TEXAS INSTRUMENTS

TMS370 Family Application Board

Technical Reference

1993

8-Bit Microcontroller Family

TEXAS INSTRUMENTS

Printed in U.S.A., June 1993
2615131-9761 revision *

SPNU029

Book Type
Two Lines
Volume #

Book Type
Volume #

Book Type
Two Lines

Title
Two Lines

year

Book Type

Title

year

TMS370 Family Application Board Technical Reference

June 1993
SPNU029

TEXAS INSTRUMENTS

IMPORTANT NOTICE

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

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

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

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

IMPORTANT NOTICE

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

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

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

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

WARNING

This equipment is intended for use in a laboratory test environment only. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to subpart J of part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.

Read This First

About This Manual

This manual describes the components of the TMS370 family application board and contains the following chapters:

Chapter 1 **Installing the Application Board and Invoking the Command Monitor.** Provides an overview and description of the functional blocks that compose the TMS370 application board; tells you how to install the board and invoke the command monitor.

Chapter 2 **Master Operation.** Discusses the functions of the master processor.

Chapter 3 **Interfacing Between the Slave and the TMS370 Application Board.** Describes how the slave devices interface with the application board.

Chapter 4 **Commands and Functions of the Command Monitor.** Describes the commands that you can use with the command monitor in TTY mode.

Appendix A **Jumper and Switch Settings.** Describes the location and function of each of the jumpers and switches on the TMS370 application board.

Appendix B **Using the TMS370 Assembly Language Tools With the Application Board.** Describes how the TMS370 assembler and code converter work with the application board.

Appendix C **Alphabetical Summary of Command-Monitor Messages.** Describes progress and error messages that the command monitor may display.

Appendix D **Schematics.** Contains schematics for the TMS370 family application board, its spare gates, its power source, the master controller, the master/slave buffer, the ’370Cx1x and ’370Cx5x slaves, the recreation ports, the protoarea connections, and the PAL descriptions of the master and slave signals.

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

Style and Symbol Conventions

This document uses the following conventions.

- Program listings, program examples, interactive displays, filenames, and symbol names are shown in a special typeface similar to a typewriter's. Examples use a **bold version** of the special typeface for emphasis; interactive displays use a **bold version** of the special typeface to distinguish commands that you enter from items that the system displays (such as prompts, command output, error messages, etc.).

  Here is a sample program listing:

```
.SECT "TIUSE",7FDEh  ;specify a new memory section
.WORD 7FE0h          ;Trap 0 vector
PUSH  ST             ;save status register
DINT                 ;stop all interrupts
BR    2000h          ;goto master control interface
JMP   $              ;used before WR and RR commands
RTI                  ;used before SS and RU commands
```

  Here is an example of a system prompt and a command that you might enter:

  >**LM 7000 700D 1 2 3 4** ↵

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

  DM *start* [*stop*]

  DM is the command. This command has two parameters, indicated by *start* and *stop*.

- Square brackets (**[** and **]**) identify an optional parameter. If you use an optional parameter, you specify the information within the brackets; you don't enter the brackets themselves. In the command syntax above, *stop* in an optional parameter.

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

  BL D {**1**|**0**}

  The region and fill value parameters for the BL command are not optional, but you need to make a choice between 1 and 0.

Information About Cautions

This is an example of a caution statement.

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

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

Related Documentation From Texas Instruments

The following books describe the TMS370 family devices and related support tools. To obtain a copy of any of these TI documents, call the Texas Instruments Literature Response Center at (800) 477–8924. When ordering, please identify the book by its title and literature number.

TMS370 8-Bit Microcontroller Family Assembly Language Tools User's Guide (literature number SPNU010) describes the assembly language tools (assembler, linker, and other tools used to develop assembly code), assembler directives, macros, common object file format, and symbolic debugging directives for the '370 family of devices.

TMS370 Family C Source Debugger User's Guide (literature number SPNU028) tells you how to invoke the TMS370 family application board and XDS versions of the C source debugger interface. This book discusses various aspects of the debugger interface, including window management, command entry, code execution, data management, and breakpoints, and includes a tutorial that introduces basic debugger functionality.

TMS370 Family Data Manual (literature number SPNS014) describes the TMS370 family of 8-bit microcontroller devices and covers topics such as pin functions, architecture, internal register structure, stack operation, instruction set, specifications, and operation of the TMS370 family modules.

Trademarks

CROSSTALK is a trademark of Microstuf, Inc.

Kermit is a registered trademark of Columbia University.

MS-DOS is a registered trademark of Microsoft Corp.

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

PROCOMM is a registered trademark of Datastorm Technologies Inc.

If You Need Assistance. . .

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

FCC Warning

This equipment is intended for use in a laboratory test environment only. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to subpart J of part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.

Contents

Figures

Tables

Chapter 1

Installing the Application Board and Invoking the Command Monitor

The ’370 family application board emulates ’370 microcontroller functions by using ’370 microprocessors. This chapter provides an overview and description of the functional blocks that compose the TMS370 application board. In addition, the chapter tells you how to install the ’370 application board and invoke the command monitor.

If you plan to install the application board and use the C source debugger, use the installation instructions in the *TMS370 Family Application Board Installation Guide.*

Topic **Page**

1.1 Overview of the TMS370 Application Board

This section summarizes the major components of the application board and describes the board's operating modes.

Summary of components

The '370 application board has these major components:

- A **TMS370C250 master microprocessor** that communicates with the host PC terminal via the RS232-C port, controls all of the application board functions, and schedules tasks to be performed by the slave. Wait-state generator and transceiver circuits enable both communication between the master and slave and also program control. The master's memory space is completely separate from the slave's memory.
- A 16K-byte **EPROM** (27C128) dedicated to the master for storing the command-monitor program (may be upgraded to 32K bytes).
- A separate **TMS370C256 slave microprocessor** that can function as an independent microprocessor. The slave executes your code, self-programs any internal memory such as EEPROM, and has its own dedicated memory space. This separate memory arrangement enables the slave to run your programs without interference from the command monitor system.
- **32K bytes of CMOS static RAM** (62C256) for the slave to store and run your programs.
- The **transceiver** (74HCT652) through which the master and slave communicate.
- The **slave wait-state generator** that allows the master to stop the slave upon any external memory cycle. You can use the wait-state generator in conjunction with the transceiver to read or write values directly on the slave's data bus.
- **Port recreation logic** that rebuilds the '370Cx5x slave's port A, B, C, and D functionality.

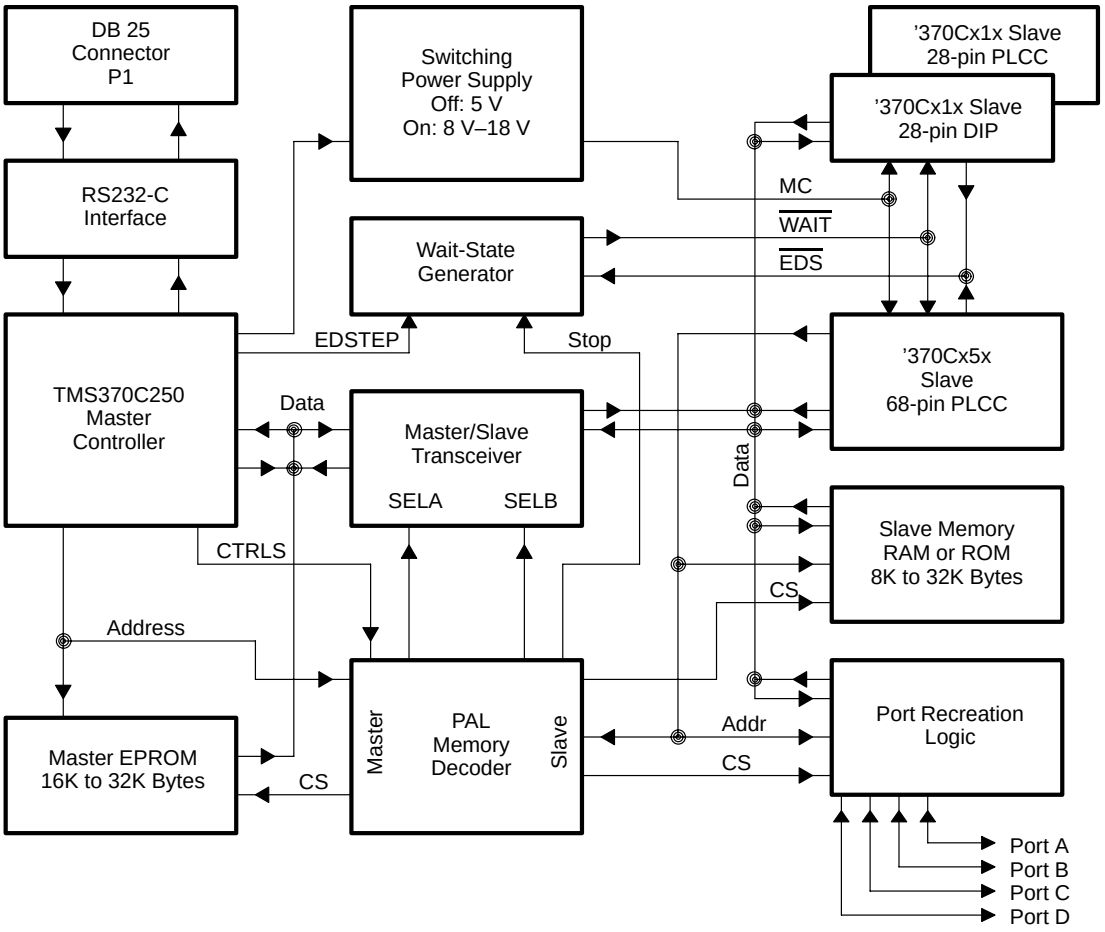
Other components of the application board include:

- A 28-pin PLCC socket and a 28-pin DIP socket for '370Cx1x devices.
- A prototyping area surrounded by all signals from the 68-pin socket, 28-pin sockets, V_{CC}, and ground.

- ❑ RS232-C communications via MAX232 line driver and DB-25 connector.
- ❑ An adjustable 5-volt to 18-volt set-up power supply for write protect override (WPO) mode of slave devices.
- ❑ Programmable array logic unit (PAL) for master and slave memory decoding.
- ❑ Power terminals for a 5-volt power supply.

Figure 1–1 shows how the board components interact.

Figure 1–1. Application Board Block Diagram



Operating modes

The ’370 application board allows you to operate in one of three modes:

- **Debugging mode.** The debugger mode is the standard TI programmer’s interface. There are many advantages to using the debugger mode of operation:

  - Multilevel debugging
  - Powerful command set
  - Window-oriented interface
  - Fully configurable, state-of-the-art, window-oriented interface
  - Comprehensive data displays
  - On-screen editing
  - Patch assembly

  The debugging mode is described in detail in the *TMS370 Family C Source Debugger User’s Guide,* literature number SPNU028.

- **TTY mode.** The TTY mode allows you to communicate using an ASCII serial protocol with equipment such as “dumb” terminals, PCs running terminal interface programs (for example, CROSSTALK or PROCOMM), or computers that are not IBM-compatible. This mode is most useful when you don’t have access to an IBM-compatible computer or if you need to do only simple operations or programming. The TTY mode also has several commands that allow you to check the operation of an application board.

  Additionally, you can perform the following memory operations in the TTY mode; these operations aren’t available in the debugging mode. You execute these commands by using the command-monitor interface.

  - Run system tests
  - Find bytes
  - Compare memory
  - Block Program
  - Directly read the analog pin
  - Load on-chip EPROM from on-board EPROM
  - Use a computer in “dumb” terminal mode
  - Reverse assemble with cycle times

- **Isolated mode.** After you have written and debugged a program for a slave, you can check the accuracy of that program by using the isolated mode. This mode “turns off” all debugger software and allows the slave device to run on its own.

1.2 What You'll Need

The following checklists detail items that you'll need in order to use the application board in TTY mode. The environmental requirements for the operation and storage of the application board are also described in this section.

Hardware checklist

☐	**host**	An IBM PC/AT or 100% compatible PC with a hard-disk system, serial port, and floppy-disk drive
☐	**memory**	Minimum of 512K bytes
☐	**display**	Monochrome or color
☐	**application board power requirements**	5 volts at approximately 500 mA
☐	**cable**	An RS232 serial cable with one male DB25 connection (25 pins).
☐	**optional hardware**	A plastic-leaded chip carrier (PLCC) extraction tool to remove processors from sockets

Note:

Be sure that you are using firmware version 1.50 or later.

Software checklist

☐	**emulator program**	If you plan to use the application board in TTY mode, you'll need a terminal emulator program (such as CROSSTALK or PROCOMM). However, you can use a standard TTY terminal in place of the PC and terminal emulator program if there are no data uploads or downloads.

Environmental requirements

Temperature and humidity requirements for both operation and storage of the '370 application board are shown in Table 1–1:

Table 1–1. Temperature and Humidity Ranges

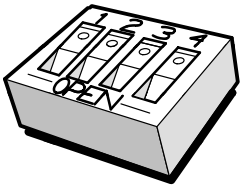
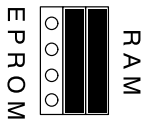
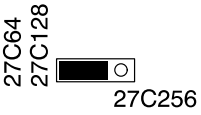
Operation		Storage	
Temperature	Humidity	Temperature	Humidity
60° F to 140° F 16° C to 60° C	30% to 80%	–40° F to 185° F –40° C to 85° C	5% to 90%

1.3 Step 1: Preparing the Application Board for Installation

Before you can use the application board, you must be sure that the board's switches and jumpers are set to configure your board correctly. Appendix A, *Jumper and Switch Settings,* describes the various jumpers and switches.

To use the application board in TTY mode, you need to set the switches and jumpers to the positions shown in Table 1–2:

Table 1–2. Switch and Jumper Settings for the TTY Mode

Board Designator	Setting	
SW2 (four switches):		
❑ SW2.1	OFF (default)	
❑ SW2.2	OFF (default)	
❑ SW2.3	OFF (default)	
❑ SW2.4	OFF (default)	
JP1	RAM (default)	
JP2	27C128 (default)	
JP3	µPROC/WPO (default)	
JP4	ON (default)	Jumper remains on the board

Note:

Before connecting the application board (as described in the following section), ensure that all chips are firmly placed in their respective sockets.

1.4 Step 2: Connecting the Cables

After you have set the jumpers and switches on the application board, you must connect the board to the host PC and to a power source.

Connecting to the host PC

An RS232 cable using only pins 2, 3, and 7 of connector P2 is sufficient to communicate with the ’370 application board. Note that you can power up the host RS232 port before the application board without any damaging effects. To connect the ’370 application board to the host PC or terminal, follow the pin assignments shown in Table 1–3.

Table 1–3. RS232 Cable and Pin Assignments for Connector P2

	Application Board Signals (Required)			Host	
Function	**Pin**	**Signal**		**Signal**	**Pin**
Data to board	2	Rx	←	Tx	2
Data to host	3	Tx	→	Rx	3
Signal ground	7	GND	—	GND	7
	Application Board Signals (Not Required)			**Host**	
Function	**Pin**	**Signal**		**Signal**	**Pin**
Attention to host	5	–9 volts	→	CTS	5
Board ready	6	9 volts	→	DSR	6
Connection established	8	9 volts	→	DCD	8
Attention to board	4†	CTS	←	RTS	4
Terminal ready	20†	DTR	←	DTR	20

† Connected, but not used by the ’370 application board monitor.

Note:

The 12.5 volts that you need to program ’370 EEPROM and EPROM devices are generated by an on-board switching power supply. A MAX232 line driver device generates RS232-C voltage levels from the 5-volt power supply. These features eliminate the need for external +12-V and –12-V power supplies. The ’370 application board power supply connections are also shown in Table 1–4.

Connecting to the power supply

To use the application board, you need to provide a 5-V, 500-mA regulated power supply. Even though the application board circuitry uses less than 300 mA of current, the power supply must be sufficient to power any additional circuitry that you may install on the prototyping area.

Note:

The ’370 application board does not provide voltage regulation or current limitation of the 5-volt signal.

Connect the power supply to the application board at connector P1, a two-pole terminal block. The wire sizes and the current ratings of the terminal block are shown in Table 1–4.

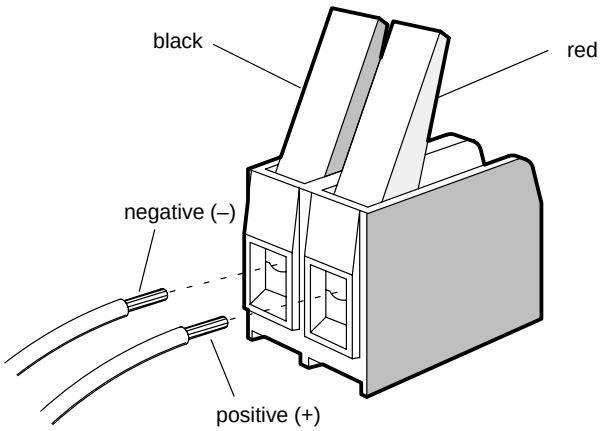
Table 1–4. Application Board Power Supply Connector P1

Terminal Post on P1	Signal	Wire Range (Solid or Stranded)	Maximum Current Into Terminal
Red lever	5 volts+/– 10%	18–24 AWG	5 amperes
Black lever	Ground	18–24 AWG	–5 amperes

Connect the power supply to the terminal block by following these steps:

1) Turn off the 5-volt power supply.
2) Lift the red and black levers on the terminal post.
3) Insert a bare wire from the power supply inside the lifted levers (see Figure 1–2).

Figure 1–2. Connecting the Power Supply to Connector P1



4) Push the levers down.

1.5 Step 3: Setting Up the RS232-C Communications

The ’370 application board communicates with the host PC or terminal in compliance with the RS232-C protocol. As a result, the application board requires the RS232-C communications format shown in Table 1–5:

Table 1–5. Format for RS232-C Communications

Characteristic	Required Setting
Data length	8 bits
Number of stop bits	1
Parity	None
Busy	XON–XOFF
Speed	75 to 19200 bits per second (set by an autobaud routine on the application board)
Autobaud character	⏎ (first character after power-up or board reset)

The host must recognize the XON–XOFF busy protocol to prevent the application board from being overrun with transmitted data. The application board firmware currently does not support hardware handshaking and busy protocols within the RS232-C standard.

A TTY terminal might allow you to use most of the command-monitor commands on the ’370 application board. However, you can enhance your developmental features by running a terminal emulator program such as CROSSTALK, which lets you download programs and blocks of data.

When downloading data to the application board, use the Intel Hex format. The application board also has a command to upload data in Intel Hex format (see the DH command description on page 4-12). You can convert the assembler output to the proper format by using the code conversion utility. The ’370 assembler and code conversion utility are included for the development of programs; however, the ’370 application board design kit does not include the linker.

1.6 Step 4: Applying Power

After you have connected the application board to the host PC and to the power supply, you should verify that you have correctly installed the application board and debugger software.

CAUTION

Always ensure that there are no metal objects beneath the ’370 application board; they might short the V_{CC}, ground, or other signals.

Always check your prototype designs for V_{CC}-to-ground shorts before applying power.

Follow these steps to verify the installation:

1) Turn on the 5-volt power supply.

2) Check to see that the LED is blinking. This indicates that the master processor is operating correctly. If the LED is not blinking, reset the application board by pressing and releasing SW1, a red-handled toggle switch.

When you apply power, the master places the ’370Cx1x and ’370Cx5x slave devices in a reset state. At power-up, the master selects the ’370Cx5x as the default slave device. The master then takes control of the slave ’370Cx5x device and waits for your first command from the RS232 port.

1.7 Step 5: Invoking the Command Monitor

1) Check to be sure that the host RS232 data format is set according to the application board's requirements. These requirements are defined in Section 1.5.

2) Press ↵ to set up the communication rate (autobaud routine).

3) Check for the correct display of the '370 application board help screen:

```
--------------------------------------------------------------------------------
 TMS370 FAMILY APPLICATION BOARD – Ver 1.50 (C) 1988–1993 Texas Instruments
--------------------------------------------------------------------------------
SL XXX  – Select processor type xxx/0   DR – Display Registers
IN      – Initialize processor          DB – Display breakpoints
RR      – Reset and run                 WR – Wait for reset and run
UH start stop – Upload Hex format       HE – Help: print this list
DH data – Download Hex data             VH data – Verify Hex with memory
MR      – Modify PC,ST,SP,A and B       RU aaaa – Run at address aaaa
MM addr              –Modify memory at addr (SP,CR,–,.,Q,ESC)
MB                   –Modify breakpoints (5 max) 0=delete (CA clears all)
MT                   –Modify Trace registers; Rxx or Pxx; 0=delete
SS xxxx              –Single step xxxx instructions– cr=1 (Rxxxx=repeat)
RA start xxxx        –Reverse assemble xxxx instructions from start
AN a b               –Read Analog channel a with reference b
BL D/P 0/1           –Block program Data/Program with 0/1
CM start stop dest   –Compare memory start to stop against dest
DM start stop        –Display memory from start to stop
FM start stop cc     –Fill memory from start to stop with cc
LM start stop a b    –Locate bytes (6 max) from start to stop for a b...f
TM start stop dest   –Transfer memory from start to stop, to dest
XM start stop        –Shift between external and Internal Program memory
DEVICE IS 370C050
>
--------------------------------------------------------------------------------
```

❑ If the help screen is not readable, but characters appear:

- Check the RS232 data formats.
- Check to see if the host XON–XOFF busy protocol is set ON.
- Reset the board to initialize the master.
- Press ↵ as the first character to set autobaud rate.

- ❑ If you don't see the help screen characters:
  - ■ Ensure that the LED is blinking after you reset the board.
  - ■ Check the RS232 cabling.
  - ■ Check the host RS232 connections and verify the proper RS232 operation of the host.
  - ■ Remove the RS232 cable from the application board and connect pins 2 and 3 of the cable together. Type any character; the same character should be echoed to the host.
  - ■ Check to be sure that SW2.1 and SW2.2 are OFF.

4) If the '370 application board LED does not blink:

- ❑ Check proper voltages and polarity at P1 and the 5-volt power supply.
- ❑ Ensure that the master, U2, is placed correctly in the socket.
- ❑ Check to see that the master EPROM, U3, is seated correctly in the socket.
- ❑ Check for metal objects between the table top and the board; they might cause a short circuit.
- ❑ Check that the application board reset switch, SW1 (red-handled toggle switch), is functioning correctly. The master's $\overline{\text{RESET}}$ pin (pin 53) should go low upon activation of the board reset switch; then it should go high when the switch is released.
- ❑ Check to see if the LED or the connecting inverter is bad.
- ❑ Check for the CLKOUT signal on pin 58 of the master. It should appear as a 4.9152-MHz square wave. If this is not present, then the master is not working.

Chapter 2

Master Operation

The master processor is responsible for communicating with the host, controlling all of the functions of the application board, and scheduling tasks for the slave. This chapter discusses the functions of the master processor and includes the following topics:

Topic **Page**

2.1 Functions of the Master Processor

As its name implies, the master processor controls all ’370 application board functions. These functions include:

- **Executing the command monitor.** The master executes the system command monitor that resides in the dedicated 16K-byte EPROM (U3). The command monitor contains the user/host interface, slave communication, slave control, and test software routines.
- **Changing the operating mode.** Upon reset, the master scans switch 2 (SW2) for the operating mode that you selected.
- **Communicating with the host through the RS232 serial port.** When you are testing and debugging your programs, the master waits for you to input commands via the RS232 serial port. You can select the slave device type, and the master releases only the selected device from its reset state for program execution. The other slave device remains in reset until you select it.

  Through the RS232 link, the master can accept commands and control the ’370Cx1x or ’370Cx5x devices. Whenever you need to download your object file from a host PC to the application board, the master reads the formatted object data and then sends the machine code to the slave for proper storage.
- **Communicating with the slave through the transceiver.** The master communicates with the slave through a memory-mapped transceiver, with access controlled by a wait-state generator circuit.
- **Controlling the slave through wait states.** The master controls the slave by transmitting a signal to the $\overline{\text{WAIT}}$ pin that forces the slave to extend the current external memory cycle. This causes the slave to halt, and the master can read or write values to the slave’s data bus.
- **Controlling slave programming.** The master can control the slave during programming by forcing the slave to execute a TRAP 0 instruction. This instruction activates the $\overline{\text{WAIT}}$ pin and causes the slave to halt. The master can then send command-monitor commands to the slave through the transceiver.

The following sections describe the functions of the master.

2.2 Executing the Command Monitor

The master processor executes the command monitor. The command monitor has numerous commands that allow you to check the operation of the application board. These commands help you to:

- Perform memory operations such as comparing, dumping, locating, and transferring memory
- Modify and display register data
- Manipulate the processor state, including:
  - Initialize the processor
  - Reset and run
  - Start program execution
  - Select the processor type
  - Single-step
- Set, clear, and list software breakpoints
- Use TTY mode commands to:
  - Read A/D channels
  - Download data into hexadecimal format
  - Display the help screen
  - Reverse assemble
  - Test board functions
  - Upload data into hexadecimal format
  - Verify host against master upload operation
  - Exchange external and internal memory

The commands that you can use to perform these functions are described in Chapter 4, *Commands and Functions of the Command Monitor.*

2.3 Changing the Operating Mode

Upon power-up reset or reset via the board reset switch (SW1), the master immediately places all slaves in a reset state. It then scans the switches in SW2 for the operating mode that you selected. SW2.1 and SW2.2 select the operating mode:

- **Debugging mode.** If you set SW2.1 to ON and SW2.2 to OFF, you enable the debugging mode. Debugging mode is the standard programmer's interface.
- **TTY mode.** If you set both SW2.1 and SW2.2 to OFF, the master initializes the application board for use with a standard TTY terminal or with a host computer running a terminal emulator package (for example, Kermit, CROSSTALK, PROCOMM).
- **Isolated mode.** If you set SW2.1 to OFF and SW2.2 to ON, you enable the isolated mode. Isolated mode allows you to run the slave without control of the master.

SW2 is described in full in Appendix A.

2.4 Communicating With the Host Through the RS232 Serial Port

The master communicates through the serial communications interface module (SCI) to the host PC via standard RS232-C communications protocol. If you want to download code, a host computer (PC) can be connected to the application board.

In order to emulate a standard TTY terminal, the host computer must execute a terminal emulator program such as CROSSTALK, PROCOMM, or Kermit. These programs are available as shareware on certain bulletin board systems or are commercially available. However, only a standard terminal is needed for the TTY terminal (command monitor) interface provided.

Interfacing to the ’370 application board requires the following RS232-C data format:

Table 2–1. Format for RS232 Communications

Characteristic	Required Setting
Data length	8 bits
Number of stop bits	1
Parity	None
Busy	XON–XOFF
Speed	75 to 19200 bits per second (set by an autobaud routine on the application board)
Autobaud character	⏎ (first character after power-up or board reset)

The host must recognize the XON–XOFF busy protocol to prevent the application board from being overrun with transmitted data. When the application board wants the host to stop sending data, it sends the XOFF character (13h ASCII) to the host. When ready to resume data transmission, the application board sends the XON character (11h ASCII) to the host. The host should use the same method to start and stop the application board while it transmits data.

The application board can support communication speeds from 75 to 19200 bits per second (baud). Upon a power-up reset or activation of the board reset switch, the master scans the RS232-C port for the first incoming character. The first character should always be a carriage return, 0Dh ASCII. From this character, the communication speed can be established between the host PC and the application board.

The RS232-C interface uses a MAX232 device to generate the RS232 voltage levels and current drive of the TTL-level inputs and outputs of the master SCI. The MAX232 device uses only the 5-volt board supply to generate the RS232 levels of approximately +9 volts and –9 volts. As a result, the ’370 application board requires only one external 5-volt power supply.

The application board has the RS232-C control lines shown in Table 2–2 connected to the master. The command monitor uses only pins 2, 3, and 7 (transmit, receive, and ground, respectively). The other signals are provided for future revisions of the command monitor firmware.

Table 2–2. RS232 Pin Assignments on DB25 Connector P2

Connector P2 Pin Number	Direction Input or Output	Cabling Requirements	Signal
1	x	Optional	Ground
2	I	Required	Serial data from host
3	O	Required	Serial data to host
4	I	Future	RTS from host (not used)
5	O	Future	–9V (CTS to terminal)
6	O	Optional	+9V (DSR to terminal)
7	x	Required	Ground
8	O	Optional	+9V (DCD to terminal)
20	I	Future	DTR from host (not used)

Note: Application board functions with only pins 2, 3, and 7.

2.5 Communicating With the Slave Through the Transceiver

The master and slave communicate through a 74HCT652 octal bus transceiver that connects the two data buses together. This transceiver is the only communication channel between master and slave. The transceiver contains two sets of octal latches, each of which can be used to store data between the master and slave. The application board configures the transceiver to:

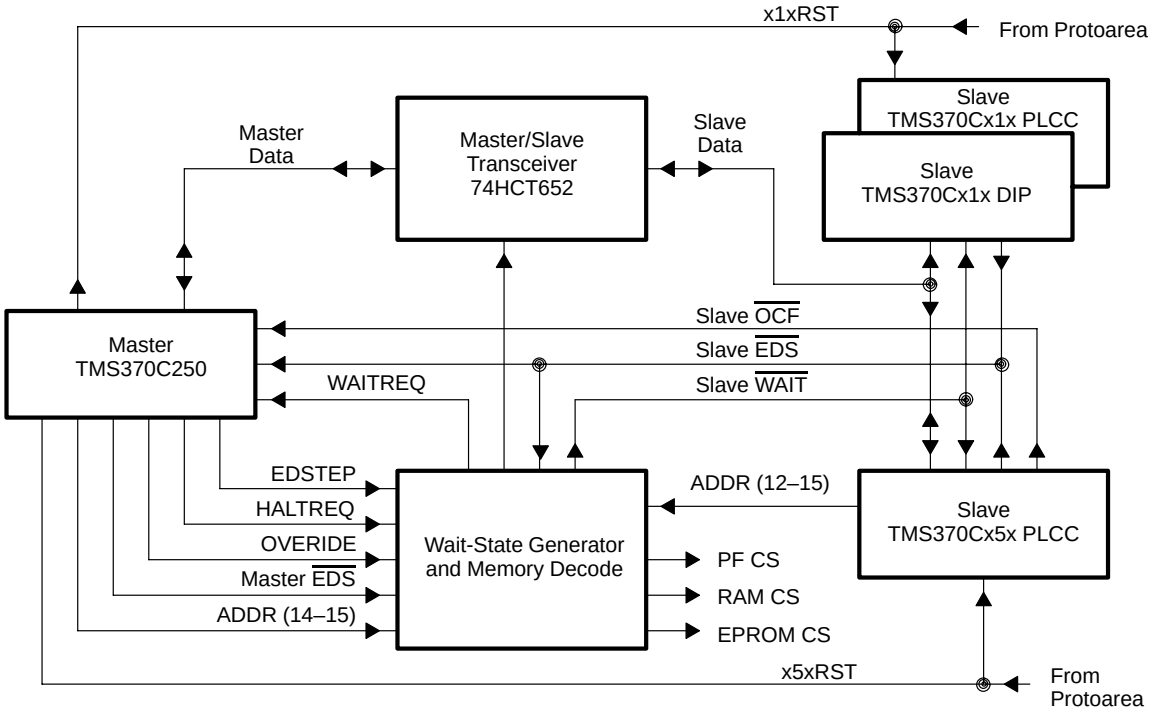
- Latch and store all data written to the transceiver by the master.
- Provide realtime data (the current slave data bus value) to the master upon a master transceiver read cycle.
- Provide latched and stored data to the slave upon a slave read cycle at memory location 2xxxh or at any memory location whenever the master applies OVERIDE.
- Ignore all slave write cycles because the master can read the slave data bus. The master can extend the slave's data bus values for any length of time by manipulating the slave $\overline{\text{WAIT}}$ pin.

Opcodes, operands, and data can be passed between the master and the slave. Most slave instructions are executed within the 2xxxh address range because the wait generator circuit automatically applies $\overline{\text{WAIT}}$ within this address range. While a slave instruction is in this range, the master can send it by writing to the transceiver and providing a rising edge on the master I/O line EDSTEP. This removes the $\overline{\text{WAIT}}$ signal and allows the slave to complete the read cycle of the opcode sent by the master. At the falling edge of the next $\overline{\text{EDS}}$ (external data strobe) cycle, the wait-state generator puts the slave into a wait state, and the master can then send the instruction's first operand.

The master signal OVERIDE is used only when the slave's memory address is outside the 2xxxh range. It disables all off-chip memory devices connected to the slave (RAM, and port logic) and allows the master to OVERIDE or place the transceiver's data on the bus. The master can then force the slave to branch to address location 2xxxh and remove the OVERIDE signal, which allows the off-chip memory devices to operate.

Figure 2–1 illustrates the communication between the master and slave.

Figure 2–1. Slave-Master Communication



The signals shown in Table 2–3 and illustrated in Figure 2–1 are generated by the master and control master-to-slave communication.

Table 2–3. Master-to-Slave Communication

Signal	Functional Description
HALTREQ	Halts slave at its next active $\overline{\text{EDS}}$ low cycle.
OVERIDE	Disables slave external memory and port recreation logic registers. On the next slave read cycle, it also places the master's data that was previously latched and stored within the 74HCT652 onto the slave's data bus.
EDSTEP	On this signal's rising edge, it releases the WAITREQ signal. The slave can halt at the next active low $\overline{\text{EDS}}$ cycle with the proper conditions, independently of the steady state value of EDSTEP.
x5xRST	Master resets the '370Cx5x slave when this signal is active low.
x1xRST	Master resets the '370Cx1x slave when this signal is active low.
x5xRSTEN	Master connects the protoarea pin RS5 to the slave '370x5x reset pin via a bidirectional analog switch (not shown in Figure 2–1).
x1xRSTEN	Master connects the protoarea pin RS1 to the slave '370x1x reset pin via a bidirectional analog switch (not shown in Figure 2–1).

The signals shown in Table 2–4 control slave-to-master communications. These signals are input directly to the master either by the slave or by the wait-state generator circuit.

Table 2–4. Slave-to-Master Communication

Signal	Functional Description
$\overline{\text{OCF}}$	This standard ’370 bus control signal goes low at the beginning an opcode fetch memory cycle. This signal is generated by the slave ’370Cx5x only and is connected directly to the master.
$\overline{\text{EDS}}$	This standard ’370 bus control signal goes low during any off-chip memory cycle. This signal is generated by both the ’370Cx1x and ’370Cx5x slaves. It is used to validate the memory address, wait timing if $\overline{\text{WAIT}}$ is to be applied, and master synchronization.
WAITREQ	This signal is generated by the wait generator circuit and is gated with the protoarea pin WAI to produce the $\overline{\text{WAIT}}$ pin value for both the ’370Cx1x and ’370Cx5x slaves. When this signal is low, the current slave device is in a wait state, and the master can take control of the slave.

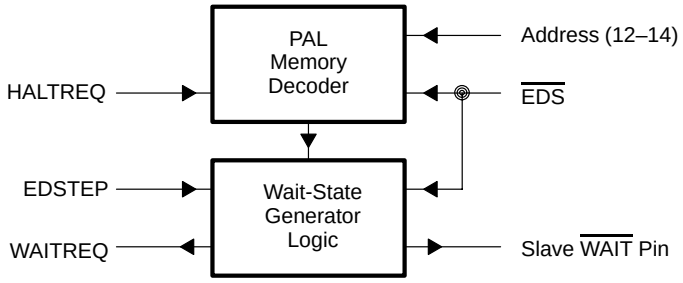
2.6 Controlling the Slave Through Wait States

The master controls the slave by transmitting a signal to the slave $\overline{\text{WAIT}}$ pin. This pin, when active, forces the slave to extend the current external memory cycle until the $\overline{\text{WAIT}}$ signal is removed. The slave can enter a wait state only during an external memory cycle, which is indicated by the $\overline{\text{EDS}}$ signal going low. $\overline{\text{EDS}}$ is not activated while accessing internal on-chip RAM, data EEPROM, or internal on-chip program memory.

While the slave is in a wait state, the slave’s address bus, data bus, $\overline{\text{EDS}}$, R/$\overline{\text{W}}$, and OCF (opcode fetch) pins continue to hold their respective logic values. This allows the master to effectively stop the slave upon any external memory cycle and, through the use of the transceiver, read or write values directly onto the slave’s data bus. Although the slave is placed in a wait state, the slave’s internal clocks continue to function and operate all on-chip peripherals. As a result, an enabled timer continues to increment during wait states, and an unexpected timer interrupt could occur within your software during the master’s control of the slave.

The slave’s $\overline{\text{WAIT}}$ pin is activated by a block of logic that is referred to as the wait-state generator. This logic, coupled with the PAL memory decoder, allows either a master I/O line, HALTREQ, or the slave’s current address to halt the slave’s operation at the falling edge of the slave $\overline{\text{EDS}}$ signal. See Figure 2–2.

Figure 2–2. Wait-State Generator



The PAL memory decoder is configured to generate the STOP signal to the wait-state generator circuit on either of the two following conditions:

- The master signal HALTREQ (halt request) is set low.
- The slave address is within the 2000h to 2FFFh range.

The wait-state generator circuit sets the slave's $\overline{\text{WAIT}}$ pin low upon the next falling edge of $\overline{\text{EDS}}$, that is, whenever the STOP signal is active from the PAL. The slave wait state is latched and held by the wait-state generator until the master releases it. The master releases the slave from the wait state by sending a rising edge (low to high) on the master's I/O line EDSTEP. The slave can then complete the current memory cycle and continue to operate until another wait state is applied. Moreover, the slave can be stepped from $\overline{\text{EDS}}$ cycle to $\overline{\text{EDS}}$ cycle with the wait-state generator and whenever the STOP signal is active.

You have access to the slave's $\overline{\text{WAIT}}$ pin through the protoarea pin WAI on connector P3. This line is ANDed with the wait-state generator's output and sent directly to the slave's $\overline{\text{WAIT}}$ pin. This allows your design to implement a wait circuit in the protoarea. See the *Memory wait interfacing* subsection, page 3-17, for more information on how to interface to this protoarea signal.

2.7 Controlling Slave Programming

The slave is controlled by the master through the $\overline{\text{WAIT}}$ line and the transceiver, as described in Sections 2.5 and 2.6. To enable the slave to execute system monitor commands, the master causes the slave to execute a TRAP 0 instruction to invoke a subroutine. The subroutine invoked by the TRAP 0 command begins at TI reserved memory location 7FE0h and ends by branching to 2000h. In this way, the master gains complete control over the slave since all memory accesses at addresses 2000h to 2FFFh activate the slave's $\overline{\text{WAIT}}$ pin and halt the slave. No actual memory resides within this address range (2xxxh); however, the transceiver is memory-mapped to these addresses so that data can be transmitted between the master and slave. Through the transceiver, the master can send the slave instructions to perform the required monitor commands.

Note:

You should never design memory-mapped peripherals within the 2000h to 2FFFh range when using the '370 application board. The application board does **not** provide a means for disabling the wait generator circuit and transceiver from this address range.

Software breakpoints are implemented in a manner similar to the description above. The monitor command MB replaces your data in alterable memory (RAM or EEPROM) with a TRAP 0 instruction. The opcode that was replaced is stored within the master's register file for future replacement within the slave's memory. When the slave executes the TRAP 0 instruction, control is returned to the master. The master can then perform monitor commands such as memory inspects or modifications.

When running the slave from a software breakpoint, the master sends your stored opcode to the slave via the transceiver in place of the TRAP 0 instruction. The slave is then allowed to execute freely. The TRAP 0 instruction remains resident within slave's memory until the software breakpoint is removed by the monitor command, MB, at which time your original opcode is restored.

To implement slave program control and software breakpoints, you must follow these guidelines:

- ❑ Do not use the TRAP 0 instruction in your program.
- ❑ Three bytes of the slave's stack space are required for program counter (PC) and status register (ST) storage.

- Slave memory at 7FDEh through 7FEDh is required for the TRAP 0 vector and subroutine instructions. Do not use these addresses in your program. The memory locations 7FE0h to 7FEBh are reserved for TI use on mask ROM parts.
- Slave memory addresses 2000h to 2FFFh are reserved for the wait-state generator and transceiver use.

Chapter 3

Interfacing Between the Slave and the TMS370 Application Board

This chapter describes how the slave devices interface with the application board.

Topic **Page**

3.1 Slave Operation

The application board supports 68-pin and 28-pin slave devices. Additionally, you can run programs that access and use the following modules contained within the ’370 microprocessor:

- ❑ Timers 1 and 2
- ❑ Serial communications interface (SCI)
- ❑ Serial peripheral interface (SPI)
- ❑ Analog-to-digital (A/D) converter
- ❑ Data EEPROM
- ❑ Program EPROM (for operation and programming)

All slave I/O pins are routed to protoarea connectors so that you can evaluate new circuit designs. The slave operates in either the microprocessor mode (as shipped) or the microcomputer mode:

- ❑ In the **microprocessor mode**, the slave uses a CMOS RAM to store the program code for execution. You can choose to remove the CMOS RAM and replace it with an EPROM for the final target code checkout. The microprocessor mode also allows the master to control and interrogate the slave through a memory-mapped transceiver.

  Since the slave must convert many of its I/O pins into an address/data bus, the application board recreates the lost I/O ports. These recreated ports are mapped to the same memory locations as the on-chip ports. Functionalities and differences among these ports are described in Section 3.4, page 3-8.

- ❑ You can select the **microcomputer mode** for the slave by setting a board jumper (JP3). In this mode, the slave executes entirely from internal memory, and the master cannot halt the slave. The master simply resets the selected slave and allows it to run without further intervention.

  You must be careful not to address the CMOS RAM, port logic, or the transceiver with the discrete I/O lines; ensure that D6 ($\overline{\text{EDS}}$) remains a logic high. This will disable all memory devices.

Slave-processor module operation

The slave modules continue to operate during software breakpoints and master control sequences. As a result, internal slave modules such as timers 1 and 2, SPI, SCI, and A/D operate all the time and cannot be halted. The internal clocks of the slave device, unlike those of the in-circuit emulators (XDS or CDT), cannot be stopped. This may cause unexpected results while single-stepping programs. Also, this may severely limit timer development for a “run to breakpoint” mode.

Command monitor operation and slave memory

Several types of slave memory, along with restrictions of commands contained within the master command monitor, are listed in Table 3–1.

Table 3–1. Memory Restrictions

	Off-Chip		On-Chip			
	RAM	**EPROM**	**Reg. File**	**ROM**	**EEPROM**	**EPROM**
Software breakpoints	Yes	No	Yes	No	Yes	No
Single step	Yes	†	No	No	No	No
Run commands	Yes	†	†	†	†	†
Halt/stop program	Yes	Yes	No	No	No	No

† You must include the following instructions in program memory at the defined location for correct operation:

```
.SECT    "TIUSE",7FDEh
.WORD    7FE0H
PUSH     ST
DINT
BR       2000h
JMP      $
RTI
```

28-pin slave operations

The application board provides sockets for 28-pin device types primarily for on-chip EEPROM and EPROM programming. Due to the lack of a complete address/data bus, only a limited number of application board commands will execute on the ’370Cx1x devices. For example, 28-pin slave devices:

- Cannot be single-stepped
- Cannot be breakpointed
- Are used primarily for programming on-chip memory

As a result, all program development/debugging should be done using the 68-pin slave device and then transferred to the on-chip memory of the 28-pin slave device.

Microcomputer mode limitations of slave devices

The application board provides you with a method to configure the slaves in microcomputer mode. When the slave operates in this mode, the on-chip ports A, B, C, and D operate as I/O pins. You must use caution when in this mode: don't allow the I/O line to be used for address selection of the external memory devices. The best preventive measure is to insure that the slave $\overline{\text{EDS}}$ pin remains high.

Slave devices in operating microcomputer mode:

- ❑ Must not configure I/O pin D7 as an output.
- ❑ Should configure I/O pin D6 as a constant logic high output.

CAUTION

The slave I/O pin, D7/$\overline{\text{CSE1}}$/$\overline{\text{WAIT}}$, is driven by a logic device on the application board. This I/O pin must not be configured as an output; if it is, damage can occur to the slave.

These cautions and recommendations apply to both the '370Cx1x and '370Cx5x slaves.

3.2 Interfacing With the Prototyping Area

A prototyping area (or “protoarea”) on the application board allows you to interface circuitry to the ’370. Adjacent to the protoarea are 150 connection holes that contain all of the signals needed for interfacing to the slaves. These holes are divided into three groups of 50 pins, with each group arranged to fit a 25×2 header or connector. Protoarea connectors P3, P4, and P5 connect the signals to the slaves:

- The signals on connectors P3 and P4 are connected directly to all the slaves on the application board. The 28-pin slaves connect to a subset of the P3 and P4 signals. Since these signals are unbuffered, you should be careful when connecting prototype designs.
- Connector P5 contains the 40 reconstructed I/O port signals, the ’370Cx1x slave prototype reset, and the ’370Cx1x CLKIN signal. Several V_{CC} and GND connections are provided to allow connection to the application board 5-volt power supply.

Standoff holes are provided on the application board for extender legs. With extender legs, you can use standard wire-wrap sockets in the protoarea. The standoff holes accept a standard 4×40 screw.

The ’370 application board also provides upper standoff holes for connection to a piggyback board. The upper standoff holes are 0.152 inches in diameter and can accept plastic board connectors. The piggyback board can be electrically connected to the slave pins with 25×2 male and female connectors in the protoarea connectors P3, P4, and P5.

Table 3–2 lists by connector and pin value all signals available on the ’370 application board.

Table 3–2. Protoarea Connector (All Available Signals)

Microprocessor P3				Module Outputs P4				Reconstructed I/O Ports P5			
VCC	1	2	GND	IN1	1	2	VCC	VCC	1	2	GND
A0	3	4	A1	IN2	3	4	VCC	PA0	3	4	PA1
A2	5	6	A3	IN3	5	6	GND	PA2	5	6	PA3
A4	7	8	A5	T1I	7	8	GND	PA4	7	8	PA5
A6	9	10	A7	T1P	9	10	–	PA6	9	10	PA7
VCC	11	12	GND	T1E	11	12	–	VCC	11	12	GND
B0	13	14	B1	T2I	13	14	–	PB0	13	14	PB1
B2	15	16	B3	T2P	15	16	–	PB2	15	16	PB3
B4	17	18	B5	T2E	17	18	–	PB4	17	18	PB5
B6	19	20	B7	SPO	19	20	–	PB6	19	20	PB7
C0	21	22	C1	SPI	21	22	–	VCC	21	22	GND
C2	23	24	C3	SPC	23	24	VCC	PC0	23	24	PC1
C4	25	26	C5	SCT	25	26	VCC	PC2	25	26	PC3
C6	27	28	C7	SCR	27	28	GND	PC4	27	28	PC5
VCC	29	30	GND	SCC	29	30	GND	PC6	29	30	PC7
D0	31	32	–	VC3	31	32	VS3	VCC	31	32	GND
D1	33	34	–	AN7	33	34	VS3	PD0	33	34	PD1
D2	35	36	GND	AN6	35	36	VS3	PD2	35	36	PD3
D3	37	38	GND	AN5	37	38	VS3	PD4	37	38	PD5
D4	39	40	–	AN4	39	40	VS3	PD6	39	40	PD7
D5	41	42	–	AN3	41	42	VS3	VCC	41	42	GND
D6	43	44	–	AN2	43	44	VS3	–	43	44	–
D7	45	46	WAI	AN1	45	46	VS3	CK1	45	46	–
VCC	47	48	RS5	AN0	47	48	VS3	–	47	48	–
CK5	49	50	SMC	VC3	49	50	VS3	RS1	49	50	–
25 × 2				25 × 2				25 × 2			

3.3 Interfacing With the TMS370 Modules

Each module of the ’370 has three or more pins dedicated to module functions. These pins interface directly from the ’370 slave to the protoarea connector (PAC) without buffering or other alteration. The module signals on connector P4 and the corresponding pins on the ’370 slaves are shown in Table 3–3.

Table 3–3. Module Output Signals on Protoarea Connector P4

P4		’370		
Connector Pin	**Label**	**Signal Name**	**’370Cx5x Pin**	**’370Cx1x Pin**
1	IN1	INT1	52	16
3	IN2	INT2	51	17
5	IN3	INT3	50	18
7	T1I	T1IC/CR	46	22
9	T1P	T1PWM	45	21
11	T1E	T1EVT	44	20
13	T2I	T2IC1/CR	27	–
15	T2P	T2IC2/PWM	26	–
17	T2E	T2EVT	25	–
19	SPO	SPISOMI	49	25
21	SPI	SPISIMO	48	23
23	SPK	SPICLK	47	24
25	SCT	SCITXD	30	–
27	SCR	SCIRXD	29	–
29	SCK	SCICLK	28	–
33	AN7	AN7	43	–
35	AN6	AN6	42	–
37	AN5	AN5	41	–
39	AN4	AN4	40	–
41	AN3	AN3	39	–
43	AN2	AN2	38	–
45	AN1	AN1	37	–
47	AN0	AN0	36	–
31,49	VC3	V_{CC3}	34	–
32,34,36	VS3	V_{SS3}	35	–
38,40,42	VS3	V_{SS3}	35	–
44,46,48	VS3	V_{SS3}	35	–
50	VS3	V_{SS3}	35	–

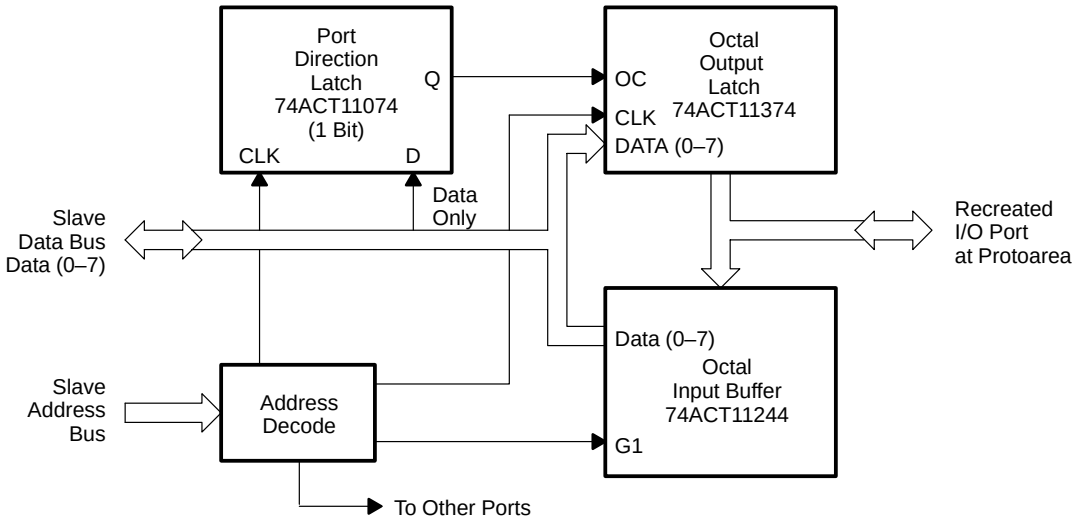
Note: Each analog channel has a V_{SS3} connection on the adjacent pin to help reduce noise. See Section 3.9 for a discussion on interfacing analog signals to the application board.

3.4 Interfacing With I/O Ports

The slave runs in microprocessor mode in order to communicate to the master and to external memory. While the slave is in the microprocessor mode, I/O ports A, B, C, and D on the ’370Cx5x slave function as an address/data bus, and memory control signals and are not available as general-purpose I/O lines. Since the primary objective of the ’370 application board is to allow you to evaluate the operation of ’370 microprocessors, the application board provides logic to recreate functions of the general-purpose I/O ports while the slave operates in microprocessor mode. All of the recreated port signals are available at the protoarea on connector P5.

The I/O ports are recreated with 74ACT logic devices. Each 8-bit port consists of a 74ACT11244 buffer to read the port, a 74ACT11374 latch to write out from the port, and a D flip-flop (74ACT11074) to control the data direction of the port. Each port register address is decoded by the application board PAL and two 74HCT138 3-to-8 decoders. See Figure 3–1.

Figure 3–1. Port Recreation



Each I/O port is memory mapped at the same address as the I/O ports that reside on the slave. As a result, using these ports is transparent to your program if you consider the following features of the recreated ports:

- **Byte-wide direction instead of bit-wide.** Each I/O port is byte-wide direction programmable, whereas the on-chip slave ports are bit-wide programmable. As a result, all eight port lines can be programmed as outputs, or all eight ports can be programmed as inputs.
- **Write-only port data direction registers (DDR).** You can’t read the stored DDR contents of the recreated ports.

- **Bit 0 of the DDR of a port controls the direction of the 8-bit port.** When you write to the DDR register of a port, bits 1–7 are not used in the port direction circuitry. However, when programming, you must write 0xFF to the DDR, even though only one bit is used. This allows your program to be transported to the internal EPROM of a ’370 device so that it can operate correctly in the microcomputer mode.

- **Differing electrical characteristics.** Since all four ’370 ports are reconstructed from discrete logic devices, their electrical characteristics are different from those of the 74ACT logic ports. These differences are illustrated in Table 3–4 (refer to the *TMS370 Family Data Manual* for more details on the electrical characteristics).

Table 3–4. Electrical Characteristics

Parameter	’370 Ports A, B, C, and D			74ACT Logic Ports			
	Min	Max	Test Condition	Min	Max	Test Condition	Unit
V_{ih}	2.0	V_{CC}	2.0				V
Vil	V_{SS}	0.8			0.8		V
V_{oh}	0.9 V_{CC}		I_{oh}=–50 μA	4.4		I_{oh}=–50 μA	V
V_{oh}	2.4		I_{oh}=–2 mA	3.8		I_{oh}=–24 mA	V
V_{ol}		0.4	I_{ol}=2 mA	0.44		I_{ol}=24 mA	V
I_i	+/–10			+/–6			μA

- **Port data register operation.** The port data register operates in the same manner as the on-chip slave data register.

- **Port control registers 1 and 2 not implemented.** Port control registers 1 and 2 are not decoded and are not implemented on the application board. The recreated ports provide input/output functionality only. The ’370 functions A and B are not implemented. If address/data bus connections are needed, then the protoarea connections should be made directly to the slave’s A, B, C, or D port pins provided on P3.

While operating in the microprocessor mode, the ’370Cx5x handles port addresses 1020–102F like external memory. Any read or write operation to a port address on the application board is mapped to the circuit logic composing the external port recreation. The application board provides a read/write data register and a write-only data direction register for each of the recreated ports.

Note:

The port recreation pins are not functional when you are using internal memory (P011.2 = 0). The main function of the internal memory is programming and not executing code. Always use external memory when using the port recreation pins.

Table 3–5 lists each of the recreated port pins on the protoarea connector P5. These signals do not connect to the ’370 slaves; they are generated by application board logic. For each of the recreated port pins, a corresponding on-chip slave port pin is shown as a guide. This allows you to have I/O operations of the slaves, even though they are configured to the microprocessor mode.

Table 3–5. Port Recreation on Protoarea Connector P5

P5		Emulation	
Connector Pin	**Connector Label**	**’370Cx5x Pin**	**’370Cx1x Pin**
3	PA0	17	14
4	PA1	18	13
5	PA2	19	11
6	PA3	20	10
7	PA4	21	9
8	PA5	22	8
9	PA6	23	7
10	PA7	24	3
13	PB0	65	—
14	PB1	66	—
15	PB2	67	—
16	PB3	68	—
17	PB4	1	—
18	PB5	2	—
19	PB6	3	—
20	PB7	4	—
23	PC0	5	—
24	PC1	7	—
25	PC2	8	—
26	PC3	10	—
27	PC4	11	—
28	PC5	12	—

Table 3–5. Port Recreation on Protoarea Connector P5 (Continued)

P5		Emulation	
Connector Pin	**Connector Label**	**’370Cx5x Pin**	**’370Cx1x Pin**
29	PC6	13	—
30	PC7	14	—
33	PD0	64	—
34	PD1	60	—
35	PD2	59	—
36	PD3	58	28
37	PD4	57	26
38	PD5	56	15
39	PD6	55	1
40	PD7	54	2

3.5 Using Microprocessor Mode

All of the slave pins are routed directly and unbuffered to the protoarea. The slave port A, B, C, and D pins are connected to protoarea connector P3. These slave pins can operate as simple I/O lines in the microcomputer mode or as an address/data/control bus in the microprocessor mode. Since the slave must operate in the microprocessor mode in order to have master and slave control as well as use of external memory, the pins will be an address/data/control bus.

Since all of the signals are present at the protoarea, you can connect extra memory and memory-mapped peripherals to the slave in the protoarea. The memory map of the ’370 application board is shown in Figure 3–6, page 3-18. You should consult the application board schematics in Appendix D to determine bus-loading characteristics before designing to the slave memory bus. Bus signals are used mainly to connect to the prototype area and are **not** to be connected to off-board circuitry.

Table 3–6 lists the microprocessor pins on protoarea connector P3 and the corresponding ’370 slave pins to which they are connected on the application board.

Table 3–6. Microprocessor Signals on Prototype Connector P3

P3		’370		
Connector Pin	**Connector Label**	**Signal Name**	**’370Cx5x Pin**	**’370Cx1x Pin**
3	A0	A0/Data0	17	14
4	A1	A1/Data1	18	13
5	A2	A2/Data2	19	11
6	A3	A3/Data3	20	10
7	A4	A4/Data4	21	9
8	A5	A5/Data5	22	8
9	A6	A6/Data6	23	7
10	A7	A7/Data7	24	3
13	B0	B0/Addr0	65	–
14	B1	B1/Addr1	66	–
15	B2	B2/Addr2	67	–
16	B3	B3/Addr3	68	–
17	B4	B4/Addr4	1	–
18	B5	B5/Addr5	2	–
19	B6	B6/Addr6	3	–
20	B7	B7/Addr7	4	–
21	C0	C0/Addr8	5	–
22	C1	C1/Addr9	7	–
23	C2	C2/Addr10	8	–
24	C3	C3/Addr11	10	–
25	C4	C4/Addr12	11	–
26	C5	C5/Addr13	12	–
27	C6	C6/Addr14	13	–
28	C7	C7/Addr15	14	–
31	D0	D0/$\overline{\text{CSE2}}$/OCF	64	–
33	D1	D1/$\overline{\text{CSH3}}$	60	–
35	D2	D2/$\overline{\text{CSH2}}$	59	–
37	D3	D3/CLKOUT	58	28
39	D4	D4/R/$\overline{\text{W}}$	57	26
41	D5	D5/$\overline{\text{CSPF}}$	56	15
43	D6	D6/$\overline{\text{CSH1}}$–/$\overline{\text{EDS}}$	55	1
45	D7	D7/$\overline{\text{CSE1}}$–/$\overline{\text{WAIT}}$	54	2

3.6 Protoarea Mode Control, Clock, Reset, and Wait Signals

The protoarea signals listed in Table 3–7 allow unique customization of the slave operation on the ’370 application board to supply special external control to slaves. The subsections that follow describe each of these signals in detail.

Table 3–7. Miscellaneous Signals on Protoarea Connectors

Prototype Area			’370		
Connector	**Pin**	**Label**	**Signal Name**	**’370Cx5x Pin**	**’370Cx1x Pin**
P3	50	SMC	MC	6	19
P3	49	CK5	XTAL2/CLKIN	31	—
P5	45	CK1	XTAL2/CLKIN	—	5
P3	48	RS5	RESET	53	—
P5	49	RS1	RESET	—	27
P3	46	WAI	PROTO WAIT†	54	2

† This signal is combined with the application board’s wait signal before it is sent to the slave’s $\overline{\text{WAIT}}$ pin. The slave’s wait pin, D7/$\overline{\text{CSE1}}$/$\overline{\text{WAIT}}$, can be directly connected at P3, pin 45.

Mode control pin—SMC

Pin 50 of the protoarea connector P3 gives you access to the slave’s mode control pin. This pin is connected to both slave MC pins and also to the center pin of jumper JP3. Jumper JP3 connects the SMC signal (slave mode control) to the on-board switching power supply (μPROC/WPO) or to ground (μCOMPUTER).

The switching power supply provides 5 volts to the slave at all times, except during master-controlled EEPROM or EPROM write cycles. This configures the slave to the microprocessor mode during the low-to-high transition on the slave $\overline{\text{RESET}}$ pin. When the master processor instructs the slave to perform an EEPROM or EPROM write cycle, the switching power supply is turned on, and 12.5 volts are applied to the slave MC pins.

The output voltage of the switcher can range from 8 volts to 18 volts. The switcher voltage can be set by adjusting potentiometer R3 on the application board. The monitor test command T0 facilitates adjustment via the master’s A/D converter (see page 4-27 for more information).

CAUTION

You must remove JP3 before adjusting the switching power supply voltage. Damage may occur if more than 13 volts are applied to the slave devices.

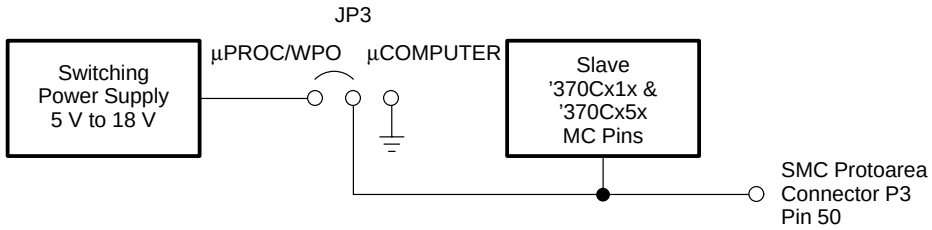
To obtain greatest accuracy of the switcher voltage measurement with the T0 command, make sure that the application board's supply voltage is 5.0 volts. The T0 measurement will drift accordingly with the main 5-volt power supply. Although the T0 measurement readings will drift, the switch's output voltage will remain constant during V_{CC} changes. A V_{CC} change will only induce T0 measurement errors.

CAUTION

If your protoarea design is controlling or driving the SMC line, you must remove jumper JP3.

When JP3 is set to μCOMPUTER, the slave MC pins are grounded, and the slave operates in the microcomputer mode. This slave mode can be used when the application board is set to the isolated mode as described in Section A.3, *Setting the Operation Mode of the Master (SW2),* page A-5. The connections of the protoarea signal SMC to the slaves JP3 and the switching power supply are illustrated in Figure 3–2.

Figure 3–2. Mode Control Circuit



CLKIN circuits

The application board provides a method to supply the system clock to the slaves from multiple sources. Both the '370Cx5x and '370Cx1x slave crystals are socketed; therefore, they can be replaced by different values. The '370 application board provides two 12-pF capacitors per crystal for the correct operation of a parallel resonant crystal in both the X2 and X3 crystal sockets.

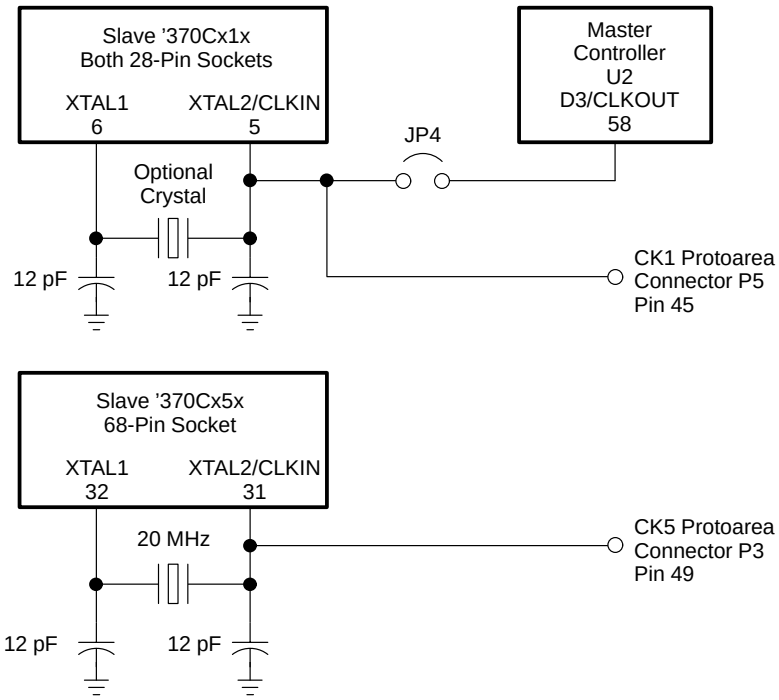
If the X3 crystal is removed, the ’370Cx5x can receive the CLKIN signal from a clock source within the prototype area. This clock source should be connected to the protoarea connector P3, pin 49 (CK5). A crystal circuit in the protoarea is not recommended, because of long signal paths. The signal should be driven from a gate or an oscillator.

The ’370Cx1x slave is normally used only for programming purposes and not for executing your code. If you decide to operate the application board in the isolated mode with a ’370Cx1x slave device, you have three different options for supplying the CLKIN signal to the ’370Cx1x slave.

- Run the ’370Cx1x at one-fourth the master processor speed by using the master CLOCKOUT signal. The master’s CLOCKOUT signal is connected to the ’370Cx1x device with jumper J4.
- Remove JP4 and insert a crystal into the X3 crystal sockets.
- Remove JP4 and provide a clock source from the protoarea. The clock source levels must meet the ’370Cx1x data sheet parameters. The clock source is connected to protoarea connector P5, pin 45 (CK1). As with the CK5 connection on the ’370Cx5x slave, a crystal circuit is not recommended.

Figure 3–3 illustrates the CLKIN circuits.

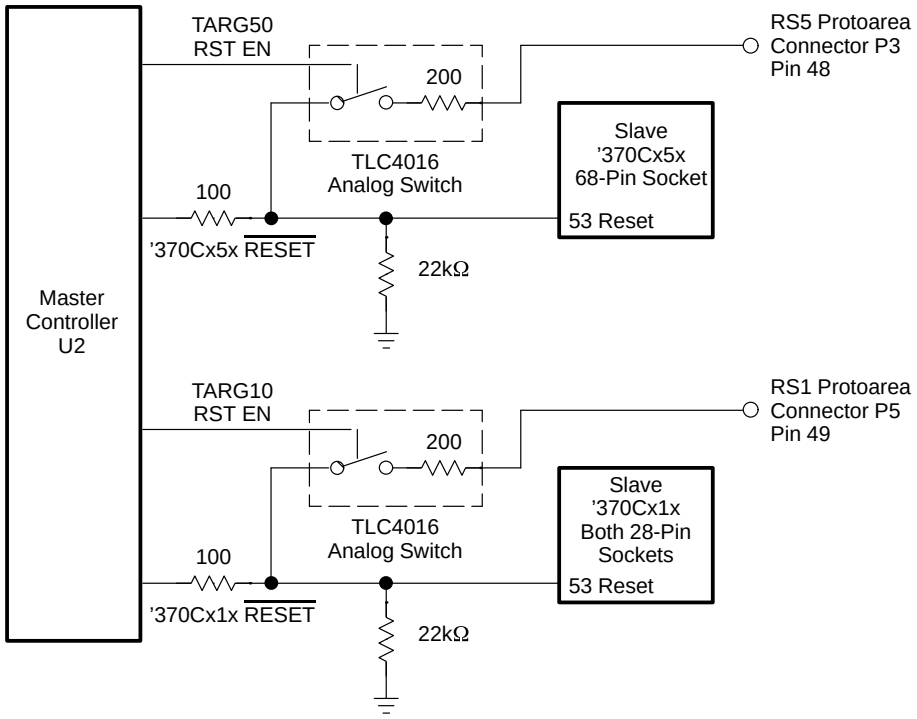
Figure 3–3. CLKIN Circuits



Slave reset circuits

The master processor normally has full control over the slave's reset, but the application board can be set to give this control over to the protoarea. The '370Cx1x and '370Cx5x have separate but identical reset circuits as shown in Figure 3–4.

Figure 3–4. '370Cx1x and '370Cx5x Reset Circuits



Each of the protoarea reset signals, RS5 and RS1, is connected to the proper slave socket pin by the analog switch. The analog switch has a series impedance of 200 ohms per channel. The '370 application board also has a 22-kΩ resistor to ground on each of the slave reset lines. These resistors insure that both slaves remain in reset while the board reset switch is closed and the master is in reset. These resistors can be removed if only one slave is installed per board. Otherwise, both '370Cx1x and '370Cx5x devices could operate during a master reset, and damage could occur. The series impedance of the analog switch and the 22-kΩ pull-down resistors should be taken into account for a prototype-reset design.

In order for the master to close the proper analog switch (U18), SW2.3 and SW2.4 must be set. For a complete description of SW2 settings and the corresponding application board operation, see Appendix A.

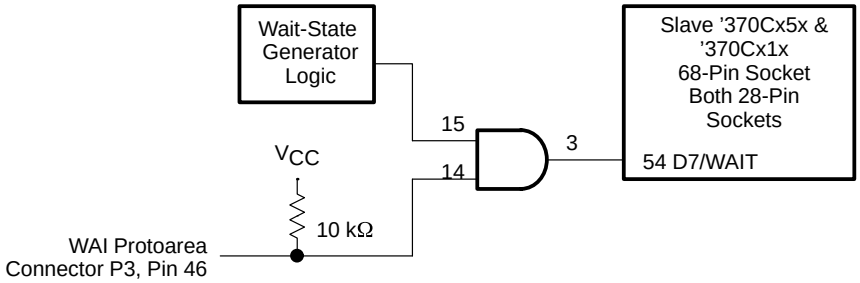
When SW2 is set to enable a prototype-reset signal, the master enables the proper analog switch (U18) and then connects the I/O line to the slave $\overline{\text{RESET}}$ pin high impedance to remove it from the circuit. Therefore, circuitry in the protoarea must be provided to ensure a high level when the master releases reset. This circuitry can be as simple as a 2-kΩ resistor to V_{CC} or as complex as a voltage-sensing power supply supervisor circuit.

For the isolated mode of application board operation, the master processor releases the slave's reset circuit approximately one millisecond after releasing the reset switch.

Memory wait interfacing

The '370 application board provides you with a method of applying wait states to the slave. Since the master controls the slave's operations with the slave's $\overline{\text{WAIT}}$ pin, the wait-state generator and the protoarea wait pin WAI (pin 46 on connector P3) are gated together to produce the final wait signal. This is shown in Figure 3–5. This operation allows the protoarea and the wait-state generator to share control of the slave $\overline{\text{WAIT}}$ pin. The slave's $\overline{\text{WAIT}}$ pin, D7, can be monitored on P3, pin 45 (D7).

Figure 3–5. The Wait Interface



The slave's $\overline{\text{WAIT}}$ pin is driven by an AND gate, 74ACT11008. The switching characteristics for this gate are listed in Table 3–8. The 74ACT11008 timing delays must be incorporated into the protoarea wait design. This ensures that the proper wait setup times are provided to the slave. The wait setup time, as defined by the data sheet, must be maintained for proper slave operation.

Table 3–8. Switching Characteristics of 74ACT11008

Parameter	From (Input)	To (Output)	Min	TYP	Max	Unit
T_{PLH}	A or B	Y	1.5	5.8	8	ns
T_{PHL}			1.5	5.2	7.7	

Note: V_{CC} = 5 V; T_A = 25 °C

The slave can enter a wait state when **either** the protoarea signal goes low or the wait-state generator goes low. The protoarea signal, WAI (P3 pin 46) is pulled high by a 10-kΩ resistor, so it will be ignored when no circuitry is attached. Care should be exercised when using this signal because the master uses it extensively to control most of the slave's operations.

3.7 Slave Memory Mapping

Memory devices and peripherals can be easily interfaced to the application board using the signals on the protoarea connectors. Since the bus signals generated from a 20-Mhz crystal are switched so quickly, you should place the high-speed circuits, such as memory, on the protoarea. The partitioning of memory and the physical location of memory for the application board is shown in Figure 3–6. More detailed information about the '370 microprocessor memory map is given in the *TMS370 Family Data Manual.*

Figure 3–6. '370 Application Board Slave Memory Map

```
0x0000 to 0x01FF  RAM
0x0200 to 0x0FFF  Reserved
0x1000 to 0x101F  Peripheral registers
0x1020 to 0x102F  Port reconstruction
0x1030 to 0x107F  Peripheral registers (x1x ends at 0x104F)
0x1080 to 0x10BF  Reserved
0x10C0 to 0x10FF  Available peripheral expansion
0x1100 to 0x1DFF  Reserved
0x1E00 to 0x1FFF  Data EEPROM
0x2000 to 0x2FFF  Used by slave/master communication
0x3000 to 0x3FFF  Available for expansion
0x4000 to 0xBFFF  Address range of slave's memory socket
0xC000 to 0xFFFF  Available for expansion
```

The instruction TRAP 0 is not available for software development. It is reserved for software breakpoints within the command monitor. The memory locations 2000 to 2FFFh and 7FDE to 7FEBh (TRAP0 vector and TI reserved space) are also reserved for command monitor usage. In addition, the application board command monitor uses 3 bytes of stack space in the slave environment.

External and internal memory areas

The ’370C7xx devices come with programmable on-chip program memory. This internal memory is mapped into the same addresses as the slave’s external static RAM. For normal program development, you must work with the static RAM for program storage because RAM memory is faster to use and has unlimited write cycles. When developing code for another application, you must program the internal memory of the ’370 slave, rather than the static RAM on the application board, for final checkout.

You can use the SL (select device) command to select which memory to work with. You have the option of addressing external RAM memory or internal program memory for those slave devices with alterable program memory. If you choose internal memory, then all memory commands in the program memory range address the on-chip program memory instead of the external RAM. If you choose external memory, then you don’t have immediate access to the internal program memory; you use the external memory instead.

When you’re working with internal memory in the stand-alone mode, the] prompt is used; when the commands use external memory, the > prompt is used.

3.8 Zero-Ohm Resistors

Slaves receive V_{CC} power through zero-ohm resistors. You can make accurate power measurements by replacing a zero-ohm resistor with a current meter. If slave power control is necessary, the resistors can be replaced by a switch. Make the switch connection with 22-gauge wire or larger.

The A/D power pins of the '370Cx5x slave can be supplied from two different sources:

- Through the zero-ohm resistors, R20 and R21. These resistors connect the A/D module power pins to the application board ground and 5-volt supply, respectively.
- Through protoarea pins on connector P4 labeled VC3 and VS3. If you want to evaluate the A/D module with a new supply voltage on V_{SS3} and V_{CC3}, remove the zero-ohm resistors and supply the desired voltage through the VS3 and VC3 protoarea connectors on P4. More details on interfacing to the A/D module are given in Section 3.9.

Table 3–9 lists the zero-ohm resistors, the corresponding slave power pins connections, and the default configurations.

Table 3–9. Zero-Ohm Resistor Assignments

Resistor Designation	Function	Connections as Shipped
R17	'370Cx1x Digital V_{CC}	5 V
R18	'370Cx5x Digital V_{CC2}	5 V
R19	'370Cx5x Digital V_{CC1}	5 V
R20	'370Cx5x Analog V_{SS3}	Gnd
R21	'370Cx5x Analog V_{CC3}	5 V

3.9 Interfacing With the Analog-to-Digital Conversion Module

Given an 8-bit analog-to-digital (A/D) conversion module, the resolution over a 5-volt range is only about 20 mV. Therefore, certain features of the application board are designed to help you avoid inaccurate A/D conversions. These features include:

- ❑ Short signal paths from the processor and protoarea
- ❑ Separate power and ground shielding within the PWB planes
- ❑ Analog ground connections for every analog channel at the protoarea connector
- ❑ Provisions for separate analog power and ground
- ❑ Special routing of high-frequency signals away from the A/D channel lines

The A/D converter has power and ground separated from processor power and ground. You can incorporate filtering circuits between the board's power bus and the analog power bus to improve the noise level. The power and ground levels should not fall outside the levels listed in the electrical specifications chapter of the *TMS370 Family Data Manual;* otherwise, inaccuracy will result.

The A/D power pins of the '370Cx5x slave are connected to the application board 5-volt power and to the ground through zero-ohm resistors. If you want to use a separate power supply for V_{SS3} and V_{CC3}, then remove zero-ohm resistors R20 and R21. The V_{SS3} and V_{CC3} pins can then be supplied from the VS3 and VC3 protoarea pins on connector P4.

Make connections to the A/D channels through the protoarea connectors. All of the analog channels reside on P4 with the labels AN0–AN7. For best results, use coaxial or twisted pair cables for each analog channel with the ground wire connected to the adjacent VS3 pin on the connector. Also, avoid long wires whenever possible.

You should take all other standard precautions to insure a clean, accurate signal to the A-to-D pins. These precautions should be incorporated into all final '370 designs.

3.10 Programming Additional TMS370 Devices

The application board allows you to program the program memory of the ’370C756 and ’370C710 devices, in addition to programming other TMS370 family devices. However, you **cannot** program the following TMS370 devices with the application board:

- ’370Cx58 devices
- ’370Cx3x devices
- Any devices that have the hard watchdog option (refer to the *TMS370 Family Data Manual* for more information).

Some devices that you can program require the addition of another programming socket. If you intend to use the ’370 application board as a device programmer for a large numbers of devices, you can interface a high-quality, easy-insertion socket to the board. Table 3–10 lists the devices that you can program with the application board. The table also provides the pin that you can connect to the prototyping area, allowing you to add programming sockets.

Any programming socket that you add according to Table 3–10 is wired in parallel to the 28-pin slave socket. You must program the devices in these additional sockets in the same way that you would program a device in the standard 28-pin slave sockets. Make sure that the 28-pin programming sockets (U26 and U27) are empty when you use any additional programming sockets. You can add more than one programming socket to the application board, but only one socket is used at a time.

Table 3–10. Interfacing Additional Programming Sockets†

Pin Name	Suggested Location	28-Pin DIP 010N 610N 710N 710JD	28-pin PLCC 010FN 610FN 710FN 710FZ	40-Pin PDIP 020N 022N 040N 042N 622N 642N 722N 722JD 742N 742JD	40-Pin SDIP 020N2 022N2 040N2 042N2 622N2 642N2 722N2 722JC 742N2 742JC	44-Pin PLCC 020FN 022FN 040FN 042FN 622FN 642FN 722FN 722FZ 742FN 742FZ	64-Pin SDIP 056NM 256NM 756NM 756JN	68-Pin PLCC 050FN 052FN 056FN 250FN 256FN 756FN 756FZ
A0	P3-3	14	14	18	18	20	15	17
A1	P3-4	13	13	17	17	19	16	18
A2	P3-5	11	11	16	16	18	17	19
A3	P3-6	10	10	15	15	17	18	20
A4	P3-7	9	9	14	14	16	19	21
A5	P3-8	8	8	13	13	15	20	22
A6	P3-9	7	7	11	11	13	21	23
A7	P3-10	3	3	10	10	12	22	24
MC	P3-50	19	19	34	34	39	5	6
CLKIN	P5-45	5	5	33	33	38	29	31
CLKOUT (D3)	P3-37	28	28	21	21	23	55	58
$\overline{\text{RESET}}$	U27-27	27	27	5	5	6	51	53
R/$\overline{\text{W}}$ (D4)	P3-39	26	26	20	20	22	54	57
$\overline{\text{EDS}}$ (D6)	P3-43	1	1	22	22	24	53	55
$\overline{\text{WAIT}}$ (D7)	P3-45	2	2	19	19	21	52	54
V_{CC}	Protoarea	4	4	9	9	10, 11	31, 57	15, 61
GND	Protoarea	12	12	12	12	14, 29	8, 58	9, 16

† Device names that end in N, FN, N2, or NM are all preceded with “TMS370C”. Devices that end in JD, FZ, JC, or JN are preceded with “SE370C”.

Appendix D

Commands and Functions of the Command Monitor

This chapter describes the commands that you can use with the command monitor in TTY mode. Topics covered in this chapter include:

Topic **Page**

D.1 Entering Commands From the Command Line

All command-monitor commands consist of two letters followed by zero to eight parameters. After you type a command name, the monitor automatically prints a space and waits for the first parameter (if appropriate). Whenever more than one parameter is required, you must enter a SPACE before you type the next parameter entry.

- Unless stated otherwise in the command description, parameter values default to zero.
- If you enter a SPACE or ↵ instead of a parameter value, the parameter is set to the default value.
- When you enter a parameter value, press SPACE or ↵ to terminate the parameter. When you enter the last required parameter, the monitor executes the command. If another parameter is required, the monitor waits for your next parameter entry.

Entering numerical parameters

The command monitor treats all numerical inputs as hexadecimal numbers, so you don't need to enter any suffixes or prefixes with these inputs. The monitor ignores any nonhexadecimal characters.

When a 4-character address parameter is required, the monitor reads only the last four characters that you enter before the SPACE or ↵. This means that if you make a mistake when typing an address, you can continue typing and enter the correct 4-character address followed by a SPACE or ↵.

Escaping commands or long displays

If you want to cancel a command that you have started typing on the command line, press ESC. This causes the current command to be aborted, and the monitor returns with a new command-line prompt.

Note:

Some terminals may have buffers that continue to print after you press ESC. If pressing ESC does not cause the terminal to respond, reset the application board by toggling the reset switch (SW1).

Some commands such as DM (display memory) and RA (reverse assemble) have long displays that cause the screen to scroll. You can stop the scrolling by pressing ESC. If you want to display one line at a time, press SPACE.

D.2 Modifying Displayed Values

Some commands allow you to change a displayed value or move to another value. These commands are:

- MB: modify breakpoints
- MM: modify memory
- MR: modify registers
- MT: modify trace registers

After a value is displayed, you can change it by typing any valid hexadecimal number. After you have modified the value, you can do one of the following:

To do this...	Press
Accept the new value and display the next value	↵ or SPACE
Accept the new value and display the previous value	– (dash)
Accept the new value and redisplay it	. (period)
Accept the new value and exit the command	Q
Ignore the new value and exit the command	ESC

If you decide not to change a value once it's displayed, you can do one of the following:

To do this...	Press
Leave the value unchanged and display the next value	↵ or SPACE
Leave the value unchanged and display the previous value	–
Leave the value unchanged and repeat the displayed value	. (period)
Leave the value unchanged and exit the command	ESC

D.3 Prompts

The command monitor display has two types of prompts:

- The **external-memory** prompt looks like this: >. This prompt indicates that the memory commands you enter will be executed on the external off-chip program memory. For example, if you are developing code with external static RAM substituted for the on-chip ROM, you must use the external-memory prompt.
- The **internal-memory** prompt looks like this:]. This prompt indicates that the memory commands you enter will be executed on the internal on-chip program memory. For example, when you are accessing the program EEPROM, you must use the internal-memory prompt.

You can choose the prompt type by using the SL (select device) command (described on page D-24). When you use the SL command to choose the slave device, the monitor asks you if the program memory is external or internal. Press E for the external-memory prompt or I for the internal-memory prompt, followed by ↵.

The following examples illustrate how to change the prompt:

```
>SL 756 ↵
DEVICE IS 370C756
Program memory External/Internal (E/I) ? I ↵
]
]SL 756 ↵
DEVICE IS 370C756
Program memory External/Internal (E/I) ? E ↵
>
```

If the device that you choose with the SL command does not have alterable program memory, the prompt is selected by default:

```
]SL 310 ↵
DEVICE IS 370C310
]
```

D.4 Protecting the EEPROM Memory

The slave devices use the write protect override mode (WPO) in order to program write-protected EEPROM devices. You can use the WPO mode in a service environment to update protected EEPROM contents. Enter the WPO mode by placing 12 volts on the MC pin after reset. (Refer to the *TMS370 Family Data Manual* for a discussion about the MC pin.) If your application target system does not have 12 volts available, then you cannot write to program EPROM or to protected data EEPROM blocks. This allows you to have control over all of the slave memory.

D.5 Using Run and Single-Step Commands

When you are using the RR (reset and run), RU (run), SS (single-step), or WR (wait for reset and run) commands, the following limitations apply:

- These commands work effectively only with full address/data bus devices. As a result, you can use only the 68-pin device with these commands because it provides the address/data bus needed for control of program execution. If you try to use these commands with a 28-pin device, an error will occur.

  Since the 28-pin device is an exact subset of the larger 68-pin device, you should use the 68-pin device instead of a 28-pin device when performing application development work. Use the SL (select device) command to choose the appropriate 68-pin device.

Note:

If you try to stop the program while executing internal on-chip memory, you will see the following message:

```
Timeout, NO xxx FROM SLAVE
```

The xxx is $\overline{\text{OCF}}$, $\overline{\text{EDS}}$, or $\overline{\text{WAIT}}$.

- To use these execution commands, you must have a support program. The monitor automatically writes this program to the static RAM whenever it's needed. If you have replaced the static RAM with an EPROM, you must include the assembly language code shown in Figure D–1 in the EPROM in order to single-step or run.

Figure D–1.Program to Support Executing Commands

```
 .SECT ”TIUSE”,7FDEh  ;specify a new memory section
 .WORD 7FE0h          ;Trap 0 vector
 PUSH  ST             ;save status register
 DINT                 ;stop all interrupts
 BR    2000h          ;goto master control interface
 JMP   $              ;used before WR and RR commands
 RTI                  ;used before SS and RU commands
```

D.6 Using Software Breakpoints

The ’370 application board allows you to set software breakpoints to halt program execution. You can do this by replacing your opcode with a TRAP 0 opcode (EF). When the program executes the TRAP 0 breakpoint, the routine defined in Figure D–1 is executed.

A software breakpoint halts the slave program execution and places the master in control. Your program is halted before the instruction stored at the breakpoint address is executed. The master immediately reloads the stored user opcode within the slave program RAM or EEPROM. The master then waits for the next monitor entry.

Software breakpoints are normally transparent to you because the proper data is restored within the slave memory. Software breakpoint addresses may be seen as EFh when you dump or modify memory.

When you are using software breakpoints, the following limitations apply:

- A program cannot use the TRAP 0 instruction or its trap vector.
- The breakpoint must be on an instruction boundary.
- Slave memory must be alterable. ROM or EPROM will not work, but operations with EEPROM will work.

D.7 Functional Summary of Command-Monitor Commands

This section summarizes the command-monitor commands according to these categories:

- Modifying and displaying memory
- Modifying and displaying register data
- Manipulating the processor state
- Managing breakpoints
- Other TTY commands

Modifying and displaying memory

To do this	Use this command	See page
Block EEPROM memory	BL	D-10
Compare memory	CM	D-11
Dump memory	DM	D-13
Fill memory	FM	D-14
Locate a specific sequence in memory	LM	D-16
Modify memory at an address	MM	D-18
Transfer memory	TM	D-30

Modifying and displaying register data

To do this	Use this command	See page
Dump registers	DR	D-13
Modify registers	MR	D-19
Modify trace registers	MT	D-20

Manipulating the processor state

To do this	Use this command	See page
Execute a single instruction (single-step)	SS	D-25
Initialize the processor	IN	D-15
Reset and run	RR	D-22
Select the processor type	SL	D-24
Start program execution	RU	D-23
Wait for reset and run	WR	D-31

Managing breakpoints

To do this	Use this command	See page
Clear (delete) all software breakpoints	CA	D-10
Display a list of all the software breakpoints that are set	DB	D-11
Set and modify software breakpoints	MB	D-17

Other TTY mode commands

To do this	Use this command	See page
Display the help screen	HE	D-14
Download data into hexadecimal format	DH	D-12
Exchange external and internal memory	XM	D-32
Read A/D channels	AN	D-9
Reverse assemble	RA	D-21
Test board functions	T0 or T1	D-28 or D-29
Upload data into hexadecimal format	UH	D-30
Verify host against master upload operation	VH	D-31

D.8 Alphabetical Summary of Command-Monitor Commands

This section summarizes and describes of all primary monitor commands.

AN — *Analog-to-Digital Converter*

Syntax

AN *a b*

Description

The AN command displays the current analog-to-digital (A/D) value on the '370Cx5x slave analog pins.

- The *a* parameter is the analog channel number; you can use values 0 through 7.
- The *b* parameter is the reference channel range; you can use values 0 through 7. However, when you use channel 0, the value refers to V_{CC}; channel 0 is not available as V_{REF}.

The slave analog pins you specify for the channel and V_{REF} are changed to analog mode by this instruction.

After you have entered the AN command and the monitor has displayed the A/D value, you can repeat the operation with the same parameters by pressing SPACE or ↵. If you want to return to the monitor command line, press ESC.

Example

```
>AN 1 SPACE ↵   Read channel 1 with reference Vcc3
22 ESC          Value of 22 read on A/D channel # 1
                then return to command line

>AN 1 2 ↵       Read channel 1 with reference on channel 2
22 SPACE        Make another reading using same parameters
33 SPACE        Make another reading
44 ESC          Return to command line
```

Successive readings are actually displayed horizontally without scrolling; for example,

```
22 SPACE 33 SPACE 44 ESC
```

The prompt does not return to command line until you press ESC.

BL — *Block Program EEPROM Memory*

Syntax

BL D {**1**|**0**}

Description

This BL command fills or clears an entire EEPROM array by using the array programming described in the *TMS370 Family Data Manual*.

The BL command has two parameters:

- The region parameter (**D**) tells the monitor that you're block programming data EEPROM.
- The value parameter tells the monitor to fill the block with all ones (**1**) or all zeros (**0**).

If you set or clear an entire array using the BL command, subsequent writes that you make will take 10 ms instead of the 20 ms taken with an uninitialized array. This will save time with the FM (fill memory) command.

The BL command verifies that the operation took place by checking every EEPROM location.

Example

```
>BL D 0 ↵       Fill all data EEPROM with 00s
>BL D 1 ↵       Fill all data EEPROM with FFs
```

CA — *Clear All Breakpoints*

Syntax

CA

Description

The CA command clears all software breakpoints that are set.

CM — *Compare Memory*

Syntax

CM *start1 stop start2*

Description

The CM command verifies whether two blocks in memory are identical; if they aren't, it lists the differences between the two blocks. The CM command has three parameters:

- The *start1* parameter defines the starting address of the first range that you want to compare.
- The *stop* parameter is ending address of the first range that you want to compare.
- The *start2* parameter defines the starting address of the second range that you want to compare. The ending address of the second range is calculated by adding the difference between the starting and ending addresses for the first range to the starting address of the second range.

If there are differences between the two ranges, the first value listed in the output comes from the first range, and the second value listed in the output comes from the second range.

Example

Compare the contents of the range 7000h–7004h to the contents of the range 7005h–7009h, given the following conditions:

7000h = 00 01 02 03 04 00 02 02 03 05

```
>CM 7000 7004 7005 ↵
7001 01 02
7004 04 05
```

DB — *Display Breakpoints*

Syntax

DB

Description

The DB command displays a list of memory addresses at which breakpoints are set in your program.

Example

The following example shows that three breakpoints are set:

```
>DB ↵
7000 7432 7593
```

DH — *Download Hex*

Syntax

DH *data ...*

Description

The DH command transfers data stored in the Intel Intellec 8/MDS object format from the host computer RS232 port to the slave's memory. The format for Intel Hex is shown in the UH (upload hex) command description on page D-30. The destination of the program memory data is external RAM if the external-memory (>) prompt is displayed or internal program memory if the internal-memory ( ]) prompt is displayed.

You will see a "Bad number on line xxxx" error message if the board receives a nonhex character on line xxxx. Line 0001 refers to the first object line received. The rest of the line is ignored after this error occurs, and the board continues from the start of the next line. You should correct the error before trying to execute the downloaded code.

The '370 application board supports only XON–XOFF protocol. The RS232 hardware handshaking signals are not currently used by the monitor. If the host cannot use XON–XOFF protocol, then the application board may overrun the host. To preclude this, either lower the baud or activate the XON–XOFF feature.

When you use the DH command program EEPROM, the data transfer rate is very slow. The data EEPROM can program as slowly as 50 bytes per second. Use the block program command (BL) to set or clear the entire EEPROM array before downloading. This will speed up the programming time to about 100 bytes per second. You **must** use the XON–XOFF protocol at this slow speed.

Another alternative to downloading EEPROM or EPROM is to allow the host computer to download to the external RAM. Then you can use the XM command to program the EEPROM or EPROM from the contents of RAM.

Some terminal emulator programs have trouble accepting error messages while downloading. The last action of the DH command is to send either "No errors found" or "xxxx errors found". If these messages are not received, the board probably encountered a fatal error and ignored further data. The most common causes for receiving no error status messages are that the code is not in hex format, or the host program is not sending data. For more information about translating code to hex format, refer to page B-2.

Example

```
>DH ↵    <activate terminal emulator download utility>
No errors found
Done
>
```

DM — *Dump Memory*

Syntax

DM [*start* [*stop*]]

Description

The DM command dumps a block of data in hex and ASCII format that is defined by the *start* address and the *stop* address.

- If you omit the *stop* address, the command monitor displays 64 bytes from the *start* address (or four lines).
- If you omit both addresses, the command monitor displays memory locations 0–3Fh.

Memory is displayed 16 bytes on a line followed by the ASCII equivalent of the bytes. If a byte is not a printable ASCII character, a period is printed instead.

- To stop or start the display at any time, press any key.
- To abort the command, press ESC.
- To display one line at a time, press SPACE.

For a continuous memory dump beginning at a specified start address, append a minus sign (–) to the start address and omit the stop address.

Example

```
>DM  7000 7050 ↵
7000  00 1E 01 41 70 70 20 42  6F 61 72 64 06 12 07 10  ...App Board....
7010  08 0E 09 0C 0A 0A 0B 08  0C 06 0D 04 0E 02 0F 00  ................
7020  10 11 12 AA 13 AA 14 AA  15 AA 16 AA 13 17 AA 10  ................
7030  18 AA 19 AA 1A AA 1B AA  1C AA 1D AA 1E AA 1F AA  ................
7040  20 21 55 22 3C 23 3C 24  3C 25 3C 26 3C 13 27 3C   !U"<#<$<%<&<.'<
7050  10
>
```

DR — *Dump Registers*

Syntax

DR

Description

The DR command prints the stack pointer, status register, program counter, register A, register B, and the reverse assembly of the instruction at the program counter address. The command also displays the trace registers.

Example

```
>DR ↵
A=01  B=03  SP=60  CNZV12xx=01010000  PC=7123  ADD  #45h,R03
R10=23 P41=34 P40=12
```

In this example, R10, P41, and P40 are user-specified trace registers (refer to the MT command, page D-20, for more information about selecting trace registers).

FM *Fill Memory*

Syntax

FM *start stop value*

Description

The FM command fills a block of memory from the *start* address to the *stop* address with the value you specify with *value*. The fill value can be 0h to FFh.

The FM command quickly initializes a block of memory to a specific value. The command also verifies after each write to ensure data integrity. When it detects a data error, it displays an error message, and the command is aborted at that address.

This command can fill the external RAM at an approximate rate of 1K bytes per second. The EEPROM can be filled at a maximum rate of 100 bytes per second; a typical rate is 60 bytes per second. It will take up to 120 seconds to completely fill a 4K-byte program EPROM section.

HE *Display Help Screen*

Syntax

HE

Description

While the board is in stand-alone operation, you can use the HE command to display the help screen. This screen contains a list of most of the command-monitor commands, followed by descriptions for each command. You can exit from the help screen and return to the command line by pressing ESC.

Example

Figure D–2 shows the help screen.

Figure D–2.Stand-Alone Help Screen

```
--------------------------------------------------------------------------------
 TMS370 FAMILY APPLICATION BOARD – Ver 1.50 (C) 1988–1993 Texas Instruments
--------------------------------------------------------------------------------
SL XXX  – Select processor type xxx/0   DR – Display Registers
IN      – Initialize processor          DB – Display breakpoints
RR      – Reset and run                 WR – Wait for reset and run
UH start stop – Upload Hex format       HE – Help: print this list
DH data – Download Hex data             VH data – Verify Hex with memory
MR      – Modify PC,ST,SP,A and B       RU aaaa – Run at address aaaa
MM addr              –Modify memory at addr (SP,CR,–,.,Q,ESC)
MB                   –Modify breakpoints (5 max) 0=delete (CA clears all)
MT                   –Modify Trace registers; Rxx or Pxx; 0=delete
SS xxxx              –Single step xxxx instructions– cr=1 (Rxxxx=repeat)
RA start xxxx        –Reverse assemble xxxx instructions from start
AN a b               –Read Analog channel a with reference b
BL D/P 0/1           –Block program Data/Program with 0/1
CM start stop dest   –Compare memory start to stop against dest
DM start stop        –Display memory from start to stop
FM start stop cc     –Fill memory from start to stop with cc
LM start stop a b    –Locate bytes (6 max) from start to stop for a b...f
TM start stop dest   –Transfer memory from start to stop, to dest
XM start stop        –Shift between external and Internal Program memory
DEVICE IS 370C050
>
--------------------------------------------------------------------------------
```

IN — *Initialize Processor*

Syntax

IN

Description

The IN command initializes the slave that was previously selected by the SL command or by power-up default. Initialization activates the $\overline{\text{RESET}}$ pin and then attempts to load the slave control program (shown in Figure D–1, page D-5) into the external RAM of the ’370Cx5x. The control program must be resident for proper slave operation of the SS, RU, RR, and WR commands. The last step of the select device command (SL) is to invoke the IN command.

The IN command is also executed upon application board power-up, reset, and activation of the application board reset switch.

Note:

Use this command after a “Timeout, no EDS from slave” error. If this does not correct the error condition, then refer to page C-9.

LM — *Locate Memory*

Syntax

LM *start stop a* [*b*] [*c*] [*d*] [*e*] [*f*]

Description

The LM command locates specific patterns in memory and prints the starting location of each pattern found.

- The *start* parameter specifies the starting address of the block to be searched.
- The *stop* address specifies the ending address of the block to be searched.
- The parameters *a* through *f* represent data values or a sequence of data values to be searched for. You can specify a maximum of six bytes.

The data sequence of 1 to 6 bytes must reside within the inclusive start and stop address range boundaries.

Example 1

Assume current memory at 7000h is:

```
7000= 00 01 02 03 04 00 01 02
7008= 03 04 00 01 02 03 04 00
7010= 00
```

```
>LM 7000 700D 1 2 3 4 ↵
7001
7006
DONE
>
```

In this example, the data pattern starting at 700Bh and ending at 700Eh was not found, because the end address 700Dh stopped the search. You could change the LM command parameters to find the last data pattern as shown in this example:

Example 2

```
>LM 7000 700E 1 2 3 4 ↵
7001
7006
700B
DONE
>
```

MB — *Modify Breakpoint*

Syntax

MB

Description

The MB command allows you to set, modify, or clear breakpoints in memory before running a program. You can set up to five software breakpoints in memory. The following limitations apply to the breakpoint addresses:

- ❑ The address must be specified on opcode boundaries.
- ❑ The address must reside in alterable memory such as RAM or EEPROM.
- ❑ The address cannot be used more than once within the same MB command; otherwise, an error will result.

See the discussion on software breakpoints in Section D.6, page D-6, to understand the functions and limitations of breakpoints.

Once a breakpoint address is displayed, you can modify it by entering a valid hex address or a zero. Entering 0 (zero) deletes the address from the breakpoint list. Entering CA as an address clears all of the breakpoints from the current address through the end of the breakpoint list.

After you have modified the value, you can do one of the following:

To do this...	Press
Accept the new value and display the next breakpoint address	↵ or SPACE
Accept the new value and display the previous breakpoint address	–
Accept the new value and redisplay the same breakpoint address	.
Accept the new value and exit the command	Q
Ignore the new value and exit the command	ESC

MM — *Modify Memory*

Syntax

MM *address*

Description

The MM command examines and/or modifies bytes in memory. The *address* parameter identifies the memory address to be modified.

MM is the primary command for making changes to the slave memory. This command can alter internal RAM, peripherals, EEPROM, on-chip EPROM, and external RAM. The command verifies any changes and beeps if the data are not the same as those that were written. You'll hear beeps most commonly when you are modifying values in the peripheral file, where the values are read-only and clear-/set-only bits.

Once the address contents are displayed, you can modify contents by entering valid hexadecimal digits. Only the last two hex digits entered per line are used as data bytes.

After you have modified the value, you can do one of the following:

To do this...	Press
Accept the new value and display the next value	↵ or SPACE
Accept the new value and display the previous value	– (dash)
Accept the new value and redisplay the same value	. (period)
Accept the new value and exit the command	Q
Ignore the new value and exit the command	ESC

The command monitor can select the program memory location (on-chip or off-chip) for each slave device type selected. The select device command (SL) selects the slave device type and the internal or external program memory location. The program memory location is indicated by a monitor prompt: the > prompt indicates that the command accesses external off-chip program memory, and the] prompt indicates that the command accesses internal on-chip program memory.

MR — *Modify Registers*

Syntax

MR

Description

The MR command examines and/or modifies slave system registers such as the PC, ST, and SP registers, as well as registers A and B. Once the address contents are displayed, you can modify contents by entering valid hexadecimal digits. Only the last two hex digits entered per line are used as data bytes. For the PC register only, the last four digits are accepted.

After you have modified the value, you can do one of the following:

To do this...	Press
Accept the new value and display the next value	↵ or SPACE
Accept the new value and display the previous value	– (dash)
Accept the new value and redisplay the same value	. (period)
Accept the new value and exit the command	Q
Ignore the new value and exit the command	ESC

Example

For this example, assume that PC = 7123h, register A = 44h, register B = 55h, SP = 60h, and ST = 20h.

```
>MR ↵
PC = 7123 7321 SPACE     Display PC, change to 7321h
A  = 44 33 SPACE         Display A , change data to 33h
B  = 55 SPACE            Display B , advance without change
SP = 60 SPACE            Display SP, advance without change
ST = 20 –                Display ST, go back to previous location
SP = 60 66 Q             Display SP, change to 66 then exit.
>
```

MT — *Modify Trace*

Syntax

MT

Description

The MT command selects the trace registers that will be displayed after each DR, RU, RR, WR, or SS command. You can choose any register or peripheral file to become one of eight different trace registers.

You can modify the displayed trace register:

- ❑ To specify a register, use this format: Rxx, where xx is any valid hexadecimal digit from 01h to FFh.
- ❑ To specify a peripheral register, use this format: Pxx.
- ❑ To delete a trace register from the list, enter 0 (zero), R00, or P00.
- ❑ To clear all trace registers from the current point to the end of the trace list, enter C.

After you have modified the value, you can do one of the following:

To do this...	Press
Accept the new value and display the next trace parameter	↵ or SPACE
Accept the new value and display the previous trace parameter	– (dash)
Accept the new value and redisplay the same trace parameter	. (period)
Accept the new value and exit the command	Q
Ignore the new value and exit the command	ESC

Example

For this example, assume that trace 1 = R10, 2 = R11, 3 = P12, 4 = R15, and 5 = RA0.

```
>MT ↵
1 = R10 P11 SPACE        display #1, change to P11 (1011h)
2 = R11 R4  SPACE        display #2, change to R04
3 = P12 0                display #3, delete #3 from trace list
4 = R15 SPACE            display #4, advance without change
5 = RA0 –                display #5, go back to previous location
4 = R15 Q                display #4, exit command
>
```

RA — *Reverse Assemble Memory*

Syntax

RA *start count*

Description

The RA command looks at your program mnemonics in memory and provides valuable information about what is contained in memory.

- The *start* address must be at the starting address of an instruction in your program.
- The *count* is the number of instructions to reverse assemble. If you want a continuous display, enter 0 or ↵ for the *count* parameter.

The display begins at the start address that you entered.

- To halt or continue the command, press any key.
- To display one line at a time, press SPACE.
- To exit the RA command, press ESC.

Several areas of information are displayed on each line of the output. The decimal cycle count is useful for calculating time. (This is a rare case in which a decimal number is used in the monitor program.)

In calculating instruction times, you must take into account wait states. When you use the automatic wait-state feature, the instruction cycle count increases for accesses to external memory devices. Note that the port replacement circuits and external (off-chip) program RAM are affected by the auto-wait-state feature. This effect occurs even though the memory addresses are the same as those of the on-chip functions. Refer to the *TMS370 Family Data Manual* for more information about generating automatic wait states.

Example

This is an example of the RA command output, as well as an explanation of the parts of the display.

```
10  (2)   76553303    7000    BTJO    #55h,R033,7007
 |   |     |           |       |        |   |    |
 1   2     3           4       5        6   7    8
```

1) 10 — Cycles for number of wait-state instructions and operands (in decimal)

2) 2 — If branch is taken, add two cycles

3) 7655.. — Actual data to give this instruction

4) 7000 — Beginning address of this instruction

5) BTJO — Opcode

6) #55h Immediate operand, in hex

7) R033 Register, in hex

8) 7007 All relative addresses are calculated to give destination addresses

RR *Reset and Run*

Syntax **RR**

Description The RR command executes instructions that begin whenever reset occurs. The command resets the slave and begins running at the location contained at reset vector address (7FFEh,7FFFh). The program continues to run until a software breakpoint is reached, any key is pressed, the board reset switch is pressed, or protoarea reset is activated (if enabled). This command executes a dump register (DR) command when a software breakpoint is reached or when you press any key.

A correct reset vector must be in memory in order for RR to work properly. See the run command (RU) for more information on using execution instructions.

Example

```
>RR ↵
Running after reset              ;breakpoint or key pressed
Stopped                          ;current status from DR command

A=01  B=03  SP=60  CNZV21xx= 00000000  PC=7234  ADD  #45h,R03
```

RU — *Run*

Syntax

RU [*start*]

Description

The RUN command executes instructions starting from the *start* address that you specify. If you omit the address, execution begins at the current PC. The program continues to run until a software breakpoint is reached, any key is pressed, the board reset switch is pressed, or protoarea reset is activated (if enabled). This command executes a dump register (DR) command when a software breakpoint is reached or when you press any key. The instruction shown in the register dump is the next instruction to be executed.

There are certain limitations for the run commands (RU, RR, and WR).

- The program start address can be any valid slave memory address except 2xxxh, which is reserved for master-to-slave communications.
- Trying to stop the program while executing from internal on-chip memory will probably result in an error from the master:

  `Timeout, NO xxx from slave`

  (where xxx = $\overline{\text{OCF}}$, $\overline{\text{EDS}}$, or $\overline{\text{WAIT}}$)

  This error indicates that the master controller could not stop the slave, and all synchronization was lost. You must reinitialize by executing the monitor command IN, which resynchronizes the master and slaves.
- The run commands can cleanly stop only those slaves that have a full address/data bus. Currently, the 68-pin devices are the only devices with a full address/data bus. Attempts to stop a 28-pin slave device after the execution of a RU, RR, or WR command will cause an error, as described above. Again, you must execute the IN command.
- The port recreation pin functions only when internal program memory is disabled (P011.2=1). Use internal memory mode only to program on-chip memory.

See Section D.5, page D-5, for more information on running.

Example 1

```
>RU 7123 ↵        ;set PC = 7123h and the run
Running           ;breakpoint or key pressed
Stopped           ;current status from DR command

A=34  B=03  SP=60  CNZV21xx= 00000000  PC=7234  ADD  #45h,R03
```

Example 2

```
>RU ↵           ;run from current PC
Running         ;breakpoint or key pressed
Stopped         ;current status from DR command
A=34  B=03  SP=60  CNZV21xx= 00000000  PC=7234  ADD  #45h,R03
```

SL *Select Device*

Syntax **SL** [*device*]

Description The SL command configures the ’370 application board for a particular slave device and initializes the selected device. You must have an appropriate physical device installed in the correct socket for this command to function.

The *device* that you enter is the last three digits of the device name. Table D–1 lists all of the available devices.

Table D–1. TMS370 Family Slave Devices

Device Name	Register Size (bytes)	Data EEPROM Size (bytes)	Program Memory Size (bytes)	Program Memory Type	Socket
TMS370C010	128	256	4K	ROM	28
TMS370C310	128	128	4K	ROM	28
TMS370C710	128	256	4K	EPROM	28
TMS370C050	256	256	4K	ROM	68
TMS370C150	256	—	—	—	68
TMS370C250	256	256	—	—	68
TMS370C350	256	—	4K	ROM	68
TMS370C052	256	256	8K	ROM	68
TMS370C352	256	256	8K	ROM	68
TMS370C056	512	512	16K	ROM	68
TMS370C156	512	—	—	—	68
TMS370C256	512	512	—	—	68
TMS370C356	512	—	16K	ROM	68
TMS370C756	512	512	16K	EPROM	68

- If you omit the *device*, it defaults to the last device that you entered.
- To list all of the available devices, enter 0 in place of the *device.*

If the device that you selected has alterable program memory, you are prompted with this question:

```
Program memory External/Internal (E/I) ?
```

- Enter I if you want all memory commands in the program memory range to apply to the internal slave memory.
- Enter E if you want all memory commands in the program memory range to apply to the external slave memory (normally a static RAM).

If the board cannot initialize the slave device, an error message is issued. Refer to Appendix C for information about resolving these errors.

Example 1

```
>SL 756 ↵
DEVICE IS 370C756
Program memory External/Internal (E/I) ? I
]
```

Example 2

```
]SL 756 ↵
DEVICE IS 370C756
Program memory External/Internal (E/I) ? E
>
```

SS *Single Step*

Syntax **SS** [*count*]

Description The SS command causes the slave to execute one or more instructions at a time and gives you current status information. The *count* parameter allows you to specify the number of instructions you want to execute. The *count* can be any value from 0h to FFFFh; if you omit the count, one command is executed.

SS executes instructions by using current register information. After each instruction, the following information is displayed: the contents of the PC, ST, SP, register A, register B, the selected trace registers, and the next instruction to be executed.

After the display has stopped, you can do the following:

- To repeat the SS command a specified number of times, enter Rxx, where xx is a number from 0h to FFFFh. If the original command specified 16 instructions at a time (SS 10), then R4 executes 4 times 16 (or 64 total instructions) while giving a register display at the end of every 16 instructions.
- To single-step through the same number of commands that you just stepped through, press SPACE.

- To exit the SS command, press ESC.
- To step until the process stops, enter R or R0 to step 64K times (or until you cause the process to stop).

Note that an asterisk is displayed at the start of every repeat group.

If a breakpoint is encountered, the single-stepping stops. The breakpoint stop is indicated by the letter B at the beginning of the line.

When single-stepping without displaying after each step (this occurs when you step more than one instruction at a time), the application board executes about 7,700 instructions per second. When single-stepping while displaying, the rate depends upon the baud and the repetition rate.

A limitation of the SS command is that you can single-step only those instructions in which the slave processor gives an $\overline{\text{EDS}}$ signal. As a result, no instruction executions can occur out of internal memory, which includes RAM and data EEPROM. If you attempt to execute an instruction out of internal memory, you will see a timeout error—"Timeout, no xxx from slave"—where xxx is $\overline{\text{OCF}}$, $\overline{\text{EDS}}$, or $\overline{\text{WAIT}}$. This error indicates that the master has waited too long for an expected signal from the slave.

When you single-step through a section of code, it is sometimes helpful to disable the global interrupts. This prevents constant timer, external, or other interrupts from delaying the routines. Disable the interrupts by setting the status register (ST) to 0 before single-stepping.

Example

Assume that the code to step through is the following:

```
7000  ADD   #01h,R000        ;Single-step example code
7003  ADD   #02h,R001
7006  JMP   7000
```

```
>SS ↵
* A=01   B=00  SP=60  CNZV21xx= 00000000  PC=7003  ADD    #02h,R001
R03=FF R04=00     SPACE
* A=01   B=02  SP=60  CNZV21xx= 00000000  PC=7006  JMP    7000
R03=FF R04=00     SPACE
* A=01   B=02  SP=60  CNZV21xx= 00000000  PC=7000  ADD    #01h,R000
R03=FF R04=00     SPACE
* A=02   B=02  SP=60  CNZV21xx= 00000000  PC=7003  ADD    #02h,R001
R03=FF R04=00     SPACE
* A=02   B=04  SP=60  CNZV21xx= 00000000  PC=7006  JMP    7000
R03=FF R04=00     ESC
>
```

```
>SS  10 ↵          -- step 10h instructions at a time
* A=06   B=0A  SP=60  CNZV21xx= 00000000  PC=7003  ADD    #02h,R001
R2 ↵               -- repeat the 10h block, two times
* A=0B   B=16  SP=60  CNZV21xx= 00000000  PC=7006  JMP    7000
A=10   B=20  SP=60  CNZV21xx= 00000000  PC=7000  ADD    #01h,R000
SPACE              -- again
* A=16   B=2A  SP=60  CNZV21xx= 00000000  PC=7003  ADD    #02h,R001
A=1B   B=36  SP=60  CNZV21xx= 00000000  PC=7006  JMP    7000
ESC
>
```

T0 *Test Command*

Syntax **T0**[**0**|**1**|**2**|**3**|**4**]

Description The T0 command provides a quick and simple test of the application board master. This test also helps you to set various switches, jumpers, and potentiometers. If you are able to run this command successfully, your board does not have any major failures. The command can run all tests, or it can run a test that you specify on the command line. The command line number corresponds to the list in Table D–2. If you input nothing or 0, the command runs all tests.

Table D–2. Test Command Parameters for T0 Command

Test Number	Test
1	MC voltage adjust
2	Switch 2 settings
3	Master EPROM checksum
4	Jumper positions

These tests do not constitute a full and complete test of the board, but they will detect numerous problems. If an area is found to be bad or you input a test number, then the command lets you repeat the test, loop continuously on this test, or skip the test (the command prompts you for an input):

```
S = skip test   CR or SP = next value   ESC = exit
```

See Appendix C for causes and corrections of errors.

Example This is an example of a successful complete test.

```
>T0 ↵
Remove JP3, now          SPACE
Adjust R3 to give 12.5
12.5                     SPACE   -- Adjust R3 until value is 12.5
Switch SW2 1234 = 0000           -- display changes if switch is changed
Switch SW2 1234 = 1000             or space key is pressed
Switch SW2 1234 = 1100
Switch SW2 1234 = 1110
Switch SW2 1234 = 1111
Switch SW2 1234 = 1110
Switch SW2 1234 = 1100
Switch SW2 1234 = 1000
Switch SW2 1234 = 0000     ↵     -- exit with switch reading 0000
Master EPROM checksum = 3C93 OK -- checksum depends on version number
Default jumper positions
JP3= uPROC/WPO
JP4= on
JP2= 27C128     JP1=RAM
SW2= all off
Finished tests, status = OK
>
```

T1 *Test Command*

Syntax **T1**[**0**|**1**|**2**|**3**|**4**|**5**|**6**]

Description The T1 command provides a quick and simple test of the application board slave areas. If you are able to run this command successfully, your board does not have any major failures. The command can run all tests or it can run a test that you specify on the command line. The command line number corresponds to the list in Table D–3. If you input nothing or 0, the command runs all tests.

Table D–3. Test Command Parameters for T1 Command

Test Number	Test
1	Ports
2	High voltage
3	Data EEPROM
4	Program EEPROM
5	Address bus
6	Memory bus

These tests do not constitute a full and complete test of the board, but they will detect numerous problems. If an area is found to be bad or you input a test number, then the command lets you repeat the test, loop continuously on this test, or skip the test (the command prompts you for an input):

```
S = skip test    CR or SP = next value    ESC = exit
```

See Appendix C for causes and corrections of errors.

Before you run this test, select the proper device.

Example This is an example of a successful complete test.

```
>T1 ↵
Ports OK
High Voltage Test – OK
Data EEPROM  WILL BE CLEARED,  S to skip test, ESC to exit
Data EEPROM OK
Program EEPROM not available on selected device
Address  OK
Memory ............... OK
Finished tests, status = OK
>
```

TM — *Transfer Memory*

Syntax **TM** *start stop destination*

Description The TM command moves a block of memory defined by the *start* and *stop* addresses to the memory starting at the *destination* address. The source is not changed, except where the locations overlap.

UH — *Upload Hex*

Syntax **UH** *start stop*

Description The UH command transfers a block of memory from the slave's memory to the host computer via the standard Intel Hex format. The block to be transferred is defined by the *start* address and the *stop* address.

Commonly, a screen save feature of the terminal emulator program is turned on after you type the last character but before you press ↵. It is turned off when the command finishes. If the host cannot use XON–XOFF protocol, then the application board may overrun the host. Either lower the baud or activate the XON–XOFF feature.

Example The format for Intel Hex is shown in this example.

```
>UH  7000 702F
:207000000F0E0D0C0B0A090807060504030201001F1E1D1C1B1A19181716151513121107F
:107020002F2E2D2C2B2A292726252524232222120EB                                | |
:007030015F  |                                                              | |
|| |   |     |                                                              | |
12 3   4     5                                                             6 7
```

1) The header character is always a colon. Line starts downloading when colon symbol is received.

2) Hex number of data bytes in line (two characters)

3) Address of first data byte (four characters)

4) 00= data; 01=terminate

5) Data bytes

6) Checksum: twos complement of the summation of the preceding bytes in the line

7) Any character before the next colon

VH — *Verify Hex*

Syntax

VH *data ...*

Description

The VH command verifies a block of host memory against slave memory by using data on the standard Intel Hex format. Usually, a file transfer feature of the terminal emulator program is turned on after the last character is typed.

If a discrepancy is found, the command prints the address, the file data, and the memory data in the following format:

```
File error at 7123; wrote/read: 01100110 00110011
```

- The address is the first number.
- The data read from the host file is the second number.
- The data read from the slave is the last number.

Some terminal emulators will not print this error message, because they cannot download and accept text at the same time.

You may see a “Bad Number on line xxxx” error if the board receives a nonhex character when it expects a hex character. The rest of the line is ignored after this error, and the board continues from the start of the next line. You should correct the error before relying on the downloaded code.

If the host cannot use XON–XOFF protocol, then the application board may overrun the host. Either lower the baud or activate the XON–XOFF feature. The format for Intel Hex is shown in the UH (upload hex) command description.

WR — *Wait for Reset and Run*

Syntax

WR

Description

The WR command forces the slave to wait in an infinite loop until an external reset occurs. After reset, the slave begins running at the location contained at the reset vector location (7FFEh,7FFFh).

The program continues to run until (1) a breakpoint is reached, (2) you press any key, or (3) a reset is activated. The WR command does a register dump after the first two conditions.

Turning SW2.3 on allows you to select external resets; also, circuitry at the protoarea connector must activate the correct slave’s reset. The connector for the ’370Cx5x slave is P3 pin 48; for the ’370Cx1x slave, it is P5 pin 49.

This command will not work properly without an external reset. Also, there must be a correct reset vector in memory. See the RU (run) command description and Section D.5 for more information on using execution instructions.

Example

```
>WR ↵
running                                          ;breakpoint or key pressed
stopped
A=01 B=03 ST=20 SP=60 PC=7234 ADD #45h,R03   ;current status
```

XM *Exchange Memory*

Syntax **XM** *start stop*

Description The XM command moves a block of memory either from external slave memory to internal memory or from internal memory to external RAM. The block of memory is defined by the *start* and *stop* addresses that are within the program memory range.

The command transfers to **alterable** external memory. The application board cannot program EEPROM or EPROM in the external slave memory location, but it can program a ’370 microprocessor that contains EEPROM or EPROM.

After you enter the command, you are prompted to specify the direction of the transfer:

```
0 = Move external RAM to internal memory; 1 = Move internal RAM. 0/1?
```

Normal transfer time is about 7 seconds for 4K bytes of code with no EPROM programming and about 120 seconds with EPROM programming. Shifting memory from external to internal EPROM is a slower process, so a counter is installed to provide surveillance.

Appendix A

Jumper and Switch Settings

This appendix describes the location and function of each of the jumpers and switches on the TMS370 application board.

Topic **Page**

A.1 Overview of the Jumpers and Switches

The jumpers and switches for the ’370 application board are shown in Figure A–1.

Figure A–1. ’370 Application Board

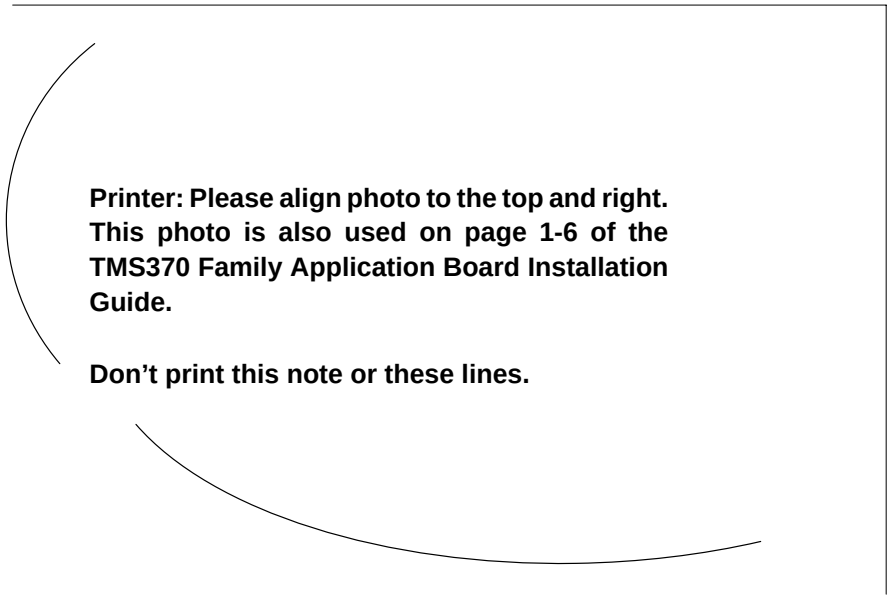


Table A–1 describes the default settings each of the jumpers and switches.

Table A–1. Default Switch and Jumper Settings

Board Designator	Description	Setting When Shipped
SW1	Resets the application board	open
SW2 (four switches)	Set the operation mode of the master	all four switches off
JP1	Sets the socket memory to RAM or EPROM	RAM
JP2	Sets the EPROM type for the slave memory	27C64/27C128
JP3	Sets the operating mode of the slave device	μPROC/WPO
JP4	Connects the slave's CLKIN pin to the master's CLKOUT signal	connected
JP5	Selects the EPROM type for the master monitor	27C128 (by PWB trace)

The sections that follow further describe the functions of the switches and jumpers.

A.2 Resetting the Application Board (SW1)

The ’370 application board reset switch is SW1. This momentary switch has a red handle and is normally in the open position (see Figure A–1).

SW1 resets the master on the application board only. The master, in turn, executes the following sequence after a reset:

1) Reads SW2.4 to determine which slave device to operate.

2) Applies reset to the ’370Cx1x and ’370Cx5x slaves.

3) Opens both analog switches to the protoarea reset pins.

4) Reads SW2.3 to determine if protoarea resets are allowed.

   If SW2.3 is ON:

   a) Closes the appropriate analog switch for the slave device selected by SW2.4. This connects the protoarea reset pin to the slave reset pin.

   b) Disables the master I/O line that drives the selected slave’s reset pin.

   If SW2.3 is OFF:

   Sets the master’s I/O line that drives the selected slave reset pin HIGH.

5) Loops blinking LED indefinitely.

The protoarea can drive the slave reset pin directly. Both resets for the ’370Cx1x and ’370Cx5x devices are brought out to the protoarea. These connections are made with analog switches on the application board. The master controls whether the switch is open or closed.

Note:

For proper operation, when using protoarea resets, external circuitry must be present on the protoarea connector. External circuitry can be as simple as a 2-kΩ resistor to V_{CC} or as complex as a voltage-sensing brownout trigger circuit.

The application board has been designed to use a power-up reset circuit in conjunction with SW1. For most applications, you will never need to use SW1 when you first apply power to the application board. When you see the blinking LED on the application board, the master is operating correctly.

A.3 Setting the Operating Mode of the Master (SW2)

When you reset the application board during power-up or when you activate the board reset switch, the master scans SW2, a four-switch DIP switch (see Figure A–2). The switches of SW2 select the operating mode of the ’370 application board (or master processor) and determine the function of the slaves.

Figure A–2. SW2 Switches

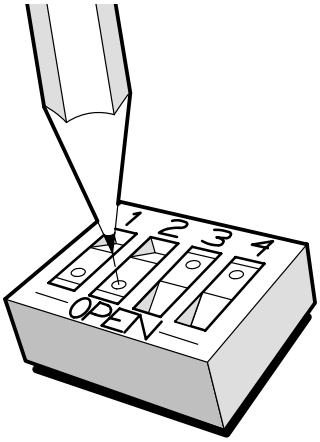


Table A–2 describes the operating modes of the master and lists the switch positions of SW2.

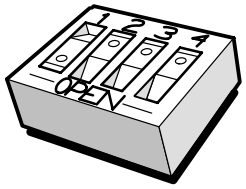
Table A–2. Switch Mode Selections for SW2

SW2.1	SW2.2	Operating Mode	Description
off	off	TTY mode	Standard terminal or terminal emulator program
on	off	Debugging mode	Full-screen PC mode
off	on	Isolated mode	Reset and run slave without control of the master
on	on	Future	Reserved for future use
SW2.3		**Function**	**Description**
off		Master reset	Slave reset from master only
on		Protoarea reset	Enable protoarea slave reset pin
SW2.4		**Function**	**Description**
off		’370Cx5x slave	Attach protoarea reset to ’370Cx5x slave (if enabled or in isolated mode)
on		’370Cx1x slave	Attach protoarea reset to ’370Cx1x slave (if enabled or in isolated mode)

Setting SW2.1 and SW2.2

SW2.1 and SW2.2 select the operating mode of the master:

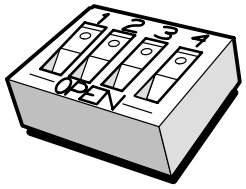
- **Debugging mode.** If you set SW2.1 to ON and SW2.2 to OFF, you enable the debugging mode.



Specifies
Debugging Mode

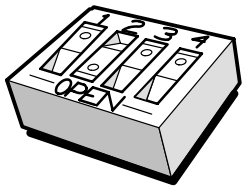
For more information about using the debugging mode, refer to the *TMS370 Family C Source Debugger Guide*.

- **TTY mode.** If you set both SW2.1 and SW2.2 to OFF, the master initializes the application board for use with a standard TTY terminal or with a host computer running a terminal emulator package (e.g., Kermit, CROSSTALK, PROCOMM).



Specifies
TTY Mode

- **Isolated mode.** If you set SW2.1 to OFF and SW2.2 to ON, you enable the isolated mode.



Specifies
Isolated Mode

Setting SW2.3

SW2.3 indicates what has control over the slave reset pin while the master is in isolated or TTY mode:

- If SW2.3 is OFF, only the master has control over the slave reset pin and sets the I/O levels.
- If SW2.3 is ON, the external circuitry in the prototyping area can control the slave reset pin during the isolated mode and TTY mode run commands (RR, RU, WR). *Slave reset circuits,* page 3-16, describes how to interface to the protoarea reset pins.

Setting SW2.4

SW2.4 selects the slave device type, ’370Cx1x or ’370Cx5x, for the isolated mode and for protoarea reset enables as described in Section A.2. When the master is in the TTY terminal mode, you can specify the individual slave types with the SL monitor command. Although you can operate either of the slave devices from the command monitor, SW2.3 and SW2.4 select the protoarea reset connections.

A.4 Setting the Socket Memory (JP1)

JP1 determines whether the slave memory resident in socket U26 is a static CMOS RAM or an EPROM. Figure A–3 illustrates how to select the slave memory.

Figure A–3. JP1 Selections

Selects RAM memory Selects EPROM memory

If you set the memory to RAM, you can download programs and set breakpoints in memory with the command monitor. With a read-only EPROM selected and installed, you can’t perform these operations. In addition, you must include the section of code listed below in the EPROM if you want to use the single-step and run capabilities from the command monitor.

```
.SECT    ”TIUSE”,7FDEh
.WORD    7FE0h
PUSH     ST
DINT
BR       2000h
JMP      $
RTI
```

A.5 Selecting the EPROM Type for the Slave Memory (JP2)

JP2 determines the type of EPROM placed in socket U26 when JP1 is set to EPROM. If you set JP1 to RAM, you don't need to set this jumper. You can select among the following EPROM devices for the slave device to use:

Device Type	Jumper Setting
27C64	27C64/27C128
27C128	27C64/27C128
27C256	27C256

Figure A–4 illustrates how to select the EPROM type.

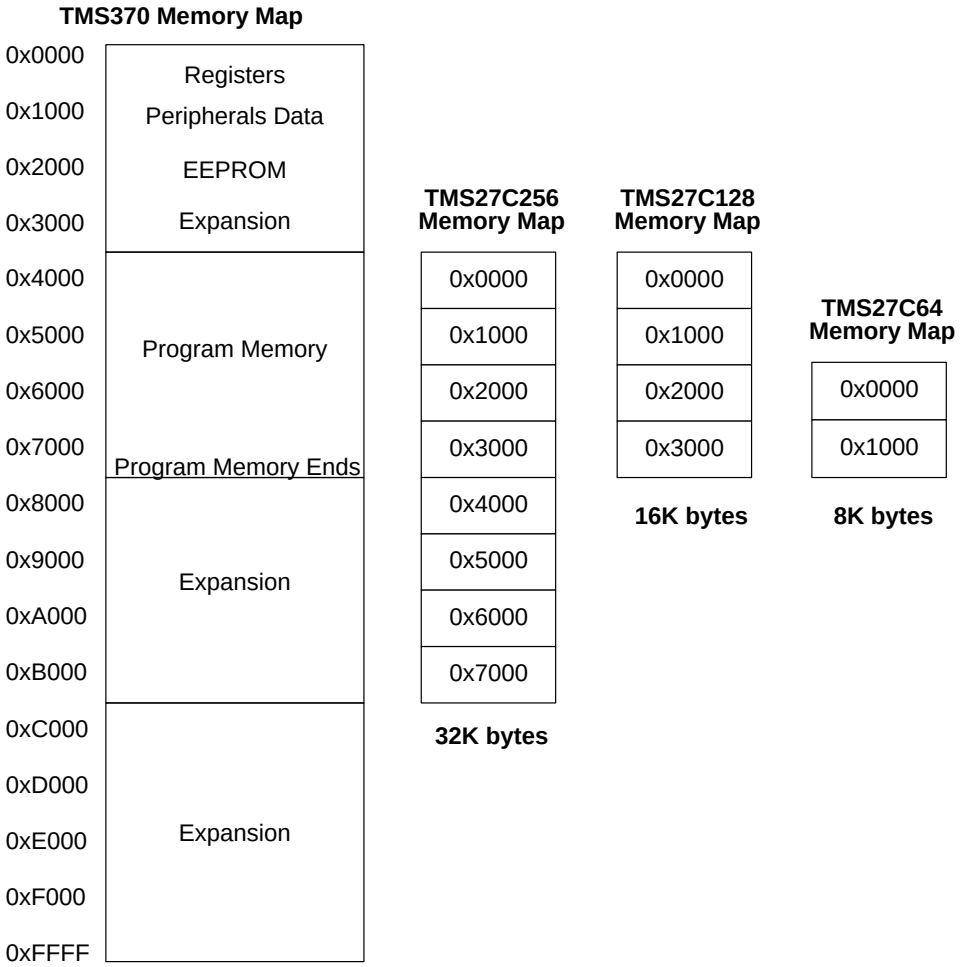
Figure A–4. JP2 Selections

Selects 2764 or 27C64; selects 27128 or 27C128

Selects 27256 or 27C256

Figure A–5 shows how memories of various sizes map into the slave's memory space.

Figure A–5. Slave Memory Map and External Memory

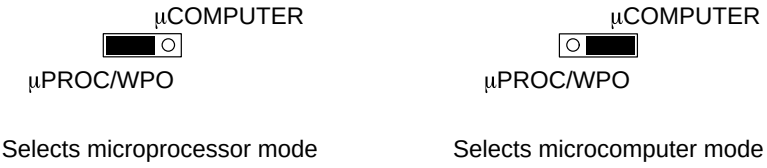


A.6 Setting the Operating Mode of the Slave Device (JP3)

JP3 determines the operating mode of the slave device: microprocessor with write-protect override (WPO) active, or microcomputer mode. These modes are enabled on the rising edge of slave reset.

Figure A–6 illustrates how to select the operating mode.

Figure A–6. JP3 Selections



Selects microprocessor mode

Selects microcomputer mode

You can use the microcomputer mode only in isolated mode. The setting forces the MC pin to ground and disables any communication between the master and the slave by disabling the complete address/data bus.

After reset, the slave device operates entirely from internal program memory in the microcomputer mode.

Note:

While using the isolated (microcomputer) mode, do not use the application board's I/O pins to select any of the memory devices on the board. To avoid this, maintain a logic high on the slave $\overline{\text{EDS}}$ I/O pin.

The microprocessor with WPO mode is used most often. This mode allows you to debug and inspect the operation of the slave by using an external CMOS RAM. This setting forces the MC pin to V_{CC} at reset and provides address bus, data bus, and control signals from the selected slave device.

When the command monitor writes to the on-chip EEPROM or EPROM memories of the slave device, a 12.5-volt write-protect override is supplied to the MC pin. This allows the master to modify all EEPROM memory locations, even if the write-protect bits are set.

The 12.5-volt WPO is supplied from the onboard switching power supply, which provides a range of 8 to 18 volts. You can use the command-monitor command T0 to adjust the voltage with the R3 potentiometer.

CAUTION

Always remove the J3 jumper when adjusting the potentiometer. You can damage the socketed slave devices with force greater than 13 volts.

A.7 Connecting the CLKIN Pin (JP4)

JP4 connects the master's CLKOUT signal to the 28-pin slave's CLKIN pin. As a result, the '370Cx1x slave operates with a 4.9152-MHz CLKIN signal from the master. Since the main purpose of the '370Cx1x socket is to program on-chip EEPROM, you do not need a high operating speed.

If you decide to evaluate a '370Cx1x operating in the isolated mode at a different speed, you must remove JP4 and install a parallel resonant crystal in the X2 sockets.

Note:

If you do not remove JP4 or install a crystal in the X2 socket, then the master-processor signal will overdrive the crystal without damaging the master.

A.8 Selecting the EPROM Type for the Master's Monitor (JP5)

JP5 selects the type of EPROM used for the master's command monitor. You can select either a 27C128 or a 27C256 EPROM. The jumper is set to 27C128 by a PWB trace on the application board. In order to select a 27C256 EPROM, you must cut the trace between the center and 27C128 pins and then install a jumper between the center and 27C256 pins.

Appendix B

Using the TMS370 Assembly Language Tools With the Application Board

Included with the TMS370 application board are the ’370 assembler and the ’370 code converter utility. This chapter briefly describes these tools and how they work with the application board. For a complete description of the assembler, code converter utility, and assembler directives, refer the *TMS370 Family Assembly Language Tools User’s Guide*, literature number SPNU010.

Topic **Page**

B.1 The Software Development Tools

The ’370 family software development tools for the application board include an assembler, a code converter, and a starter program.

The assembler

The assembler creates a machine-language program from a source file that you provide. The assembler contains assembly language instructions, directives, and macro directives. It substitutes absolute operation codes for symbolic operation codes and absolute addresses for symbolic addresses.

The code converter for TTY mode

The code converter converts COFF object files into Intel-formatted object files. The Intel format can be downloaded to the application board. Code conversion is most efficient if the assembler, converter, and terminal emulator are included in one batch file on a DOS machine (for an example, refer to Section B.3).

The command for using the code converter with the application board is:

rom370 –i *input file* [*output file*]

If you don’t specify an input file, the converter prompts you for it. The code converter assumes an .obj extension if you leave off the extension.

The output filename is optional; if you don’t specify a name, the converter uses the input filename with an .int extension.

These examples perform the same operation:

```
rom370 –i starter.obj starter.int ↵
rom370 –i starter ↵
```

Both of these commands convert the object code from the starter program, starter.obj, into an Intel hex file called starter.int.

The starter program

The starter program serves as a beginning point for programs included on the application board diskette and provides examples of section directives, trap vectors, interrupt vectors, and register definitions. It also includes the program required to use EPROM programs.

In order to write code effectively, you must use the starter file at the beginning of your program. You must write the Data.asm and Prog.asm sections of this program.

B.2 Using the Assembler Without the Linker

The *TMS370 Family Assembly Language Tools User's Guide* contains all of the information you need to use the assembler and the code converter. It also describes the format and assembler commands you need to write a program when you have access to a linker.

The application board package does not include a linker. This section contains additional notes that explain how to use the assembler without the linker program.

When creating programs for the application board, you should use the **absolute** options on the assembler directives:

Use Directives That Have *Absolute* Options...	**Instead of Directives That Have *Relocatable* Options...**
.TEXT, .DATA, .SECT, .REG, .REGPAIR	.BSS, .GLOBAL, .GLOBREG, .USECT, .DEF, .REF
These directives require an address that is permanently assigned to a memory location.	These directives require the linker to adjust all references to a symbol when the symbol's address changes.

All code is assigned to reside in one of three section types: .TEXT, .DATA, or .SECT.

The .TEXT and .DATA sections are pre-defined, and the .SECT directive allows you to name additional sections. The program can enter, exit, and change sections many times; each section occupies a contiguous space in the memory map after assembly. An absolute address is required on the first use of the section, but subsequent use of the section will begin where the same name section left off. See Figure B–1.

Figure B–1. Sections Directives

```
------------------------------------------------------------------------
ADDRESS  CODE                COMMENTS
------------------------------------------------------------------------
         .TEXT   7000H         ;pre–defined program section
7000       code   ...          ;program code
7100       code
         .DATA   7800H         ;pre–defined data section
7800       data    ...         ;data bytes
7900       data
         .SECT   "NAME1",7D00h ;named section  (1 or more)
7D00       code or data ...
         .TEXT                 ;begins where last text section left off
7101       more code ...       ;additional  program code
         .DATA                 ;begins where last data section left off
7901       more data ...       ;additional data code
         .SECT   "NAME2",7E00h ;additional named section
7E00       code or data
------------------------------------------------------------------------
```

The .REG and .REGPAIR directives always start at location 0. These directives are useful for assigning register names without concern for actual register addresses. The register section where all .REG(PAIR) names are assigned is located at 00–0FFh. The first two locations used by .REG(PAIR) are 00 and 01, which are called registers A and B and have special functions. The example below uses register assignment directives.

```
                ;first use of .REG(PAIR)
.REG     REGA   ;another name for register A
.REG     REGB   ;another name for register B
.REGPAIR UVW    ;user–defined register pair
.REG     XYZ    ;user–defined name
```

B.3 Example Batch Files

The following batch files illustrate the process of developing and assembling code that can be used with a terminator emulator program. The letters to the left of the statements correspond to the lettered notes following the examples.

Example 1:

```
a) MYEDITOR STARTER.ASM
b) ASM370   STARTER.ASM
c) IF ERRORLEVEL 1 GOTO ERR
d) ROM370-I STARTER.OBJ STARTER.INT <TTY mode only>
e) TRMEMU TMS370
   :ERR
```

Example 2:

```
   :ERR
   PAUSE
a) MYEDITOR STARTER.ASM
b) ASM370   STARTER.ASM
c) IF ERRORLEVEL 1 GOTO ERR
d) ROM370-I STARTER.OBJ STARTER.INT <TTY mode only>
f) EXIT
```

a) MYEDITOR represents the text editor that you use to make corrections in the assembly program. You can use any ASCII text editor.

b) The assembler assembles the program. Normally, the same source files are used during the development cycle.

   The name of the program (starter.asm) could be replaced by a %1 to allow a program name to be entered at the command line. See the batch command section your DOS manual.

c) If there are errors during assembly, the batch file exits (as in Example 1) or re-enters the editor (as in Example 2).

d) The application board running the stand-alone command monitor (TTY mode) accepts only Intel Hex object format for downloading code. Use the ’370 family code converter to translate the object code files.

e) TRMEMU is the terminal emulator program with a configuration file named TMS370. Terminal emulator programs such as Kermit are available as shareware on bulletin board systems. Other programs such as PROCOMM and CROSSTALK are commercially available.

f) In the second example, the terminal emulator program has the ability to terminate and stay resident in the computer. Meanwhile, you can re-edit and assemble the program by using the batch commands, then return to the terminal emulator task with an EXIT command.

Appendix C

Alphabetical Summary of Command-Monitor Messages

This appendix contains an alphabetical listing of the progress and error messages that might be displayed while you operate the command monitor. Each message contains both a description of the situation that caused the message and an action to take if the message indicates a problem or an error.

A

Address out of range

Description

- ❑ The second address that you entered was not greater than the first address.
- ❑ The address that you entered was not within the accepted range for the command (for example, the XM command won't execute if you enter an address outside the program memory).
- ❑ You attempted to modify or dump memory in the 2000h–2FFFh range.

Actions

- ❑ Re-enter the command with the second address greater than the first address.
- ❑ Ensure that the address you are entering is within the accepted range and is not within the 2000h to 2FFFh range.

B

Bad number on line xxx

Description When downloading hex with the DH command or verifying hex with the VH command, the board received a nonhex character on line xxx. Line 0001 refers to the first object line received. The rest of the line is ignored after this error occurs, and the board continues from the start of the next line.

Action Correct the error before trying to execute the downloaded code.

Block error at aaaa; wrote/read: xxxxxxxx yyyyyyyy

Description

- ❑ The wrong part was selected, or the selected part does not have EEPROM.
- ❑ 12 volts are not available at the MC pin.
- ❑ The EEPROM location is bad (especially if it is not the first array location).
- ❑ The EEPROM array is bad (especially if it is the first memory array location).

Actions

- ❑ Check to be sure you're selecting the correct part.
- ❑ Write to other locations to determine if there is a byte or array problem.
- ❑ Try an MM command in the array. If program memory is involved, make sure to use the] prompt, indicating internal memory control. Use the SL command to configure the prompt.
- ❑ Check jumper JP3; the jumper should be on pins 1 and 2 (left).
- ❑ Try a different part (the EEPROM may be bad).
- ❑ Check for 12 volts on pin 6 of U25 (a U4 or related component may be bad). Run test command T0 and check the voltage level at the MC pin of the slave.

C

Check address line Axx where xx= 00–12

Description The T1 <5> address bus test reads and writes to approximately 14 different locations in an attempt to find a floating, grounded, or stuck-high address line. This test works best when only one address line is bad. The test uses the 8K-byte RAM in the U26 memory socket and checks only its address lines A0–A12.

When this message is displayed, one of the following is wrong:

- ❑ The line is shorted, grounded, or floating.
- ❑ JP1 is not on RAM, or the EPROM in the external memory socket.

Actions

- ❑ Turn off the power to the board. Check for continuity between the ground, V_{CC}, and other address pins.
- ❑ Use command T1 <3> to loop on an address; use a scope to check the address line.

Checksum error at line #*xxxx*

Description The Intel Hex line did not add up to the correct checksum. There is an error on this line, but it does not affect other hex lines.

Causes

- ❑ There is an error in the file. Look on or around the given line number.
- ❑ The XON–XOFF protocol of the host is not working, and the command found the wrong values.
- ❑ The connection between the host and the application board was bad. Look for errors on the screen, or display memory for indications of a dirty signal.

Actions

- ❑ Remake the download file. Do not try to patch the error.
- ❑ Reconfigure the host for the XON–XOFF protocol.

Command not available in this mode

Description
- ❑ You tried to single-step, RR, or WR a ’370Cx1x device.
- ❑ You tried to set a breakpoint on a ’370Cx1x device.

Action Perform all memory development/debugging on a 68-pin device; then transfer the program to the on-chip memory of the 28-pin device.

D

Data EEPROM error at 1Fxx; wrote/read xxxxxxxx yyyyyyyy
Data EEPROM not available on selected device

Description The T1 <3> data EEPROM test performs an array-clear, checks every location, and then does an array-set and checks every location in data EEPROM.

When this message is displayed, one of the following is wrong:

- ❑ The improper device is selected.
- ❑ There is no voltage on the MC pin to change the protected memory.
- ❑ The cell is bad.
- ❑ The array is bad.
- ❑ JP3 is not in microprocessor mode.

Actions
- ❑ Select the appropriate device with the SL command (avoid the ’370C3xx parts).
- ❑ Run test T0 <1> to check the MC voltage.
- ❑ Test other locations with the MM command (is the cell or array bad?).
- ❑ Try another TMS370 device.

Data out of range

Description The data value is greater than allowable values; for example,

```
BK P 0 or 1 , AN 0–8
```

Action Use a valid data value. Refer to the individual command descriptions for a list of valid values.

E

(E)EPROM error at aaaa; wrote/read: xxxxxxxx yyyyyyyy

Causes

- ❑ The wrong part was selected, or the selected part does not have EEPROM or EPROM.
- ❑ 12 volts are not available at the MC pin.
- ❑ The EEPROM location is bad.
- ❑ The EEPROM array is bad.

Actions

- ❑ Check to be sure you're selecting the correct part.
- ❑ Check jumper JP3; the jumper should be on pins 1 and 2 (left).
- ❑ Write to other locations to determine if there is a byte or array problem.
- ❑ Try an appropriate BL command. (This will destroy data in the array.)
- ❑ Try a different part (EEPROM may be bad).
- ❑ Check for 12 volts on pin 6 of U25 (a U4 or related component may be bad). Run test command T0 and check for 12.5 volts on the MC line.

H

Hex number error at line #*xxxx*

Description A nonhex character was encountered while downloading (for example, a character not in the range of 0–9h or A–Fh).

Causes

- ❑ There is an error in the file. Look on or around the given line number.
- ❑ The XON–XOFF protocol of the host was not working, and the command found a colon while looking for a hex character.
- ❑ The connection between the host and the application board was bad.

Actions

- ❑ Remake the download file. Do not try to patch the error.
- ❑ Reconfigure the host for the XON–XOFF protocol.
- ❑ Look for errors on the screen, or display memory for indications of a dirty signal.

M

Memory error at xxxx; wrote/read: xxxxxxxx yyyyyyyy

Description The T1 <6> memory test writes and reads four values to every location in the 8K-byte RAM memory. If the write value does not agree with the read value, an error message is generated. This error can indicate any of the following:

- ❑ JP1 is not set on RAM, or the EPROM is in the external memory socket.
- ❑ The device is not seated properly in the socket.
- ❑ The memory socket is bad.
- ❑ The memory location is bad.
- ❑ The memory device is bad.

Actions

- ❑ If the bad memory location is at 6000h, then the write sequence, rather than a specific location, is bad.
- ❑ Replace the device.
- ❑ Check the slave data bus using the T1 <4> command.

P

PORT error address 102x; wrote/read: xxxxxxxx yyyyyyyy

Description This error can occur while you're performing a T1 <1> port test. The error indicates that the device in the port recreation logic is bad. Table C–1 shows devices associated with the address. See the schematics in Appendix D for the circuit.

Table C–1. Device Addresses

		Associated Devices		
Address	**Port**	**Read**	**Write**	**Direction**
1022	A	U16	U22	U8A
1026	B	U15	U21	U8B
102A	C	U14	U20	U6A
102E	D	U13	U19	U6B

Action Check the voltages of the protoarea pins associated with the port:

- ❑ If the value at the port does not agree with the value written, the problem is likely in the write devices.
- ❑ If the value agrees, then the error is probably in the read device.

Program EEPROM error at: xxxx; wrote/read xxxxx yyyyy
Program EEPROM not available on selected device

Description The T1 <4> program EEPROM test performs an array-clear operation, checks every location, and then does an array-set and checks every location in program EEPROM.

When this message is displayed, one of the following is wrong:

- ❑ The improper device was selected.
- ❑ There is no voltage on MC pin to change protected memory.
- ❑ The cell is bad.
- ❑ The array is bad.

Actions

- ❑ Select the correct device with the SL command (use the ’370C7xx parts).
- ❑ Run test T0 <1> to check the MC voltage.
- ❑ Test other locations with MM command (is the cell or array bad?).
- ❑ Try another TMS370 device.

S

SCCR0 read as 00000000

Description The T1 <2> HI voltage test looks at a bit on the slave that indicates that the processor is in the WPO mode. The actual MC voltage is adjusted in test T0 <1>.

Actions

- ❑ Make sure that JP3 is in the μPROC/WPO position (pins 1 and 2 or left).
- ❑ Measure the MC voltage with the T0 <1> command. See test T0 <1> if failure occurs.
- ❑ Measure the MC voltage at the SMC pin at the prototype area (4). Check the slave socket. The actual pin is low voltage, but socket contact might read okay.

T

T0 <1> voltage adjust test

Description

- ❑ The test fails to read the value or reads a value near 0 volts or near 20 volts.
- ❑ The value fails to change when you turn the R3 potentiometer.

Causes

- ❑ Jumper JP3 is not off, and the board is sinking too much current.
- ❑ The R3 potentiometer is bad.
- ❑ TLC497 circuit is bad.
- ❑ The board power supply is not at 5.0 volts.

Actions

- ❑ Check the value of the center pin of JP3 on a voltmeter.
- ❑ Check and replace the R3 potentiometer.
- ❑ Check the schematics in Appendix D for debugging.

T0 <2> SW2 switch test

Description The master fails to respond to a switch change.

Causes

- ❑ The switch is dirty.
- ❑ The switch is bad.
- ❑ The PWB trace is broken.
- ❑ The master socket is bad.
- ❑ The master is bad.

Actions

- ❑ Clean the switch with an electrical switch cleaner.
- ❑ Replace the switch.
- ❑ Test the trace for continuity and repair as necessary.
- ❑ Examine the master socket.
- ❑ Exchange the master and slave, then repeat the test.

T0 <3> master UVEPROM checksum test

Description If the application board ran the EPROM test, then it is not a gross EPROM failure—possibly just one or two bad bits. These bad bits can cause major program flaws under certain conditions.

Symptom Master shows a bad checksum.

Causes

- ❑ The socket is dirty or bad.
- ❑ The EPROM is bad.
- ❑ The device was left in the sun without the foil EPROM window cover.

Actions

- ❑ Clean or replace the socket.
- ❑ Replace the EPROM (the latest code should be available on TMS370 Bulletin Board).

Timeout, No WAIT from slave
Timeout, No EDS from slave
Timeout, No OCF from slave

Description The master controls the slave through the $\overline{\text{EDS}}$ and $\overline{\text{WAIT}}$ signals. If the signal is not at the correct level when the master needs it, this error is generated. Of the three timeout errors, the $\overline{\text{WAIT}}$ error is the most likely to occur.

If $\overline{\text{WAIT}}$ is correct, then the master checks for $\overline{\text{EDS}}$. If $\overline{\text{EDS}}$ is correct and if this is an opcode cycle, then the master checks for $\overline{\text{OCF}}$.

Causes

- ❑ The wrong device was selected, or there is no device at the selected location.
- ❑ The slave is attempting to single-step or stop running in internal memory.
- ❑ Master-slave communication is not synchronized.
- ❑ The jumpers and/or SW2 are in the wrong position. (Check JP1, JP2, JP3, JP4, and SW2.)

- ❑ The socket is bad (especially the slave $\overline{\text{EDS}}$ pin or master ANx).
- ❑ The slave device is bad.
- ❑ The master device is bad.
- ❑ There is a problem with the board.

Actions

- ❑ Execute the initialize command (IN) and retry the command.
- ❑ Use SW1 to reset the master; then press ⏎.
- ❑ Turn off the power to the board and try again.
- ❑ Execute the T2 command.
- ❑ Try a different slave device. Use the other type of processor, if available (’370Cx1x or ’370Cx5x).
- ❑ Check for an oscillator. For 68-pin devices, check the crystal or ceramic resonator.
- ❑ Try a different socket.
- ❑ Check the power to the board (5 V and ground).

Use the schematics in Appendix D for the next actions:

- ❑ Select the correct processor with the SL command and check for the slave CLOCKOUT on pin 58 of U25, or pin 28 of U12 or U27 (68-pin or 28-pin device).
- ❑ Check for the slave CLKIN on pin 31 of U25, or pin 5 of U12 or U27 (68-pin or 28-pin device).
- ❑ Check the $\overline{\text{RESET}}$ line—pin 53 of U25, or pin 27 of U12 or U27. The active device should have $\overline{\text{RESET}}$ high. The inactive device should have $\overline{\text{RESET}}$ low. The IN command should pulse the $\overline{\text{RESET}}$ line to the active device.
- ❑ Check the $\overline{\text{WAIT}}$ line—pin 54 of U25, or pin 2 of U12 or U27. The line should be low (active).

V

Verify file error at aaaa; wrote/read: xxxxxxxx yyyyyyyy

Description This is not really an error, but a normal function of the VH (verify hex) command to indicate the differences between host file and slave memory. Wrote= file data, Read= memory data.

Appendix D

Schematics

This appendix contains schematics for the following:

- TMS370 application board spare gates
- TMS370 application board
- Power source for the application board
- Master controller
- Master/slave buffer
- ’370Cx1x and ’370Cx5x slaves
- Recreation ports
- Protoarea connections
- PAL description of slave signals
- PAL description of master signals

Appendix E

Glossary

A

analog-to-digital (A/D) converter: An 8-bit successive-approximation converter with internal sample-and-hold circuitry.

application board: An emulator that helps you design, develop, and implement microcontroller designs. The application board comes with a debugger, assembler, and code converter utility.

assembler: A software program that creates a machine-language program from a source file that contains assembly language instructions, directives, and macro directives. The assembler substitutes absolute operation codes for symbolic operation codes, and absolute or relocatable addresses for symbolic addresses.

assembly language: A symbolic language that describes the binary machine code in a more readable form. Each of the 73 unique instructions of the TMS370 family converts to one machine operation.

B

baud: The communication speed for serial ports; equivalent to bits per second.

byte: A sequence of 8 adjacent bits operated upon as a unit.

C

code conversion utility: A software program that translates a COFF object file into one of several standard ASCII hexadecimal formats, suitable for loading into an EPROM programmer.

COFF: *Common Object File Format.* An implementation of the object file format of the same name developed by AT&T. The TMS370 compiler, assembler, and linker use and generate COFF files.

comment: A source statement (or portion of a source statement) that documents or improves the readability of a source file. Comments are not compiled, assembled, or linked; they have no effect on the object file.

constant: A value that does not change during execution.

CPU: An 8-bit register-oriented processor with a status register, program counter, and stack pointer. The TMS370 CPU uses the register file, accessed in one bus cycle, as working registers.

D

debugger: A window-oriented software interface that helps you to debug ’370 programs running on an emulator, CDT370, or application board.

design kit: A low-cost tool that allows you to analyze the hardware and software capabilities of the TMS370 family. The kit includes the TMS370 family application board.

download: To call for and receive a file from another storage medium.

E

EEPROM: *Electrically erasable programmable read-only memory.* Memory that has the capability to be programmed and erased under direct program control.

EPROM: *Erasable programmable read-only memory.* Memory that has the capability to be programmed under direct program control.

I

instruction: The basic unit of programming that causes the execution of one operation; consists of an opcode and operands, along with optional labels and comments.

interrupt: A signal to the CPU that stops the flow of a program and forces the CPU to execute instructions at an address corresponding to the source of the interrupt. When the interrupt is finished, the CPU resumes execution at the point where it was interrupted.

L

label: A symbol that begins in column 1 of a source statement and corresponds to the address of that statement.

linker: A software tool that combines object files to form an object module that can be allocated into system memory and executed by the devices.

LSB: *Least significant bit.*

LSbyte: *Least significant byte.*

M

MC pin: *Mode control pin.* The pin that determines the operating mode of the TMS370 device, depending on the voltage applied to the pin. Twelve volts on the MC pin after reset places the processor in the write protection override (WPO) mode.

memory map: A description of the addresses of the various sections and features of the TMS370 processor. The map depends on the operating mode.

microcomputer mode with external expansion: An operating mode in which the address, control, and data memory extend off-chip to access external memory or peripherals.

microcomputer single-chip mode: An operating mode in which the device uses only on-chip memory.

microprocessor mode with internal program memory: An operating mode in which the on-chip program memory is available to the processor.

microprocessor mode without internal program memory: An operating mode in which the on-chip program memory is not available to the processor. The processor must have external memory.

MSB: *Most significant bit.*

MSbyte: *Most significant byte.*

O

opcode: *Operation code.* The first byte of the machine code that describes to the CPU the type of operation and combination of operands. Some TMS370 instructions use 16-bit opcodes.

P

peripheral file (PF): The 128 or 256 bytes of memory, starting at 1000h, that contain the registers that control the on-board peripherals and system configuration.

peripheral file frames: A set of sixteen contiguous peripheral file registers, usually related by function.

privilege mode: A mode immediately following reset in which the program can alter the privileged registers and bits. Once the privilege mode is disabled, these registers cannot be changed before another reset. This mode does not affect the EEPROM or the watchdog registers.

program counter (PC): A CPU register that identifies the current statement in the program.

PWM: *Pulse width modulation.* A serial signal in which the information is contained in the width (duration) of a pulse of a constant frequency signal. By using the timer compare features, a TMS370 device can output a PWM signal with a constant duty cycle without any program intervention.

R

RAM: *Random access memory.*

ratiometric conversion: An analog-to-digital conversion in which the conversion value is a ratio of the V_{REF} source to the analog input. As V_{REF} is increased, the input voltage required to produce a certain conversion value changes; however, all conversion values keep the same relationship to V_{REF}.

register file (RF): The first 128 or 256 bytes of memory that can be accessed by the majority of the instructions.

$\overline{\text{RESET}}$ pin: A pin that, when held low, starts hardware initialization and insures an orderly software startup. If the MC pin is low when the $\overline{\text{RESET}}$ signal returns high, then the processor enters the microcomputer mode. If the MC pin is high when the $\overline{\text{RESET}}$ signal returns high, then it enters the microprocessor mode.

S

serial communications interface (SCI): A built-in serial interface that can be programmed to be asynchronous or isosynchronous. Many timing, data format, and protocol factors are programmable and controlled by the SCI module in operation.

serial peripheral interface (SPI): A built-in serial interface that facilitates communication between networked master and slave CPUs. As in the SCI, the SPI is set up by software; from then on, the CPU takes no part in timing, data format, or protocol.

stack: The part of the register file used as last-in, first-out memory for temporary variable storage. The stack is used during interrupts and calls to store the current program status. The area occupied by the stack is determined by the stack pointer and by the application program.

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

status register (ST): A CPU register that monitors the operation of the instructions and contains the global interrupt enable bits.

symbol table: A portion of a COFF object file that contains information about the symbols that are defined and used by the file.

W

$\overline{\text{WAIT}}$ pin: The pin that allows an external device to cause the processor to wait an indefinite number of clock cycles. When the wait line is released, the processor resynchronizes with the rising edge of the CLOCKOUT signal and continues with the program.

wait states, automatic: Extra clock cycles inserted automatically on every external memory access to accommodate peripherals or expansion memory with slower access time than the TMS370 processor. These wait states are governed by two control bits: PF AUTOWAIT (SCCR0.5) and AUTOWAIT DISABLE (SCCR1.4).

write-protect override (WPO): The only mode in which a TMS370 device can modify the on-board EEPROM. The WPO mode is entered when external circuitry applies 12 volts to the MC pin after the device has been reset into one of its normal operating modes.

X

XDS/22: A code-development tool that is external to the target system and provides direct control over the TMS370 processor that is on the target system.

Index

A

B

C

D

E

F

H

I

J

L

M

O

P

R

S

T

U

V

W

X

Z

IMPORTANT NOTICE

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

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

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

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