated, includes single opcode instructions that do not qualify as a General or Extended addressing mode, and dual opcode instructions that do not qualify as an Offset Indirect addressing mode.

‖ The MOV instruction also includes the following options and their opcodes: Rn,Pn {71}; Pn,Rn {A2}; A,label(B) {AB}; A,n(SP) {F2}; A,n(Rn) {F4 EB}; label,A {8A}; n(SP),A {F1}

∫ The SBIT0 instruction consists of the following options and their opcodes; Rname {73}; Pname {A3}

∫' The SBIT1 instruction consists of the following options and their opcodes; Rname {74}; Pname {A4}

** The TRAP instruction consists of 15 options using operands 0 through 15 with opcodes EF through E0 respectively.

E. Glossary

This appendix provides definitions of terms and concepts unique to the TMS370 family of devices. Other common terms are included if the use of those terms varies from generally accepted usage.

absolute address: An addressing mode in which code or operands produce the actual address.

A/D pins: The 10 pins that connect the A/D module to the external world; includes AN0-7, V_{SS3}, and V_{CC3}.

addressing mode: The method by which an instruction calculates the location of its required data.

AN0-AN7 pins: Eight analog input channels to the A/D converter or digital inputs; seven of which can be configured as the Voltage reference channel.

analog-to-digital (A/D) converter: The TMS370 A/D Converter is an 8-bit successive-approximation converter with internal sample-and-hold circuitry.

assembly language: A symbolic language that describes the binary machine code in a more readable form. Each of the 73 unique instructions of the TMS370 family converts to one machine operation.

Asynchronous communications mode: An serial communications format that needs no synchronizing clock. This format consists of a start bit followed by data bits, an optional parity bit and ends with a stop bit. This format is commonly used with RS-232-C communications and PC serial ports.

BCD: Binary coded decimal; each 4 bit nibble expresses a digit from 0-9, and usually packed two digits to a byte giving a range of 0-99.

baud rates: The communication speed for serial ports; equivalent to bits per second.

Capture register: A Timer 1 or Timer 2 register which 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.

Compare register: The compare register, in the Timer 1 or Timer 2 module, contains a value which is compared to the counter value. The compare function triggers when the counter matches the contents of the compare register.

constant: A value which does not change during execution.

CPU: The TMS370 CPU is an 8-bit register oriented processor with Status register, Program Counter register, and Stack Pointer. The CPU uses the Register File, accessed in one bus cycle, as working registers.

edge detection: Edge detection circuitry senses an active pulse transition on a given timer input and provides appropriate output transitions to the rest of the module. The active transition can be configured to be low-to-high or high-to-low.

EEPROM: Electrically Erasable Programmable Read Only Memory; has the capability to be programmed and erased under direct program control.

Extended Addressing mode: An addressing mode with an 16-bit range.

General Addressing mode: An addressing mode with an 8-bit range.

Halt mode: The Halt mode 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.

IDLE instruction: The IDLE instruction causes the device to enter one of three modes; Idle, Halt, or Standby.

Idle mode: In the Idle mode, the CPU stops processing and waits for the next interrupt.

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 which causes the execution of one operation; consisting of an opcode and operands along with optional labels

interrupts: A signal input to the CPU to stop the flow of a program and force the CPU to execute instructions at an address corresponding to the source of the interrupt. When the interrupt is finished, the CPU resumes execution at the point where it was interrupted.

INT1 pin: A pin connected to external devices to allow them to interrupt the CPU; INT1 can be software configured as a non-maskable interrupt.

INT2 and INT3 pins: Pins connected to external devices to allow them to interrupt the CPU.

Isosynchronous Communications mode: An SCI mode in which data transmission is synchronized by a clock signal (SCICLK) common to both the sender and receiver. The format is identical to the asynchronous mode and consists of a start bit, data bits, an optional parity bit and a stop bit.

machine code: The actual bytes read by the CPU during an instruction execution *usually read by a programmer as hexadecimal bytes.*

MC pin: Mode Control pin, the voltage on this pin during Reset determines the operating mode of the TMS370 device; 12 volts on the MC pin after reset places the processor in the Write Protection Override mode (WPO).

memory map: A description of the addresses of the various sections and features of the TMS370 processor. The map depends on the operating mode.

Microcomputer mode w/external expansion: An operating mode in which the address, control and data buses extend off-chip to access external memory or peripherals.

Microcomputer single-chip mode: An operating mode in which the device uses only on-chip memory.

Microprocessor mode w/ internal program memory: An operating mode in which the on-chip program memory is available to the processor.

Microprocessor mode w/o internal program memory: An operating mode in which the on-chip program memory is not available to the processor; thus, the processor must have external memory.

μP/μC Mode bit: Microprocessor / Microcomputer Mode bit; determines whether the device initializes into one of the microcomputer or a microprocessor operating mode.

mnemonic: A symbol chosen to aid human memory; commonly used to refer to the symbol representing the opcode part of an assembly language instruction.

multiprocessor communications: A SCI format option which enables one processor to efficiently send blocks of data to other processors on the same serial link.

nested interrupts: The ability of an interrupt to suspend the service routine of a prior interrupt; implemented in TMS370 devices by executing an interrupt service routine

which uses the EINT, EINTL or EINTH instructions to set the global Interrupt Enable bits in the status register.

non-maskable interrupt (NMI): activation of a NMI always causes the processor to execute the NMI routine. On TMS370 devices; INT1 can be configured as an NMI.

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 which 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 which tells the programmer where the CPU will fetch or store data.

output compare: See Compare register.

Peripheral File (PF): The 256 bytes of memory, starting at 1000h, containing the registers which control the on-board peripherals and system configuration.

peripheral file frame: A set of 16 contiguous peripheral file registers, usually related by function.

powerdown mode: One of two power reduction modes; see Halt mode and Standby mode.

PPM: Pulse Position Modulation; a serial signal in which the information is contained in the frequency of a signal with a constant pulse width. A TMS370 device can output a PPM signal with a constant duty cycle without any program intervention using the Timer compare features.

prescaler: A circuit which slows the rate of a clocking source to the counter. On TMS370 devices, the prescaler can slow the clocking source by a factor of 4, 16, 64, or 256.

privilege mode: A mode immediately following reset in which the program can alter the privileged registers. Once the priviledged mode is disabled, these registers cannot be changed until another reset. This mode does not affect the EEPROM or the Watchdog registers.

prototyping device: A device used before masked ROM devices are available which have identical functions, pinout, size and timings. Programmable memory such as EEPROM or EPROM is used in place of the masked ROM.

pulse accumulation: A Timer 1 mode which keeps a cumulative count of SYSCLK pulses gated by the T1EVT signal.

PWM: Pulse Width Modulation; A serial signal in which the information is contained in the width of a pulse of a constant frequency signal. A TMS370 device can output a PWM signal with a constant duty cycle without any program intervention using the Timer compare features.

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; but all conversion values keep the same relationship to V_{REF}.

Register File (RF): The first 256 bytes of memory which can be accessed by the majority of the instrucions.

relative: Operands and code which produce an absolute address at some distance from the current location.

RESET pin: A low level on this pin starts hardware initialization and ensures an orderly software startup. If the MC pin is low when the RESET signal returns high, then the processor enters the Microcomputer mode. If the MC pin is high when the RESET signal returns high, then it enters the Microprocessor mode.

Serial Communications Interface (SCI): The SCI module is a built-in serial interface which 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.

SCICLK pin: Serial Communications Interface Clock pin; used as a synchronizing clock input or output in the Isoynchronous mode, or as a general purpose I/O pin.

Serial Peripheral Interface (SPI): The SPI module is a built-in serial interface which facilitates communication between networked master and slave CPUs. As in the SCI, the SPI is setup by software and 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.

SPICLK pin: Serial Peripheral Interface Clock. If the SPI is in the Master mode, this pin provides the serial clock for the entire serial communications network. If the SPI is in the Slave mode, this pin is the serial clock input.

SPISIMO pin: Serial Peripheral Interface Slave In, Master Out; In the master mode, data is output on the SPISIMO pin on the first SPICLK edge and latched from the SPI-SOMI pin on the opposite edge of SPICLK. In the slave mode, data is output on the SPISOMI pin on the first SPICLK edge and latched from the SPISIMO pin on the opposite edge of SPICLK.

SPISOMI pin: Serial Peripheral Interface Slave Out, Master In; see SPISIMO.

Stack: That part of the Register File used as last-in, first-out memory for temporary variable storage; used during interrupts and calls, to store the current program status. The area occupied by the stack is determined by the Stack Pointer and the application program.

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

Standby mode: A power reduction mode in which the CPU stops processing, but the on-chip oscillator remains active. Timers remain active and can cause the CPU to exit the standby mode.

Status register (ST): A CPU register which monitors the operation of the instructions and contains the global interrupt enable bits.

TRAP: Trap-to-subroutine. An assembly language instruction which is a one-byte subroutine call. An operand <n> is a trap number that identifies a location in the trap vector table, addresses 07FC0h to 07FDFh in memory, containing the address of the subroutine.

T2IC1/CR pin: Timer 2 Input Capture 1 / Counter Reset pin. A Timer 2 module pin which is an input to the counter reset, input capture, or PWM circuit.

T2IC2/PWM pin: Timer 2 Input Capture 2 / Pulse Width Modulation pin. A Timer 2 module pin which is the Pulse Width Modulation output or a second input capture.

unsigned integer: a number system used to express positive integers.

WAIT pin: 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 clock out signal and continues with the program.

wait states, automatic: extra clock cycles inserted automatically on every external memory access to accomodate peripherals or expansion memory with slower access time than the TMS370 processor. These Wait states are governed by two control bits: PF AUTO WAIT and AUTOWAIT DISABLE.

watchdog timer: A Timer 1 module option which can be programmed to generate an interrupt when it times out. This function provides a hardware monitor over the software to prevent a "lost" program. If not needed as 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.

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

V

W

X

Z

1

TEXAS INSTRUMENTS

Printed in U.S.A.
March 1988

SPNS01