TEXAS INSTRUMENTS

TMS370 Family EPROM/EEPROM Programming Tool

Getting Started Guide

1996 Microprocessor Development Systems

Printed in U.S.A., September 1996
M411007-9741 revision B

SPNU128A

TMS370 Family EPROM/EEPROM Programming Tool Getting Started Guide

SPNU128A
September 1996
BOOK BLOCKS ONLY

TEXAS INSTRUMENTS

Printed on Recycled Paper

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.

Preface

Read This First

About This Manual

This manual helps you install the TMS370 Family EPROM/EEPROM programmer. You can use the programmer to program EPROM and EEPROM devices. The interface to the programmer is similar to that of the TMS370 Family C source debugger. Chapter 2 of this manual describes the programmer interface; for additional information about the interface, see the *TMS370 Family C Source Debugger User's Guide*.

Notational Conventions

This document uses the following conventions.

- The TMS370 family of processors is referred to as the '370.
- Programmer commands are not case sensitive; you can enter them in lowercase, uppercase, or a combination. To emphasize this fact, commands are shown throughout this user's guide in both uppercase and lowercase. However, parameters for certain commands *are* case sensitive; the book describes these instances.
- 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:

  prog *address, length* [*, filename*]

  prog is the command. This command has three parameters, indicated by *address*, *length*, and *filename*. The third parameter is optional.
- 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. Here's an example of a command that has an optional parameter:

  ver *address, length* [*, filename*]

  The VER command has three parameters; the third parameter, *filename*, is optional.

Information About Cautions

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

TMS370 Family C Source Debugger User's Guide (literature number SPNU028) tells you how to invoke the '370 XDS/22 emulator and application board 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. It also includes an advanced tutorial that introduces the breakpoint, trace, and timing features.

TMS370 Family Data Manual (literature number SPNS014) discusses hardware aspects of the TMS370, such as pin functions, architecture, stack operation, and interfaces. The manual also contains the TMS370 assembly language instruction set.

TMS370 and TMS370C8 8-Bit Microcontroller Family Optimizing C Compiler User's Guide (literature number SPNU022) describes the TMS370/C8 8-bit C compiler. This C compiler accepts ANSI standard C source code and produces assembly language source code for the TMS370/C8 8-bit family of devices.

TMS370 and TMS370C8 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 TMS370/C8 8-bit family of devices.

If You Need Assistance . . .

If you want to . . .	Contact Texas Instruments at . . .	
Visit TI online	World Wide Web:	http://www.ti.com
Receive general information or assistance	World Wide Web:	http://www.ti.com/sc/docs/pic/home.htm
	North America, South America:	(214) 644–5580
	Europe, Middle East, Africa	
	Dutch:	33–1–3070–1166
	English:	33–1–3070–1165
	French:	33–1–3070–1164
	Italian:	33–1–3070–1167
	German:	33–1–3070–1168
	Japan (Japanese or English)	
	Domestic toll-free:	0120–81–0026
	International:	81–3–3457–0972 or 81–3–3457–0976
Ask questions about microcontroller product operation or report suspected problems	Hotline:	(713) 274–2370
	Fax:	(713) 274–4203
	Email:	*H370@msg.ti.com
	World Wide Web:	http://www.ti.com/sc/micro
	BBS:	(713) 274–3700 8–N–1
Request tool updates	Software:	(214) 638–0333
	Software fax:	(214) 638–7742
	Hardware:	(713) 274–2285
Order Texas Instruments documentation (see Note 1)	Literature Response Center:	(800) 477–8924
Make suggestions about or report errors in documentation (see Note 2)	Email:	comments@books.sc.ti.com
	Mail:	Texas Instruments Incorporated Technical Publications Manager, MS 702 P.O. Box 1443 Houston, Texas 77251–1443

Notes: 1) The literature number for the book is required; see the lower-right corner on the back cover.

2) Please mention the full title of the book, the literature number from the lower-right corner of the back cover, and the publication date from the spine or front cover.

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.

Trademarks

Microsoft and Windows are registered trademarks of Microsoft Corp.

PC is a trademark of International Business Machines Corporation.

Pentium is a trademark of Intel Corporation.

Contents

Chapter 1

Installing the Programming Tool With Windows

This chapter helps you install the ’370 EPROM/EEPROM programming tool on a PC™ running Windows™. When you complete the installation, turn to Chapter 2, *Using The Programmer*, for more information about using the ’370 programming tool.

Topic **Page**

1.1 System Requirements

To install and use the ’370 family programmer, you need the items listed in the following hardware and software checklists.

Hardware checklist

☐	**Host**	32-bit x86-based or Pentium™ PC with a free full-duplex communication (serial) port, a hard-disk system, and a 1.44M-byte floppy-disk drive
☐	**Memory**	Minimum of 3M bytes RAM
☐	**Display**	Monochrome or color monitor (color recommended)
☐	**Power supply**	5-V/500-mA power supply
☐	**Cable**	RS-232 “straight-through” serial cable
☐	**Optional hardware**	Microsoft™-compatible mouse
☐		EGA- or VGA-compatible graphics display card and a large (17” or 19”) monitor. The programmer has two options that allow you to enlarge the overall size of the programmer display. To use a larger screen size, you must invoke the programmer with an appropriate option. For more information, see Table 2–1 on page 2-2.
☐	**Miscellaneous materials**	Blank, formatted disks

Software checklist

☐	**Operating system**	Windows version 3.1 or later
☐	**Software tools**	TMS370 8-bit microcontroller family assembler and linker
☐		Optional: TMS370 8-bit microcontroller family C compiler
☐	**Required file included as part of debugger package**	*sk370dv.prg* defines the different TMS370 EPROM/EEPROM family devices that can be used for programming and other information specific to the the programmer. **This file must not be modified.**
☐	**Optional files included as part of debugger package**	*init.cmd* is a general-purpose batch file that contains programmer commands. This batch file defines a ’370 memory map. When you start using the programmer, this memory map should be sufficient for your needs. Later, you may want to define your own memory map. For information about setting up your own memory map, see Section 2.4, *Defining a Memory Map,* on page 2-5.
☐		In addition to init.cmd, there are several other .cmd files in the *maps* directory. These .cmd files define memory maps for standard ’370 devices. If you want to emulate a specific device, copy the appropriate .cmd file to your init.cmd file.
☐		*init.clr* is a general-purpose screen configuration file. If init.clr isn’t present when you invoke the programmer, the programmer uses the default screen configuration.
☐		Several .clr (for color monitors) and .mon (for monochrome monitors) screen configuration files are included in the *screens* directory. When you invoke the programmer the first time, the default screen configuration should be sufficient for your needs. Later, you may want to define your own custom configuration. For information about these files and about setting up your own screen configuration, see the *TMS370 Family C Source Debugger User’s Guide*.

1.2 Step 1: Installing the Programmer Board

The programmer board (shown in figure 1-1) is designed to accept five types of integrated circuits (ICs):

- 40-pin dual inline packages (DIPs)
- 28-pin DIPs
- 28-pin plastic leaded chip carriers (PLCCs)
- 68-pin PLCCs
- 44-pin PLCCs

These types of ICs fit the following ’370 devices:

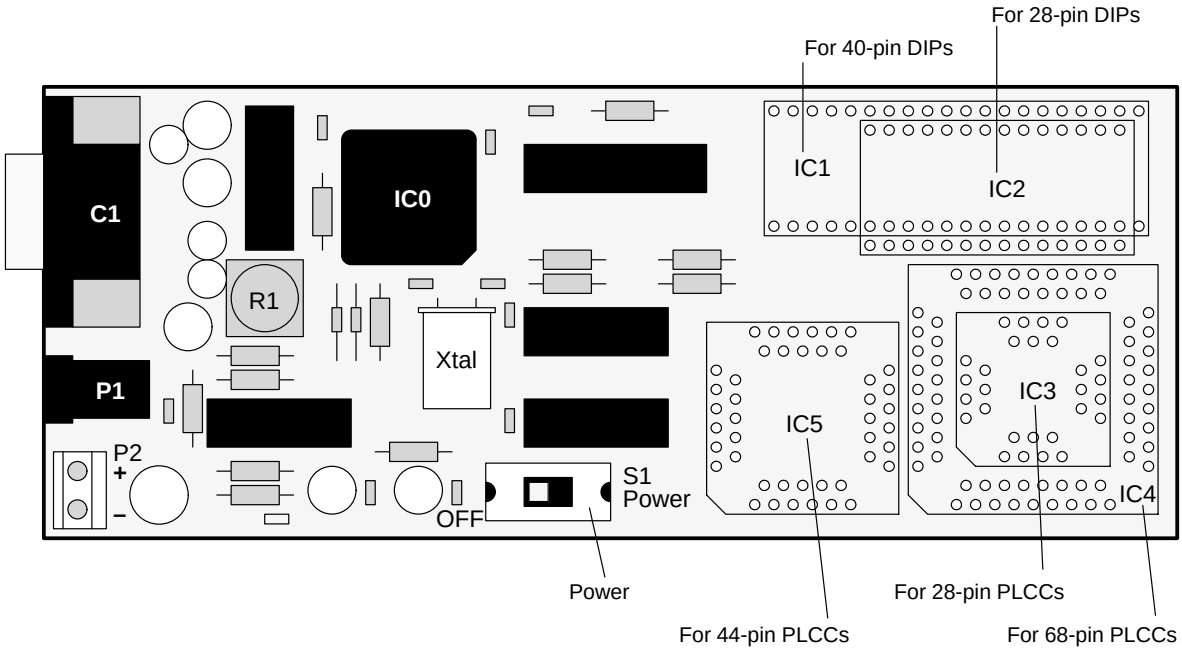
TMS370C702	TMS370C712	TMS370C722	TMS370C732
TMS370C742	TMS370C250	TMS370C256	TMS370C756
TMS370C758	TMS370C768	TMS370C777	

The programmer board also programs the EEPROM for the ROM devices that the one-time programmable (OTP) devices above support. Prototyping devices (devices with SE prefixes) are also supported.

The locations of the different ICs in the programmer board are represented in Figure 1–1 as IC1 to IC5.

A slide switch (S1) applies power to the IC to be programmed.

Figure 1–1. Programmer Board Layout



Improper connection of the programmer board can damage your programmer and/or your PC.

Do not attempt to plug in multiple devices. Plugging in multiple devices can damage your PC and/or programmer.

To install the programmer board, follow these steps:

1) Turn the power switch (S1) on the programmer board to the OFF position.

2) Connect the programmer to the PC. In order to use the ’370 programmer board, you must use a “straight-through” RS-232 cable with the connections between the programmer board and a full-duplex communications (serial) port on your PC, as shown in Table 1–1.

Table 1–1. Host Cable Pin and Signal Assignments

	Programmer			Host Computer		
Function	**DB-9**	**Signal**		**Signal**	**DB-25**	**DB-9**
Data carrier detect	1	DCD	→	DCD	8	1
Data to host	2	RX	→	RX	3	2
Data to programmer	3	TX	←	TX	2	3
Not used	4	DTR	—	DTR	20	4
Signal ground	5	GND	—	GND	7	5
Programmer ready	6	DSR	→	DSR	6	6
Host ready to send	7	CTS	←	RTS	4	7
Programmer ready to send	8	RTS	→	CTS	5	8
Not connected	9	RI	x—	RI	22	9

3) Insert the desired programming socket into the peel-away holes on the programmer board. Figure 1-1 shows the location of the different sets of peel-away holes (IC1–IC5). See page 1-7 for recommended PLCC sockets.

4) Connect a 5-V/500-mA power supply to the programmer board via the barrel jack (P1). Alternatively, you can connect the power supply to the screw power socket (P2). The + and – input lines are labeled on the printed board. Be sure that the power to the programmer board is off.

Inserting or removing ICs while power is applied to the socket can damage your ICs.

5) Insert the integrated circuit (IC).

6) Turn on the programmer by sliding switch S1 to the POWER position. If necessary, you can reset the programmer board via the reset push button (R1).

Programming notes

For PLCC sockets, TI recommends using Yamhichi easy-extract sockets.

28 Pin: IC120–0284–308

44 Pin: IC120–0444–306

68 Pin: IC120–0684–304

You can find Yamhichi Electronics on the World Wide Web at http://www.yeu.com.

The TMS370C758, TMS370C768, and TMS370C777 devices have a dual-bank EPROM:

TMS370 Device	EPROM Block 1		EPROM Block 2	
	Start Address	End Address	Start Address	End Address
’C758	0x2000	0x5FFF	0x6000	0x9FFF
’C768	0x2000	0x5FFF	0x6000	0x9FFF
’C777	0x2000	0x5FFF	0x6000	0x7FFF

For accessing the EPROM area of these devices, the programmer does not allow you to specify an array (start address and length) that overlaps the EPROM bank 1 and bank 2. Such an access must be performed in two passes.

Devices with more than 32k of EPROM are not supported by the programmer.

1.3 Step 2: Installing the Programmer Software

This section explains the process of installing the programmer software on a hard-disk system.

1) Make a backup copy of the product disk.

2) On your hard disk or system disk, create a directory named *370prg.* This directory will contain the programmer software. To create this directory, enter:

   MD C:\370PRG ↵

3) Insert the product disk into drive A. Copy the contents of the disk:

   XCOPY /V /S A:*.* C:\370PRG ↵

You may want to create a Windows program group and use a program-item icon to make it easier to invoke the programmer from within the Windows environment.

Creating a program group

A program group contains program-item icons. You can use program groups to help you organize your icons. To create a program group, follow these steps:

1) From the Windows Program Manager, select File → New. This displays the New Program Object dialog box.

2) Make sure Program Group is selected.

3) Click on OK. This displays the Program Group Properties dialog box.

4) Enter a name for the program group in the Description field.

5) Click on OK. This displays an empty program group with the name you entered.

Using a program-item icon

A program-item icon represents an application that you can run from Windows. Program-item icons are contained inside program groups. The programmer software already has a standard icon that you can use. To use the standard programmer icon, follow these steps:

1) If the program group in which you want to place the icon is not already open, double-click on it to open it.

2) Open the Windows File Manager.

3) Arrange the windows so you can see the program group and the File Manager at the same time.

4) In the prg370 directory of the Windows File Manager, click once on prg370w.exe to select the executable file.

5) Drag and drop the prg370w.exe into the program group. An icon that looks like this displays:

   You can now close the File Manager.

6) Click once on the program icon to select it.

7) In the Program Manager, select File → Properties. This displays the Program Item Properties dialog box.

8) Modify the information in the Command Line field to include the options you normally use at start-up. For a summary of the options that you can use, see Table 2–1, *Summary of Programmer Options*, on page 2-2.

1.4 Step 3: Setting Up the Programmer Environment

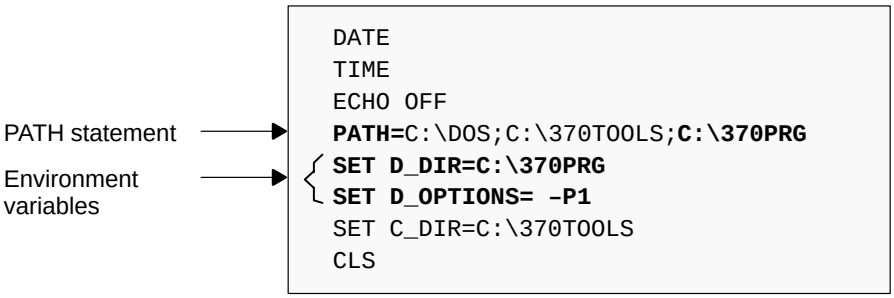
To ensure that your programmer works correctly, you must perform some tasks before you invoke the programmer for the first time or anytime you power up or reboot your PC. You can perform these tasks by entering individual DOS commands, but it is simpler to put the commands in a batch file. You can edit your system's autoexec.bat file, but in some cases, modifying that file can interfere with other applications running on your PC. You can create a separate batch file to perform these tasks instead. No matter which way you choose to do them, these are the tasks you must perform:

- ❑ Modify the PATH statement to identify the 370prg directory.
- ❑ Define environment variables so that the programmer can find the files it needs.

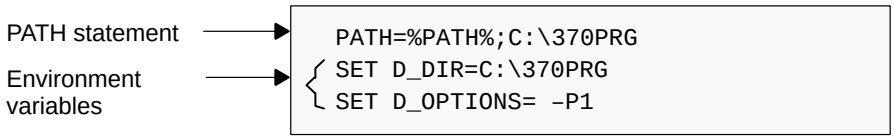
Figure 1–2 *(a)* shows an example of an autoexec.bat file that contains the suggested modifications. Figure 1–2 *(b)* shows a sample batch file that you could create instead of editing the autoexec.bat file. The subsections following the figure explain these modifications.

Figure 1–2. DOS-Command Setup for the Programmer

(a) Sample modified autoexec.bat file to use with the programmer



(b) Sample separate batch file to use with the programmer



Modifying the PATH statement

Define a path to the programmer directory. The general format for doing this is:

```
PATH=C:\370PRG
```

This allows you to invoke the programmer without specifying the name of the directory that contains the programmer executable file.

- If you are modifying an autoexec.bat file that already contains a PATH statement, simply include **;C:\370prg** at the end of the statement, as shown in Figure 1–2 *(a)*.
- If you are creating your own batch file, use a different format for the PATH statement:

  ```
  PATH=%PATH%;C:\370PRG
  ```

  The addition of **%path%;** ensures that this PATH statement won't override PATH statements in any other batch files (including the autoexec.bat file).

Setting up the environment variables

An environment variable is a special system symbol that the programmer uses to find certain types of information. The programmer uses two environment variables, D_DIR and D_OPTIONS. Use the following instructions to set up these environment variables.

- Identify the 370prg directory with D_DIR. Enter:

  ```
  SET D_DIR=C:\370PRG
  ```

  (Be careful not to precede the equal sign with a space.)

  This directory contains auxiliary files (such as sk370dv.prg) that the programmer needs.

- Identify invocation options that you want to use regularly with D_OPTIONS. Use this format:

  SET D_OPTIONS= [*filename*] [*options*]

  (Be careful not to precede the equal sign with a space.)

  The *filename* parameter identifies the optional object file for the programmer to load, and the *options* parameters list the options you want to use at invocation.

  For information about options, see the invocation instructions in Section 2.1, *Invoking the Programmer*.

Invoking the new or modified batch file

- If you modify the autoexec.bat file, be sure to invoke it before invoking the programmer for the first time. To invoke this file, enter:

  AUTOEXEC ↵

- If you create your own batch file, you must invoke it *before* entering Windows. You'll need to invoke your batch file any time that you power up or reboot your PC. For the purpose of this discussion, assume that this sample batch file is named *initprg.bat*. To invoke this file, enter:

  INITPRG ↵

1.5 Step 4: Verifying the Installation

To ensure that you have correctly installed the programmer software, invoke the software and load the sample program:

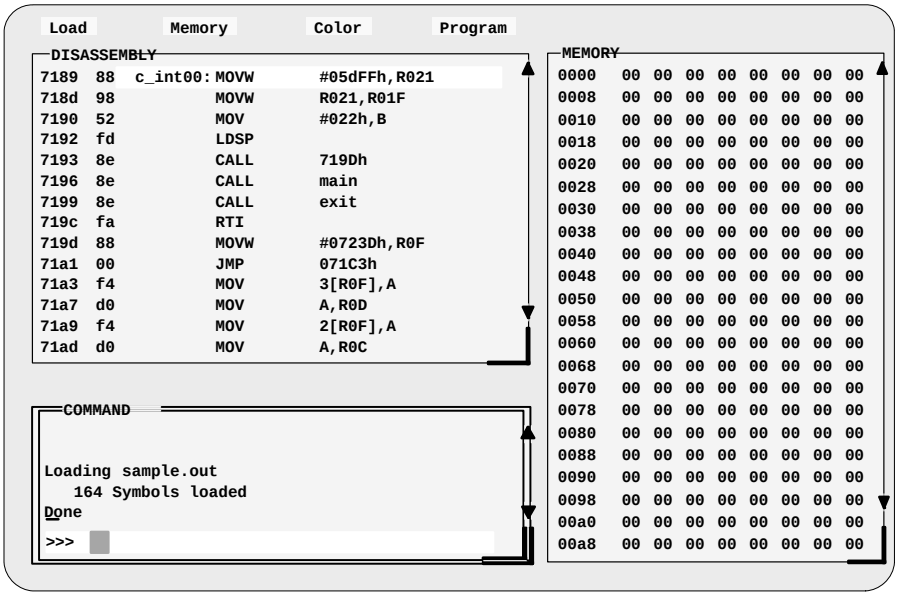
- If you set up an icon for the programmer software, follow these steps:
  1) Start Windows.
  2) Open the program group that contains the programmer icon.
  3) Double-click on the programmer icon.
  4) When the programmer window appears, enter the following from the command line:

     `load sample` ↵

- If you did not set up an icon for the programmer software, follow these steps:
  1) Start Windows.
  2) In the Program Manager or File Manager, select Run... from the File menu.
  3) In the Command line field of the Run dialog box, enter:

     `c:\370prg\prg370w sample` ↵

You should see a display similar to this:



If you see a display similar to this, you have correctly installed your programmer board and software.

If you do not see this kind of display, then your programmer may not be installed properly. Go back through the installation instructions and be sure you have followed each step correctly; then double-click on the program icon.

Notes:

1) Using Windows, you can freely move or resize the programmer display on the screen. If the resized window is bigger than the programmer requires, the extra space is not used. If the resized window is smaller than required, the display is clipped. Note that when the display is clipped, it can't be scrolled.

2) When running Windows, you should run it in either the standard mode or the 386 enhanced mode to get the best results.

Chapter 2

Using the Programmer

This chapter briefly describes the programmer interface. The programmer interface is similar to that of the C source debugger described in the *TMS370 Family C Source Debugger User's Guide*. Refer to the user's guide for additional information about using the interface.

Topic **Page**

2.1 Invoking the Programmer

Before turning on the programmer, be sure that it is plugged in and correctly connected as described in Section 1.2. Turn on the programmer's power switch before running the programmer software.

Here is the basic format for the command that invokes the programmer:

prg370w [*filename*] [*options*]

prg370w is the command that invokes the programmer software.

filename is an optional parameter that lets you select an object file for the programmer to load into memory during invocation. The programmer looks for the file in the current directory; if the file is not in the current directory, you must supply the entire pathname. If you don't supply an extension for the filename, the programmer assumes that the extension is .out.

options supply the programmer with additional information (Table 2–1 summarizes the available options).

You can also specify filename and option information with the D_OPTIONS environment variable (see *Setting up the environment variables* discussion on page 1-11. Table 2–1 lists the programmer options.

Table 2–1. Summary of Programmer Options

To change your...	To...	Use...
Screen options	80 characters by 25 lines 80 characters by 43 lines 80 characters by 50 lines	none (default display) **–b** (any EGA or VGA display) **–bb** (VGA only)
Port options	Identify a serial port	**–p** *serial port* ❑ For serial communication port 1, type: **prg370w –p1** (default) ❑ For serial communication port 2, type: **prg370w –p2**
Initialization file	Identify a new initialization file to use instead of init.cmd	**–t** *filename*
D_OPTIONS	Ignore D_OPTIONS	**–x**

2.2 Descriptions of the Interface Windows and Their Contents

The programmer interface has three windows. Each window is identified by a name in its upper left corner.

- The *COMMAND window* provides an area for typing in commands and displays various types of information such as progress messages, error messages, or command output.
  - The *command line* is where you enter keyboard commands.
  - The *display area* echoes any command that you enter, shows any output from the command, and displays programmer messages.
- The *DISASSEMBLY window* displays the disassembly (assembly language version) of memory contents.
- The *MEMORY window* displays the contents of a range of memory. A MEMORY window has two parts:
  - The first column of numbers identifies the *addresses* of the first column of displayed data. No matter how many columns of data you display, only one address column is displayed. Each address in this column is the address of the data immediately to its right.
  - The remaining columns display the *data* values subsequent addresses. You can display more data by making the window wider and/or longer. You can also edit the data by clicking on the value you want to change and typing in the new value. You can only edit mapped data. The programmer software displays mapped data in white. You cannot edit data displayed in red (not mapped).

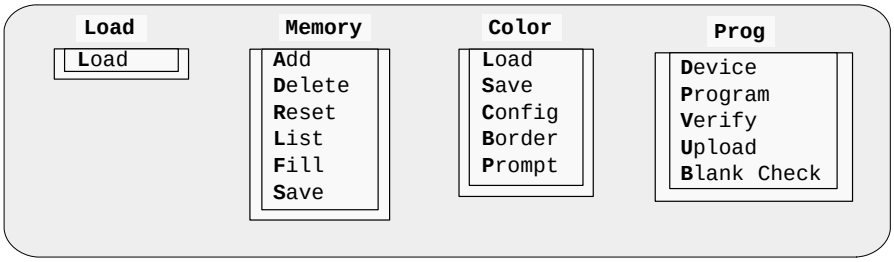
A window's size and its position in the display of the programmer interface aren't fixed—you can resize and move windows. You can also modify a window's contents.

The windows in the programmer interface work the same way that the debugger windows work. For more information about debugger windows, see the *TMS370 Family C Source Debugger User's Guide*.

2.3 Using the Menu Bar and the Pulldown Menus

In addition to the command window, the programmer has a menu bar, which gives you an alternative method for entering many of the programmer commands. The menu bar is at the top of the display.

All of the selections on the menu bar have pulldown menus; if they could all be pulled down at once, they'd look like this:



There are several ways to display the pulldown menus and then execute your selections from them. Executing a command from a menu has the same effect as executing a command by typing it in.

- To use the pulldown menus with a mouse, follow these steps:
  1) Point the mouse cursor at one of the menu-bar selections.
  2) Click the left mouse button. This displays the menu until you are ready to make a selection.
  3) Point the mouse cursor at your selection on the pulldown menu.
  4) When your selection is highlighted, click the left mouse button.
- To use the pulldown menus with the keyboard, follow these steps:
  1) Press the ALT key; don't release it.
  2) Press the key that corresponds to the highlighted letter in the selection name; release both keys. This displays the menu and freezes it.
  3) Press and release the key that corresponds to the highlighted letter of your selection in the menu.

If you display a menu and then decide that you don't want to make a selection from this menu, you can do one of the following:

- Press ESC.
- Point the mouse outside of the menu; press and then release the left mouse button.

2.4 Defining a Memory Map

Before you begin a programmer session, you must supply the programmer with a valid memory map. The memory map tells which areas of memory the software can access and which are inaccessible. Typically, you define the map to match the MEMORY definition in your linker command file.

You can define your memory map in the initialization batch file (init.cmd). The programmer software supplies you with memory maps for standard devices. The programmer loads init.cmd with the ’C756 memory map as the default.

Use the following command format in the command window to access a different memory map file:

take maps*device name***.cmd**

The *device name* is the last four characters of the TMS370 device name. For example, the following command accesses the memory map for the TMS370C758 device.

```
take maps\c758.cmd
```

You can also add, delete, reset, or list memory map ranges by using the Memory pulldown menu selections or by using the MA, MD, MR, and ML commands. For more information about these commands, see the *Defining a Memory Map* chapter in the *TMS370 Family C Source Debugger User’s Guide*.

For the MA command, the programmer accepts only two keywords for the *type* parameter:

EPROM: Identifies a memory area that will be programmed on the EPROM
EEPROM: Identifies a memory area that will be programmed on the EEPROM

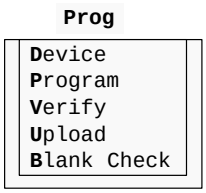
2.5 Programming Memory Devices

The programmer allows you to transfer data from a PC to on-chip EPROM/EEPROM in order to program a ’370 device. The programmer provides you with a complete set of commands for programming a memory device. These commands allow you to:

- ❏ Set the device name.
- ❏ Verify that the device you want to program is blank.
- ❏ Program the device.
- ❏ Verify that the device was programmed correctly.
- ❏ Upload data from the ’370 device’s on-chip EPROM/EEPROM.

The commands that perform these actions are described in the sections that follow. You can enter these commands by:

- ❏ Typing the commands on the programmer command line.
- ❏ Using the Prog menu:



- ❏ Including the commands in a batch file and executing the file with the TAKE command from the command line.

Setting the device name

The internal configuration of programmable devices differs from one device to another, and the programmer must be able to access information about the device that you want to program. Table 2–2 shows the programmer abbreviations for specific device names.

Table 2–2. Programmer Abbreviations for Specific Device Names

Programmer Abbreviation	Device Name	Programmer Abbreviation	Device Name	Programmer Abbreviation	Device Name
c002	TMS370C002	c042	TMS370C042	c756	TMS370C756
c702	TMS370C702	c742	TMS370C742	c758	TMS370C758
c010	TMS370C010	c050	TMS370C050	c758_	TMS370C758_†
c012	TMS370C012	c052	TMS370C052	c067	TMS370C067
c712	TMS370C712	c056	TMS370C056	c068	TMS370C068
c020	TMS370C020	c058	TMS370C058	c069	TMS370C069
c022	TMS370C022	c059	TMS370C059	c768	TMS370C768
c722	TMS370C722	c250	TMS370C250	c077	TMS370C077
c032	TMS370C032	c256	TMS370C256	c777	TMS370C777
c732	TMS370C732	c452	TMS370C452	c0b6	TMS370C0B6
c040	TMS370C040	c456	TMS370C456		

† For TMS370C758A or TMS370C758B, choose device c758. For the obsolete TMS370C758, choose c758_.

You can set the device name so that the programmer can access the correct information about your device by either of two ways:

- Use the PGDEVICE command with the following syntax:

  pgdevice *devicename*

  The *devicename* is the abbreviation of the device you want to program. If you enter the command without specifying a device name, the programmer displays a dialog box for you to enter the device name.

- Select Device from the Prog menu.

  If you select Device from the Prog menu, the programmer displays a dialog box for you to enter the device name.

If you see the following error, make sure you selected a device that the programmer supports.

```
Device selection failed
```

Verifying that the device EPROM is blank

If you want to program EPROM on a ’370 device, you must first verify that the device is blank. EEPROM does not need to be blank before programming. To verify that an EPROM is blank, use the BLANK command.

You can use the BLANK command in either of two ways:

- Use the BLANK command with the following syntax:

  blank *address, length*

  If you enter BLANK without any parameters, the programmer displays a dialog box for you to enter the address and length parameters.

- Select Blank Check from the Prog menu.

  If you select Blank Check from the Prog menu, the programmer displays a dialog box for you to enter the address and length parameters.

When you use the BLANK command, you must supply data for the *address* and *length* parameters.

- The *address* parameter defines the starting address of the range you want to program. It must be within the boundaries of one of the EPROM ranges that you defined previously with the MA command. This parameter can be an absolute address, a C expression, the name of a C function, or an assembly language label.

  The address defaults to the address of the first EPROM range defined in the memory map; after you have blank checked a device, the address defaults to the previous address specified in the memory map.

- The *length* parameter defines the length of the range. The length must be within the boundaries of one of the EPROM ranges that you defined previously with the MA command. This parameter can be any C expression.

  The length defaults to the length of the first EPROM range defined in the memory map; after you have blank checked a device, the length defaults to the previous length specified in the memory map.

Transferring data into the device

Once you have prepared your ’370 device for programming, you can program the device with the contents of the programmer’s memory. To do so, use the PROG command.

You can use the PROG command in either of two ways:

- Use the PROG command with the following syntax:

  prog *address, length*[*, filename*]

  If you enter the PROG command without any parameters, the programmer displays a dialog box for you to enter the address, length, and filename.

- Select Program from the Prog menu

  If you select Program from the Prog menu, the programmer displays a dialog box for you to enter the address, length, and filename.

The PROG command accepts three parameters. *Address* and *length* define a memory range and are required. The *filename* parameter is optional.

- The *address* parameter defines the starting address of the range you want to program. It must be within the boundaries of one of the EPROM/EEPROM ranges that you defined previously with the MA command. This parameter can be an absolute address, a C expression, the name of a C function, or an assembly language label.

  The address defaults to the address of the first EPROM/EEPROM range defined in the memory map; after you have programmed a device, the address defaults to the previous address specified in the memory map.

- The *length* parameter defines the length of the range. The length must be within the boundaries of one of the EPROM/EEPROM ranges that you defined previously with the MA command. This parameter can be any C expression.

  The length defaults to the length of the first EPROM/EEPROM range defined in the memory map; after you have programmed a device, the length defaults to the previous length specified in the memory map.

- The optional *filename* parameter allows you to specify a file that records any errors. If you do not supply a filename, the filename defaults to $$err$$ the first time you enter PROG. After you have specified a filename during the programming session, the default becomes that filename.

If the device programming fails, the software creates an error file in your prg370w directory. Click on the file icon to view the error message. The contents of the error file will look something like this:

```
*******************************************
  PROGRAM EPROM DEVICE

address fail = 0x6024
Array is not successfully programmed

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

Troubleshooting

If you see the following message, ensure that your RS-232 cable is connected, the power switch is in the ON position, and that your device is seated properly in its socket. You will also see this message if you try to program a blank EPROM.

```
Error while accessing device
```

Verifying that the device was programmed correctly

To verify whether a device has been programmed successfully, you can use the VER command. This command compares the on-chip EPROM/EEPROM of the specified address range with that of the programmer memory and logs all of the mismatches in a file.

You can use the VER command in either of two ways:

- Use the VER command with the following syntax:

  ver *address, length*[*, filename*]

  If you enter the VER command without any parameters, the programmer displays a dialog box for you to enter the address, length, and filename.

- Select Verify form the Prog menu

  If you select Verify from the Prog menu, the programmer displays a dialog box for you to enter the address, length, and filename.

The VER command accepts three parameters. *Address* and *length* define a memory range and are required. The *filename* parameter is optional.

- The *address* parameter defines the starting address of the range you want to program. It must be within the boundaries of one of the EPROM/ EEPROM ranges that you defined previously with the MA command. This parameter can be an absolute address, a C expression, the name of a C function, or an assembly language label.

  The address defaults to the address of the first EPROM/EEPROM range defined in the memory map; after you have programmed a device, the address defaults to the previous address specified in the memory map.

- The *length* parameter defines the length of the range. The length must be within the boundaries of one of the EPROM/EEPROM ranges that you defined previously with the MA command. This parameter can be any C expression.

  The length defaults to the length of the first EPROM/EEPROM range defined in the memory map; after you have programmed a device, the length defaults to the previous length specified in the memory map.

- The optional *filename* parameter allows you to specify a file that stores all the mismatches. If you do not supply a filename, the filename defaults to $$err$$ the first time you enter VER. After you have specified a filename during the programming session, the default becomes that filename.

If the device programming fails, the software creates an error file in your prg370w directory. Click on the file icon to view the error message. The contents of the error file will look something like this:

```
************************************************
                      Value            Value
Address            (on device)     (on programmer)
~~~~~~~            ~~~~~~~~~~~     ~~~~~~~~~~~~~~~
0x6000                0x40             0xee
0x6001                0x00             0xee
0x6002                0x40             0xee
0x6003                0x01             0xee
0x6004                0x40             0xee
0x6005                0x02             0xee

Verification complete, 6 error(s) occurred.

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

Uploading data from the device's on-chip EPROM/EEPROM

Once you have programmed a device, you can load its contents from the on-chip EPROM/EEPROM of the device to the programmer memory by using the UPLOAD command.

You can use the UPLOAD command in either of two ways:

- Use the UPLOAD command with the following syntax:

  upload *address, length*

  If you enter UPLOAD without any parameters, the programmer displays a dialog box for you to enter the address and length.

- Select Upload from the Prog menu.

  If you select Upload from the Prog menu, the programmer displays a dialog box for you to enter the address and length.

When you use the UPLOAD command, you must supply data for the *address* and *length* parameters.

- The *address* parameter defines the starting address of the range you want to program. It must be within the boundaries of one of the EPROM/EEPROM ranges that you defined previously with the MA command. This parameter can be an absolute address, a C expression, the name of a C function, or an assembly language label.

  The address defaults to the address of the first EPROM/EEPROM range defined in the memory map; after you have programmed a device, the address defaults to the previous address specified in the memory map.

- The *length* parameter defines the length of the range. The length must be within the boundaries of one of the EPROM/EEPROM ranges that you defined previously with the MA command. This parameter can be any C expression.

  The length defaults to the length of the first EPROM/EEPROM range defined in the memory map; after you have programmed a device, the length defaults to the previous length specified in the memory map.

2.6 Functional Summary of Programmer Commands

This section summarizes the programmer commands according to categories:

- Managing windows
- Displaying files, loading programs, and changing data
- Programming memory devices
- Performing system tasks
- Memory mapping
- Customizing the screen

The programmer commands are similar to the C source debugger version of the same commands. For more information, see the *TMS370 Family C Source Debugger User's Guide*.

Managing windows

These commands enable you to select the active window and move or resize the active window.

To do this...	Use this command...
Reposition the active window	move
Resize the active window	size
Select the active window	win
Make the active window as large as possible	zoom

Displaying files, loading programs, and changing data

These commands enable you to change the display in the DISASSEMBLY window and to load object files into memory. You can also perform the LOAD command by using the Load pulldown menu.

To do this...	Use this command...
Display assembly language code at a specific point	addr
Display assembly language code at a specific address	dasm
Load an object file	load
Display a different range of memory in the MEMORY window	mem
Modify disassembly with the patch assembler	patch

Programming memory devices

These commands allow you to transfer data from emulator RAM to on-chip EPROM/EEPROM to program a memory device.

To do this...	Use this command...	Or this pulldown menu selection...
Set the device name	pgdevice	Prog → Device
Verify that the device EPROM is blank	blank	Prog → Blank Check
Program the device	prog	Prog → Program
Verify that the device is programmed correctly	ver	Prog → Verify
Load data from the device's on-chip EPROM/ EEPROM	upload	Prog → Upload

Performing system tasks

These commands enable you to perform several DOS-like functions and provide you with some control over the target system.

To do this...	Use this command...
Change the current working directory from within the programmer environment	chdir
Clear all displayed information from the COMMAND window display area	cls
List the contents of the current directory or any other directory	dir
Record the information shown in the COMMAND window display area	dlog
Exit the programmer	quit
Associate a beeping sound with the display of error messages	sound
Execute commands from a batch file	take
Check the current version of the programmer	version

Memory mapping

These commands enable you to define the areas of target memory that the programmer can access.

To do this...	Use this command...	Or this pulldown menu selection...
Initialize a block of memory	fill	Memory → Fill
Add an address range to the memory map	ma	Memory → Add
Delete an address range from the memory map	md	Memory → Delete
Display a list of the current memory map settings	ml	Memory → List
Reset the memory map (delete all ranges)	mr	Memory → Reset
Save a block of memory to a system file	ms	Memory → Save

Customizing the screen

These commands allow you to customize the programmer display, then save and later reuse the customized displays.

To do this...	Use this command...	Or this pull-down menu selection...
Change the border style of any window	border	Color → Border
Change the screen colors, but don't update the screen immediately	color	
Change the command-line prompt	prompt	Color → Prompt
Change the screen colors and update the screen immediately	scolor	Color → Config
Load and use a previously saved custom screen configuration	sconfig	Color → Load
Save a custom screen configuration	ssave	Color → Save