HP 64770

TLCS-9000 Emulator Softkey Interface

User's Guide

HEWLETT PACKARD

HP Part No. 64770-97001
June 1995

Edition 1

Notice

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchant ability and fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

Hewlett-Packard Company
P.O. Box 2197
1900 Garden of the Gods Road
Colorado Springs, CO 80901-2197, U.S.A.

RESTRICTED RIGHTS LEGEND Use, duplication,or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013. Hewlett-Packard Company, 3000 Hanover Street, Palo Alto, CA 94304 U.S.A. Rights for non-DOD U.S.Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

Printing History

New editions are complete revisions of the manual. The date on the title page changes only when a new edition is published.

A software code may be printed before the date; this indicates the version level of the software product at the time the manual was issued. Many product updates and fixes do not require manual changes and, manual corrections may be done without accompanying product changes. Therefore, do not expect a
one-to-one correspondence between product updates and manual revisions.

Edition 1	**64770-97001, June 1995**

Using this Manual

This manual shows you how to use the following emulators with the Softkey Interface.

- HP 64770A/B TLCS-9000 emulator

This manual:

- Shows you how to use emulation commands by executing them on a sample program and describing their results.
- Shows you how to use the emulator in-circuit (connected to a target system).
- Shows you how to configure the emulator for your development needs. Topics include: restricting the emulator to real-time execution.

This manual does not:

- Show you how to use every Softkey Interface command and option; the Softkey Interface is described in the *Softkey Interface Reference* manual.

For the most part, the HP 64770A and HP 64770B emulators all operate the same way. Differences of between the emulators are described where they exist. Both the HP 64770A and HP 64770B emulators will be referred to as the "HP 64770A/B TLCS-9000 emulator" or "TLCS-9000 emulator". In the specific instances where HP 64770B emulator differs from HP 64770A emulator, it will be described as "HP 64770A emulator".

Organization

Chapter 1 **Introduction to the TLCS-9000 Emulator.** This chapter briefly introduces you to the concept of emulation and lists the basic features of the TLCS-9000 emulator.

Chapter 2 **Getting Started.** This chapter shows you how to use emulation commands by executing them on a sample program. This chapter describes the sample program and how to: load programs into the emulator, map memory,display and modify memory, display registers, step through program, run programs, set software breakpoints, search memory for data, and use the analyzer.

Chapter 3 **"In-Circuit" Emulation.** This chapter shows you how to install the emulator probe into a demo board and target system and how to use "in-circuit" emulation features.

Chapter 4 **Configuring the Emulator.** This chapter shows you how to: restrict the emulator to real-time execution, allow the target system to insert wait states, and select foreground or background monitor.

Chapter 5 **Using the Emulator.** This chapter describes emulation topics which are not covered in the "Getting Started" chapter.

Conventions

Example commands throughout the manual use the following conventions:

bold Commands, options, and parts of command syntax.

bold italic Commands, options, and parts of command syntax which may be entered by pressing softkey.

normal User specified parts of a command.

$ Represents the HP-UX prompt. Commands which follow the "$" are entered at the HP-UX prompt.

<RETURN> The carriage return key.

Notes

Contents

1 Introduction to the TLCS-9000 Emulator

2 Getting Started

3 In-Circuit Emulation Topics

4 Configuring the Emulator

Illustrations

Tables

Notes

Introduction to the TLCS-9000 Emulator

Introduction

The topics in this chapter include:

- Purpose of the emulator
- Features of the emulator
- Limitations and Restrictions of the emulator

Purpose of the Emulator

The TLCS-9000 emulator is designed to replace the TLCS-9000 microprocessor series in your target system to help you debug/integrate target system software and hardware. The emulator performs just like the processor which it replaces, but at the same time, it gives you information about the bus cycle operation of the processor. The emulator gives you control over target system execution and allows you to view or modify the contents of processor registers, target system memory, and I/O resources. Refer to "Memory Mapping" section in the "Using the Emulator" chapter.

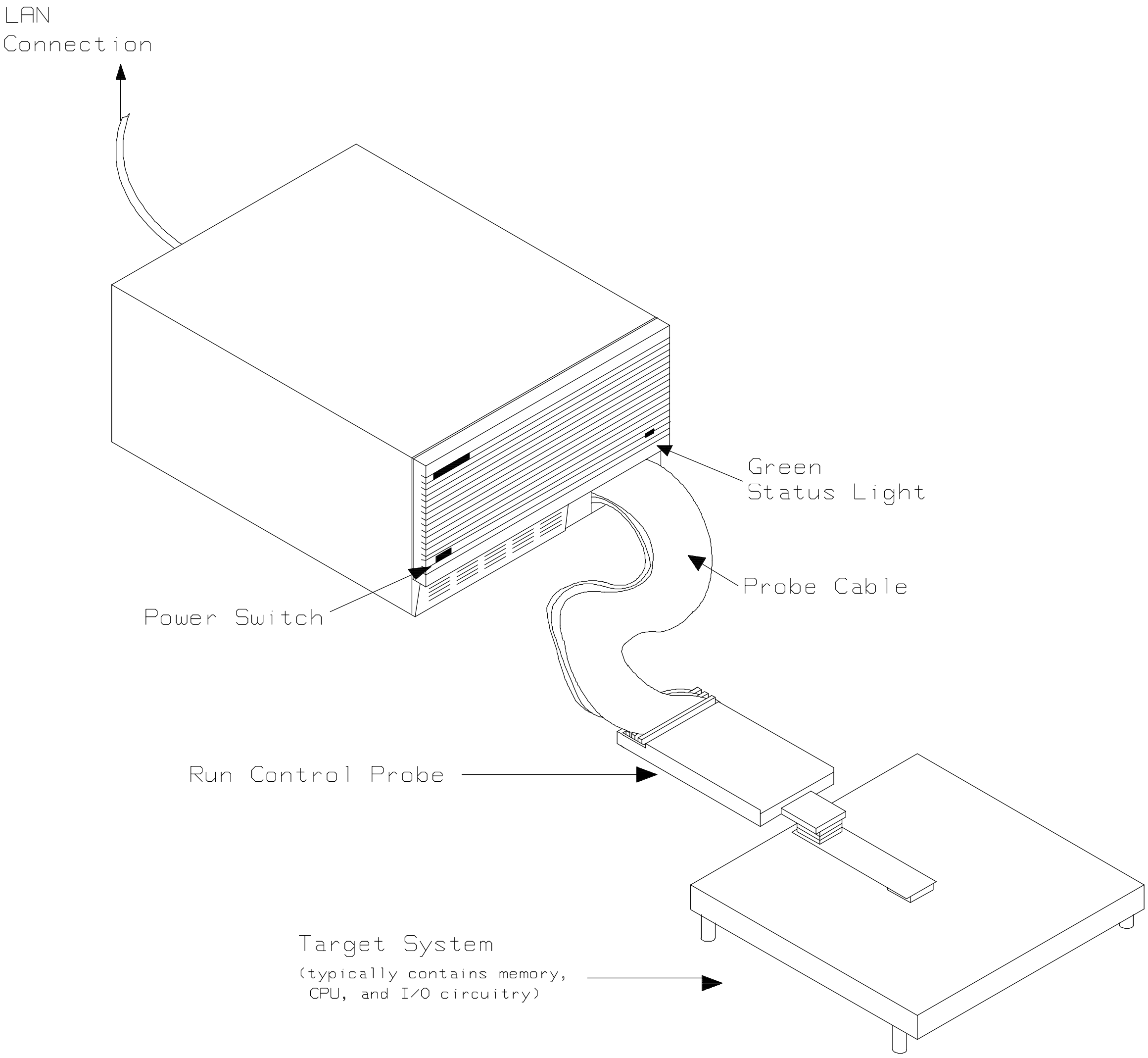


Figure 1-1 HP 64770A/B Emulator for TLCS-9000

Features of the TLCS-9000 Emulator

This section introduces you to the features of the emulator. The chapters which follow show you how to use these features.

Supported Microprocessors

The HP 64770A emulator supports the microprocessors listed in Table 1-1. The HP 64770B emulator supports the microprocessors listed in Table 1-2.

Table 1-1 Supported Microprocessors (HP 64770A)

Supported Microprocessors	Internal ROM size	Internal RAM size
TMP97C241F	0	2K byte
TMP97PS40F	64K byte	2K byte
TMP97CS40F	64K byte	2K byte
TMP97CM40F	32K byte	1K byte
TMP97PW40F	128K byte	4K byte
TMP97CW40F	128K byte	4K byte

Table 1-2 Supported Microprocessors (HP 64770B)

Supported Microprocessors	Internal ROM size	Internal RAM size
TMP97CS42	64K byte	3.5K byte
TMP97PU42	64K byte	3.5K byte
	96K byte	5.25K byte
TMP97CU42	96K byte	5.25K byte
TMP97PW42	128K byte	5.25K byte
TMP97CW42	128K byte	5.25K byte

Clock Speeds

The HP 64770A emulator runs with a target system clock from 4 to 20 MHz. The HP 64770B emulator runs with a target system clock from 4 to 16 MHz.

Emulation memory

The HP TLCS-9000 emulator can be used with one of the following Emulation Memory Modules.

- HP 64171A 256K byte Emulation Memory Module(35 ns)
- HP 64171B 1M byte Emulation Memory Module(35 ns)
- HP 64172A 256K byte Emulation Memory Module(20 ns)
- HP 64172B 1M byte Emulation Memory Module(20 ns)
- HP 64173A 4M byte Emulation Memory Module(25 ns)

You can define up to 7 memory ranges. You can characterize memory ranges as emulation RAM, emulation ROM, target system RAM, target system ROM, or guarded memory. The emulator generates an error message when accesses are made to guarded memory locations. You can also configure the emulator so that writes to memory defined as ROM cause emulator execution to break out of target program execution. Refer to the "Memory Mapping" section in the "Using the emulator" chapter.

Analysis

The HP 64770A emulator is used with one of the following analyzers which allows you to trace code execution and processor activity.

- HP64704A 80-channel Emulation Bus Analyzer
- HP64794A/C/D Deep Emulation Bus Analyzer

The Emulation Bus Analyzer monitors the emulation processor using an internal analysis bus.

Registers

You can display or modify the TLCS-9000 internal register contents.

Emulation Monitor

The emulation monitor is a program that is executed by the emulation processor. It allows the emulation controller to access target system resources, and emulation memory. For example, when you display target system memory, it is monitor program that executes TLCS-9000 instructions which read the target memory locations and send their contents to the emulation controller.

The emulation monitor takes up 64K bytes of processor’s address space.

Single-Step

You can direct the emulation processor to execute a single instruction or a specified number of instructions.

Breakpoints

You can set up the emulator/analyzer interaction so that when the analyzer finds a specific state, emulator execution will break to the emulation monitor.

You can also define software breakpoints in your program. The emulator uses the undefined instruction(7F9Fh) to provide software breakpoint. When you define a software breakpoint, the emulator places a this undefined instruction at the specified address; after the undefined instruction causes emulator execution to break out of your program, the emulator replaces undefined instruction with the original opcode.

Reset Support

The emulator can be reset from the emulation system under your control, or your target system can reset the emulation processor.

Real-Time Operation

Real-time operation signifies continuous execution of your program without interference from the emulator. (Such interference occurs when the emulator temporarily breaks to the monitor so that it can access register contents or memory.)

You can restrict the emulator to real-time execution. When the emulator is executing your program under the real-time restriction, commands which display/modify registers, display/modify memory are not allowed.

Coverage

The TLCS-9000 emulator does not support coverage test.

Easy Products Upgrades

Because the HP 64700 Series development tools (emulator, analyzer, LAN board) contain programmable parts, it is possible to reprogram the firmware and some of the hardware without disassembling the HP 64700B Card Cage. This means that you'll be able to update product firmware, if desired, without having to call an HP field representative to your site

Limitations, Restrictions

Reset While in Monitor

If monitor program is running, $\overline{\text{RESET}}$ signal from target system is ignored while in monitor.

User Interrupts While in Monitor

If the monitor is running, $\overline{\text{NMI}}$, INT0-7(edge sense) for HP 64770A, IREQ for HP 64770B signals from target system are suspended until the emulator goes into user program operation. Other interrupts are ignored.

While Executing Step Command

While stepping user program, interrupts are ignored. While single stepping, BUSRQ from target system is always ignored even if BUSRQ from target system is enabled.

Note

You should not use step command in case the interrupt handler's punctuality is critical.

Watch Dog Timer (HP 64770A Only)

When the HP 64770A emulator breaks into the monitor, the watched dog timer is resets, and disabled until the emulator goes into user program operation.

You must display/modify MDMOD register by "reg" command instead of "m" command.

Vector Area

You need to configure vector entry for the emulator to realize the following features.

- Break
- Single-Step
- Software Break Point

Refer to the "Vector Area Setting" section in the "Using the Emulator" Chapter in this manual.

Register Bank

When the emulator breaks into the monitor, the PC and PSW are stored at register bank of "CBP-1" in the same way as the emulator accepts interrupts.

Unbreaking into the Monitor

The emulator can not break into the monitor when the emulation processor is the following states.

- Standby Mode by HALT instruction
- Power Save state(Hardware standby mode) by PS signal
- Hold Mode by $\overline{\text{BUSRQ}}$ signal
- Reset state by $\overline{\text{RESET}}$ signal from target

Emulation Memory

When you use the emulator in single chip mode, you need the emulation memory because the emulator maps internal ROM/RAM area as emulation memory.

If you use the emulator in single chip mode or the emulation processor does burst fetch, the emulation memory module is restricted by clock speed as following.

HP 64770A

If clock speed is equal to 18MHz or greater 18MHz, you need HP64712A/B emulation memory module. If clock speed is less than 18MHz, you can use HP64712A/B and HP64713A emulation memory modules. If clock speed is less than 15MHz, you can use HP64171A/B, HP64172A/B and HP64713A emulation memory module.

HP 64770B

If clock speed is equal to 16MHz or less than 16MHz, you can use HP64712A/B and HP64713A emulation memory modules. If clock speed is less than 15MHz, you can use HP64171A/B, HP64172A/B and HP64713A emulation memory module.

Evaluation Chip

Hewlett-Packard makes no warranty of the problem caused by the TLCS-9000 Evaluation chip in the emulator.

2

Getting Started

Introduction

This chapter will lead you through a basic, step by step tutorial that shows how to use the HP 64770A/B emulator (for the TLCS-9000 microprocessor) with the Softkey Interface.

This chapter will:

- Tell you what must be done before you can use the emulator as shown in the tutorial examples.
- Describe the demo program used for this chapter’s examples.

This chapter will show you how to:

- Start up the Softkey Interface.
- Load programs into emulation and target system memory.
- Enter emulation commands to view execution of the demo program.

Before You Begin

Prerequisites

Before beginning the tutorial presented in this chapter, you must have completed the following tasks:

1. Connected the emulator to your computer. The *HP 64700 Series Installation/Service* manual show you how to do this.

2. Installed the Softkey Interface software on your computer. Refer to the *HP 64700 Series Installation/Service* manual for instructions on installing software.

3. In addition, you should read and understand the concepts of emulation presented in the *Concepts of Emulation and Analysis* manual. The *Installation/Service* manual also covers HP 64700 system architecture. A brief understanding of these concepts may help avoid questions later.

   You should read the *Softkey Interface Reference* manual to learn how to use the Softkey Interface in general. For the most part, this manual contains information specific to the TLCS-9000 emulator.

A Look at the Demo Program

The demo program is *spmt_demo* consisting of source program *spmt_demo.c* and *init.s.*

Where is the spmt_demo Software?

The demo program is shipped with the Softkey Interface and may be copied from the following directory.

/usr/hp64000/demo/emul/hp64770

Assembling/Compiling the Demo Program

The demo program is written for and compiled/linked with the MICROTEC RESEARCH Inc. MCCT9K C Compiler Package. The demo program was assembled/compiled with the following commands.

```
$ mcct9k -c -g spmt_demo.c <RETURN>
$ asmt9k -f debug,casemcct9k -l init.s >
init.lis <RETURN>
```

Linking the Demo Program

The following command was used to generate the absolute file. The "spmt_demo.cmd" linker command file is shown in figure 2-1.

```
$ lnkt9k -c spmt_demo.cmd -M<RETURN>
```

```
DEBUG_SYMBOLS
SECT stack=$400
SECT zerovars=$800
SECT code=$a00
LOAD init.o
LOAD spmt_demo.o
END
```

Figure 2-1 Linker Command File

Entering the Softkey Interface

If you have installed your emulator and Softkey Interface software as directed in the *HP 64700 Series Emulators Softkey Interface Installation Notice*, you are ready to enter the interface. The Softkey Interface can be entered from the HP-UX shell.

From the HP-UX Shell

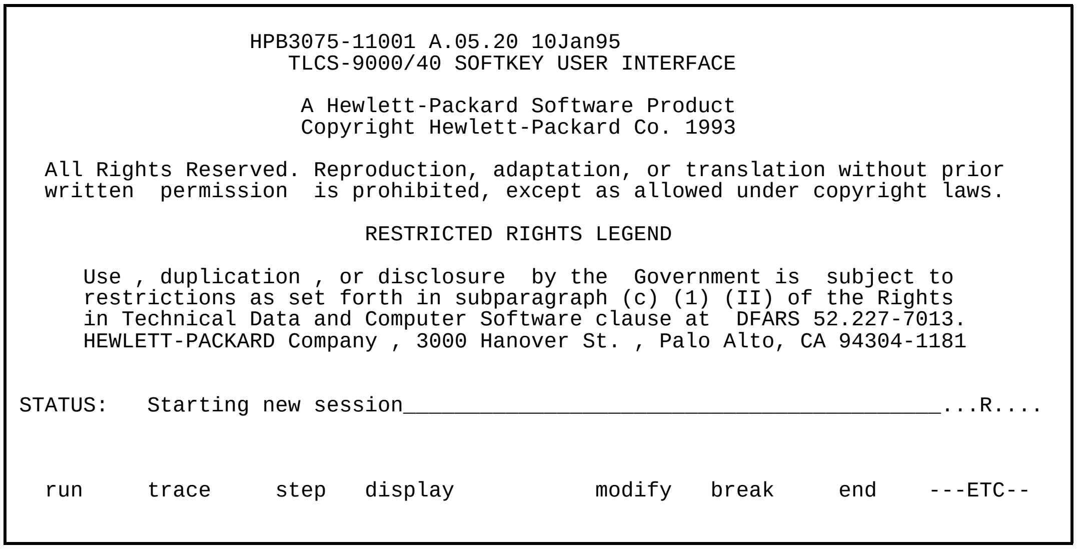
If **/usr/hp64000/bin** is specified in your PATH environment variable, you can also enter the Softkey Interface with the following command.

```
$ emul700 <emul_name> <RETURN>
```

The "emul_name" in the command above is the logical emulator name given in the HP 64700 emulator device table (/usr/hp64000/etc/64700tab.net).

```
#--------+------------+-----------+-------------------------------------------
# Channel|  Logical   | Processor | Remainder of Information for the Channel
#  Type  |    Name    |   Type    | (IP address for LAN connections)
#--------+------------+-----------+-------------------------------------------
    lan:      tlcs        t9k40        21.17.9.143
```

If this command is successful, you will see a display similar to figure 2-2. The status message shows that the default configuration file has been loaded. If the command is not successful, you will be given an error message and returned to the HP-UX prompt. Error messages are described in the *Softkey Interface Reference* manual.



```
                  HPB3075-11001 A.05.20 10Jan95
                     TLCS-9000/40 SOFTKEY USER INTERFACE

                      A Hewlett-Packard Software Product
                      Copyright Hewlett-Packard Co. 1993

  All Rights Reserved. Reproduction, adaptation, or translation without prior
  written  permission  is prohibited, except as allowed under copyright laws.

                           RESTRICTED RIGHTS LEGEND

     Use , duplication , or disclosure  by the  Government is  subject to
     restrictions as set forth in subparagraph (c) (1) (II) of the Rights
     in Technical Data and Computer Software clause at  DFARS 52.227-7013.
     HEWLETT-PACKARD Company , 3000 Hanover St. , Palo Alto, CA 94304-1181


STATUS:   Starting new session__________________________________________...R....



  run     trace     step   display          modify   break     end    ---ETC--
```

Figure 2-2 Softkey Interface Display

Configure the Emulator for Examples

To do operations described in this chapter (loading absolute program into emulation memory, displaying memory contents, etc), you need to configure the emulator as below. For detailed description of each configuration option (question), refer to the "*Configuring the Emulator*" chapter.

To get into the configuration session of the emulator, enter the following command.

modify configuration <RETURN>

Answer to the series of questions as below.

`Restrict to real-time runs?` ***no*** <RETURN>

When you use HP 64770A emulator, answer this question as shown.

`Processor type?` ***97CM40*** <RETURN>

When you use HP 64770B emulator, answer this question as shown.

`Processor type?` ***97CU42*** <RETURN>

```
Processor operation mode?  external_bus <RETURN>
Monitor base address?  0F0000H <RETURN>
Enable emulation VBP?  yes <RETURN>
```

When you use HP 64770A emulator, answer this question as shown.

```
Vector base address (97PS/CM40)?  0FF0000H <RETURN>
```

When you use HP 64770B emulator, answer this question as shown.

```
Vector base address (97CU42)?  0FE7800H <RETURN>
Initial CBP value?  01H <RETURN>
Modify memory configuration?  yes <RETURN>
```

Now you should be facing memory mapping screen. If you use HP 64770A emulator, two mapper terms must be specified for the demo program. Enter the following lines to map the program code area as emulation ROM, data area as emulation RAM. If you use HP 64770B emulator, you do not need to map because mapper terms for the demo program are specified automatically.

```
    400h thru 9ffh emulation  ram <RETURN>
    0a00h thru 0fffh emulation  rom <RETURN>
    end <RETURN>
Modify emulator pod configuration?  no <RETURN>
Modify debug/trace options?  no <RETURN>
Modify simulated I/O configuration?  no <RETURN>
Modify interactive measurement specification?  no <RETURN>
```

If you wish to save the configuration specified above, answer this question as shown.

```
Configuration file name?  spmt_demo <RETURN>
```

Now you are ready to go ahead. Above configuration is used through out this chapter.

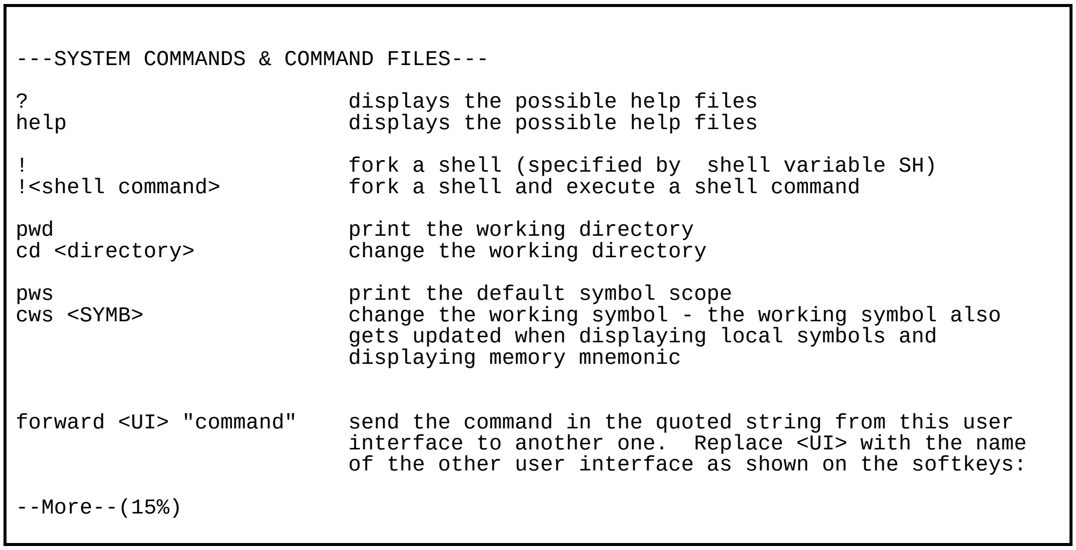
On-Line Help

There are two ways to access on-line help in the Softkey Interface. The first is by using the Softkey Interface help facility. The second method allows you to access the firmware resident Terminal Interface on-line help information.

Softkey Driven Help

To access the Softkey Interface on-line help information, type either "help" or "?" on the command line; you will notice a new set of softkeys. By pressing one of these softkeys and <RETURN>, you can cause information on that topic to be displayed on your screen. For example, you can enter the following command to access "system command" help information.

```
? system_commands <RETURN>
```



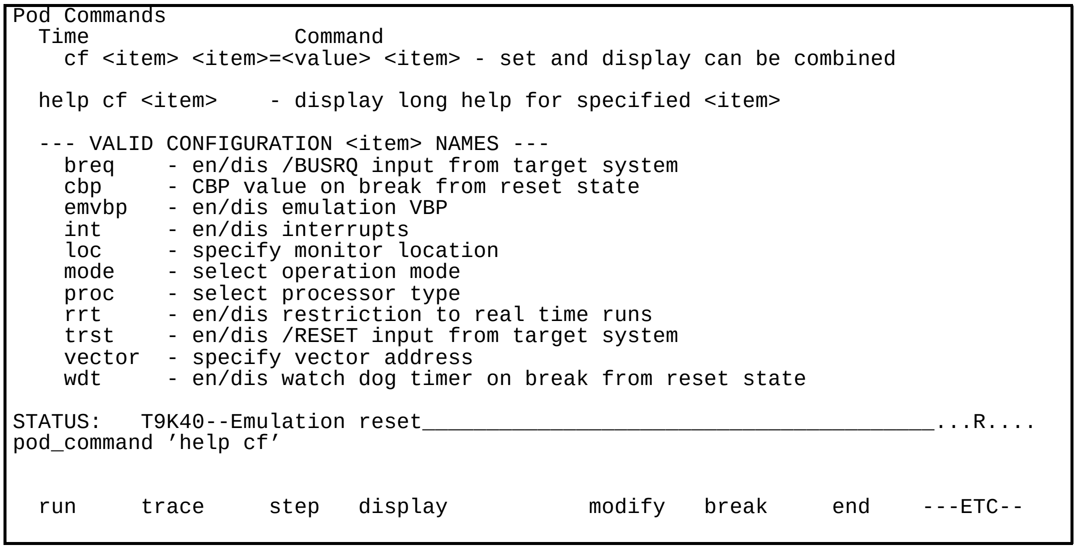
```
---SYSTEM COMMANDS & COMMAND FILES---

?                          displays the possible help files
help                       displays the possible help files

!                          fork a shell (specified by  shell variable SH)
!<shell command>           fork a shell and execute a shell command

pwd                        print the working directory
cd <directory>             change the working directory

pws                        print the default symbol scope
cws <SYMB>                 change the working symbol - the working symbol also
                           gets updated when displaying local symbols and
                           displaying memory mnemonic


forward <UI> "command"     send the command in the quoted string from this user
                           interface to another one.  Replace <UI> with the name
                           of the other user interface as shown on the softkeys:

--More--(15%)
```

The help information is scrolled on to the screen. If there is more than a screenful of information, you will have to press the space bar to see the next screenful, or the <RETURN> key to see the next line, just as you do with the HP-UX **more** command. After all the information on the particular topic has been displayed (or after you press "q" to quit scrolling through information), you are prompted to press <RETURN> to return to the Softkey Interface.

Pod Command Help

To access the emulator’s firmware resident Terminal Interface help information, you can use the following commands.

display pod_command <RETURN>
pod_command ’help cf’ <RETURN>



```
Pod Commands
  Time                Command
    cf <item> <item>=<value> <item> - set and display can be combined

  help cf <item>    - display long help for specified <item>

  --- VALID CONFIGURATION <item> NAMES ---
    breq    - en/dis /BUSRQ input from target system
    cbp     - CBP value on break from reset state
    emvbp   - en/dis emulation VBP
    int     - en/dis interrupts
    loc     - specify monitor location
    mode    - select operation mode
    proc    - select processor type
    rrt     - en/dis restriction to real time runs
    trst    - en/dis /RESET input from target system
    vector  - specify vector address
    wdt     - en/dis watch dog timer on break from reset state

STATUS:   T9K40--Emulation reset________________________________________...R....
pod_command ’help cf’


  run     trace     step   display          modify   break     end    ---ETC--
```

The command enclosed in string delimiters (", ’, or ^) is any Terminal Interface command, and the output of that command is seen in the pod_command display. The Terminal Interface help (or ?) command may be used to provide information on any Terminal Interface command or any of the emulator configuration options (as the example command above shows).

Note

If you want to use the Terminal Interface command by entering from keyboard directly, you can do it after entering the following command.

pod_command keyboard

Loading Absolute Files

The "load" command allows you to load absolute files into emulation or target system memory. You can load absolute files in the following formats:

- IEEE-695
- HP absolute(No symbols)

The "load" command has no special options for loading different absolute file formats; instead, the contents of the file are examined to determine the format being used. If you wish to load only that portion of the absolute file that resides in memory mapped as emulation RAM or ROM, use the "load emul_mem" syntax. If you wish to load only the portion of the absolute file that resides in memory mapped as target RAM, use the "load user_mem" syntax. If you want both emulation and target memory to be loaded, do not specify "emul_mem" or "user_mem". For example:

```
load spmt_demo <RETURN>
```

Note

When you use HP 64770B emulator, you must enter "break" command before you load a program. Enter the following command.

```
break  <RETURN>
```

Note

When loading a program if the status line shows

```
"ERROR:      No absolute file, No database:
spmt_demo
```

, you may NOT be in the directory that your program is in. To find out what directory you are in, enter:

```
! pwd <RETURN>
```

The **"!"** allows you to use an HP-UX shell command. To move into the correct directory, enter:

```
cd <directory path>  <RETURN>
```

You can also specify the pathname where your program resides. For example, you could enter:

```
load
/usr/hp64000/demo/emul/hp64770/spmt_demo
<RETURN>
```

Displaying Symbols

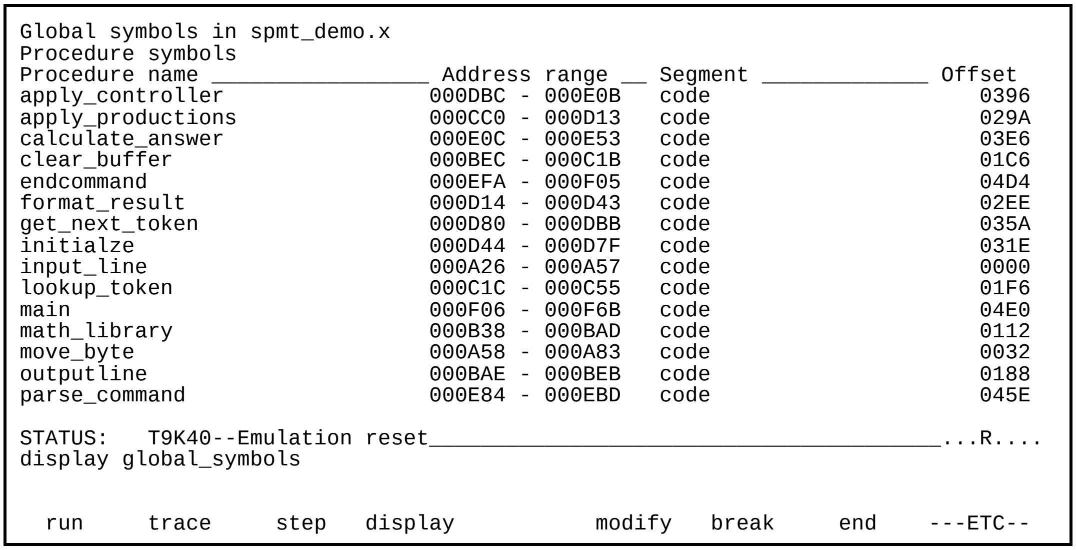
When you load an absolute file into memory (unless you use the "nosymbols" syntax), symbol information is also loaded. Both global symbols and symbols that are local to a source file can be displayed.

Global

To display global symbols, enter the following command.

display global_symbols <RETURN>

Listed are address ranges associated with a symbol, the segment that the symbol is associated with, and the offset of that symbol within the segment.



```
Global symbols in spmt_demo.x
Procedure symbols
Procedure name _________________ Address range __ Segment _____________ Offset
apply_controller                000DBC - 000E0B   code                    0396
apply_productions               000CC0 - 000D13   code                    029A
calculate_answer                000E0C - 000E53   code                    03E6
clear_buffer                    000BEC - 000C1B   code                    01C6
endcommand                      000EFA - 000F05   code                    04D4
format_result                   000D14 - 000D43   code                    02EE
get_next_token                  000D80 - 000DBB   code                    035A
initialze                       000D44 - 000D7F   code                    031E
input_line                      000A26 - 000A57   code                    0000
lookup_token                    000C1C - 000C55   code                    01F6
main                            000F06 - 000F6B   code                    04E0
math_library                    000B38 - 000BAD   code                    0112
move_byte                       000A58 - 000A83   code                    0032
outputline                      000BAE - 000BEB   code                    0188
parse_command                   000E84 - 000EBD   code                    045E

STATUS:   T9K40--Emulation reset________________________________________...R....
display global_symbols


  run     trace     step   display          modify   break     end    ---ETC--
```

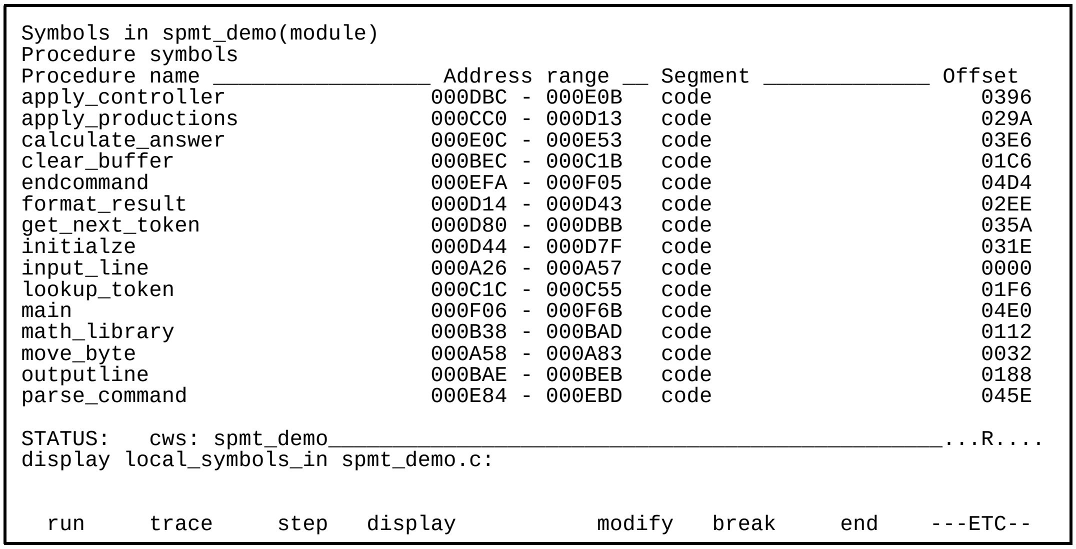
Local

When displaying local symbols, you must include the name of the source file in which the symbols are defined. For example,

```
display local_symbols_in spmt_demo.c:
<RETURN>
```

As you can see, the procedure symbols and static symbols in "spmt_demo.c" are displayed.
To list the next symbols, press the <PGDN> or <Next> key. the source reference symbols in "spmt_demo.c" will be displayed.

Listed are: address ranges associated with a symbol, the segment that the symbol is associated with, and the offset of that symbol within the segment.



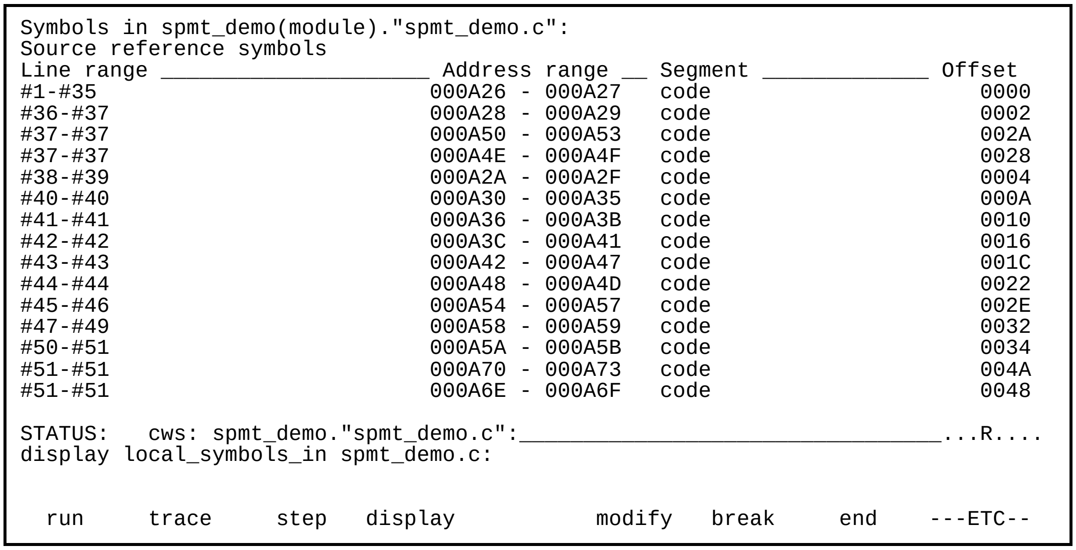
```
Symbols in spmt_demo(module)
Procedure symbols
Procedure name _________________ Address range __ Segment _____________ Offset
apply_controller                000DBC - 000E0B   code                     0396
apply_productions               000CC0 - 000D13   code                     029A
calculate_answer                000E0C - 000E53   code                     03E6
clear_buffer                    000BEC - 000C1B   code                     01C6
endcommand                      000EFA - 000F05   code                     04D4
format_result                   000D14 - 000D43   code                     02EE
get_next_token                  000D80 - 000DBB   code                     035A
initialze                       000D44 - 000D7F   code                     031E
input_line                      000A26 - 000A57   code                     0000
lookup_token                    000C1C - 000C55   code                     01F6
main                            000F06 - 000F6B   code                     04E0
math_library                    000B38 - 000BAD   code                     0112
move_byte                       000A58 - 000A83   code                     0032
outputline                      000BAE - 000BEB   code                     0188
parse_command                   000E84 - 000EBD   code                     045E

STATUS:   cws: spmt_demo________________________________________________...R....
display local_symbols_in spmt_demo.c:


  run     trace     step   display          modify   break     end    ---ETC--
```

Source Lines

To display the address ranges associated with the program's source file, you must display the local symbols in the file. For example:

display local_symbols_in spmt_demo.c:
<RETURN>

And scroll the information down on the display with up arrow,or <Next> key.

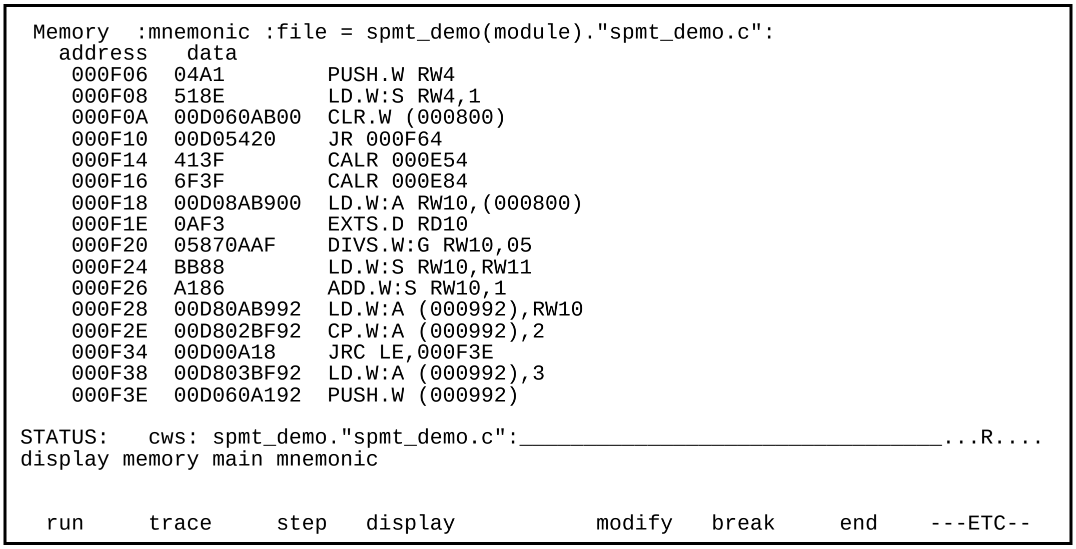


```
Symbols in spmt_demo(module)."spmt_demo.c":
Source reference symbols
Line range ____________________ Address range __ Segment _____________ Offset
#1-#35                          000A26 - 000A27   code                     0000
#36-#37                         000A28 - 000A29   code                     0002
#37-#37                         000A50 - 000A53   code                     002A
#37-#37                         000A4E - 000A4F   code                     0028
#38-#39                         000A2A - 000A2F   code                     0004
#40-#40                         000A30 - 000A35   code                     000A
#41-#41                         000A36 - 000A3B   code                     0010
#42-#42                         000A3C - 000A41   code                     0016
#43-#43                         000A42 - 000A47   code                     001C
#44-#44                         000A48 - 000A4D   code                     0022
#45-#46                         000A54 - 000A57   code                     002E
#47-#49                         000A58 - 000A59   code                     0032
#50-#51                         000A5A - 000A5B   code                     0034
#51-#51                         000A70 - 000A73   code                     004A
#51-#51                         000A6E - 000A6F   code                     0048

STATUS:   cws: spmt_demo."spmt_demo.c":_________________________________...R....
display local_symbols_in spmt_demo.c:


  run     trace     step   display           modify   break     end    ---ETC--
```

Displaying Memory in Mnemonic Format

You can display, in mnemonic format, the absolute code in memory. For example to display the memory of the demo program,

```
display memory main mnemonic <RETURN>
```



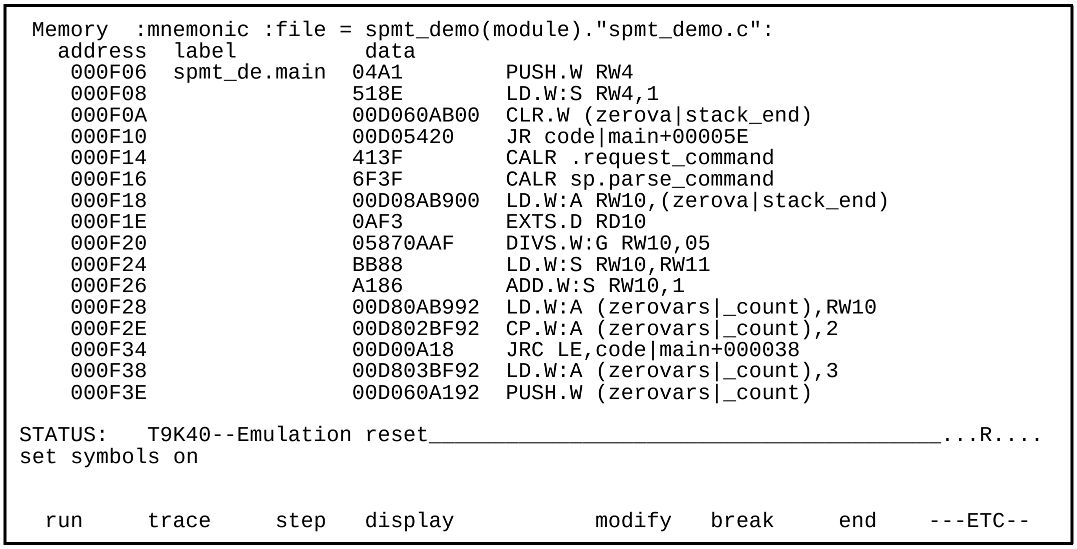
```
 Memory  :mnemonic :file = spmt_demo(module)."spmt_demo.c":
   address   data
    000F06  04A1        PUSH.W RW4
    000F08  518E        LD.W:S RW4,1
    000F0A  00D060AB00  CLR.W (000800)
    000F10  00D05420    JR 000F64
    000F14  413F        CALR 000E54
    000F16  6F3F        CALR 000E84
    000F18  00D08AB900  LD.W:A RW10,(000800)
    000F1E  0AF3        EXTS.D RD10
    000F20  05870AAF    DIVS.W:G RW10,05
    000F24  BB88        LD.W:S RW10,RW11
    000F26  A186        ADD.W:S RW10,1
    000F28  00D80AB992  LD.W:A (000992),RW10
    000F2E  00D802BF92  CP.W:A (000992),2
    000F34  00D00A18    JRC LE,000F3E
    000F38  00D803BF92  LD.W:A (000992),3
    000F3E  00D060A192  PUSH.W (000992)

STATUS:   cws: spmt_demo."spmt_demo.c":_________________________________...R....
display memory main mnemonic


  run     trace     step   display          modify   break     end    ---ETC--
```

Notice that you can use symbols when specifying expressions. The global symbol **main** is used in the command above to specify the starting address of the memory to be displayed.

Display Memory with Symbols

If you want to see symbol information with displaying memory in mnemonic format, the emulator Softkey Interface provides "set symbols" command. To see symbol information, enter the following command.

set symbols on <RETURN>



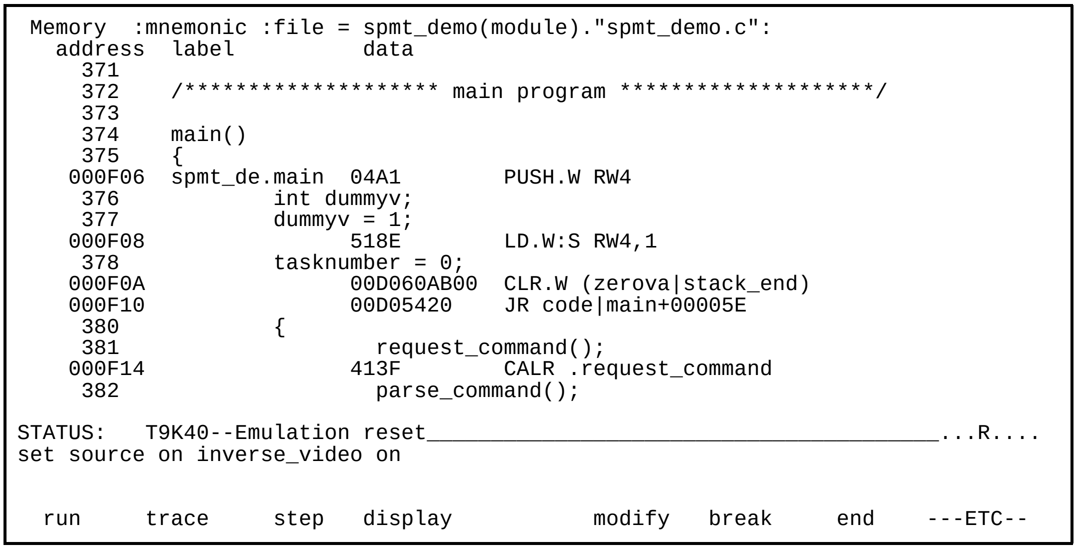
```
 Memory  :mnemonic :file = spmt_demo(module)."spmt_demo.c":
   address  label          data
    000F06  spmt_de.main  04A1        PUSH.W RW4
    000F08                518E        LD.W:S RW4,1
    000F0A                00D060AB00  CLR.W (zerova|stack_end)
    000F10                00D05420    JR code|main+00005E
    000F14                413F        CALR .request_command
    000F16                6F3F        CALR sp.parse_command
    000F18                00D08AB900  LD.W:A RW10,(zerova|stack_end)
    000F1E                0AF3        EXTS.D RD10
    000F20                05870AAF    DIVS.W:G RW10,05
    000F24                BB88        LD.W:S RW10,RW11
    000F26                A186        ADD.W:S RW10,1
    000F28                00D80AB992  LD.W:A (zerovars|_count),RW10
    000F2E                00D802BF92  CP.W:A (zerovars|_count),2
    000F34                00D00A18    JRC LE,code|main+000038
    000F38                00D803BF92  LD.W:A (zerovars|_count),3
    000F3E                00D060A192  PUSH.W (zerovars|_count)

STATUS:   T9K40--Emulation reset________________________________________...R....
set symbols on


  run     trace     step   display           modify   break     end    ---ETC--
```

As you can see, the memory display shows symbol information.

Display Memory with Source Code

If you want to reference the source line information with displaying memory in mnemonic format, the emulator Softkey Interface provides "set source" command. To reference the source line information in inverse video, enter the following command:

set source on inverse_video on <RETURN>



```
 Memory  :mnemonic :file = spmt_demo(module)."spmt_demo.c":
   address  label          data
     371
     372    /******************** main program ********************/
     373
     374    main()
     375    {
    000F06  spmt_de.main  04A1        PUSH.W RW4
     376            int dummyv;
     377            dummyv = 1;
    000F08                518E        LD.W:S RW4,1
     378            tasknumber = 0;
    000F0A                00D060AB00  CLR.W (zerova|stack_end)
    000F10                00D05420    JR code|main+00005E
     380            {
     381                    request_command();
    000F14                413F        CALR .request_command
     382                    parse_command();

STATUS:   T9K40--Emulation reset________________________________________...R....
set source on inverse_video on


  run     trace     step   display           modify   break     end    ---ETC--
```

To see the memory without source line referencing, enter the following command:

set source off <RETURN>

Running the Program

The "run" command lets you execute a program in memory. Entering the "run" command by itself causes the emulator to begin executing at the current program counter address. The "run from" command allows you to specify an address at which execution is to start.

From Transfer Address

The "run from transfer_address" command specifies that the emulator start executing at a previously defined "start address". Transfer addresses are defined in assembly language source files with the END assembler directive (i.e., pseudo instruction). Enter:

```
run from transfer_address <RETURN>
```

From Reset

The "run from reset" command specifies that the emulator begin executing from reset vector as actual microprocessor does.

(See "Running From Reset" section in the "In-Circuit Emulation" chapter).

Displaying Memory

The demo program "spmt_demo.c" alters memory.

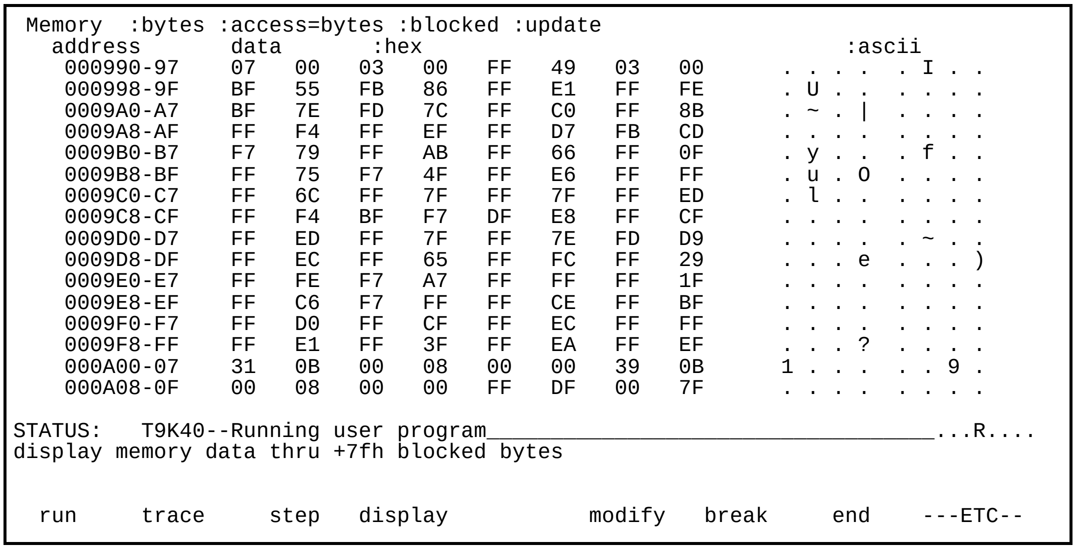
Using Symbolic Addresses

In the following display, the memory range is displayed using symbolic addresses **data**.

The memory display window is periodically updated. For example, enter the following command:

```
display memory data thru +7fh blocked bytes
<RETURN>
```

This command string is used to specify the range of memory from **data** to **data+7fh**.



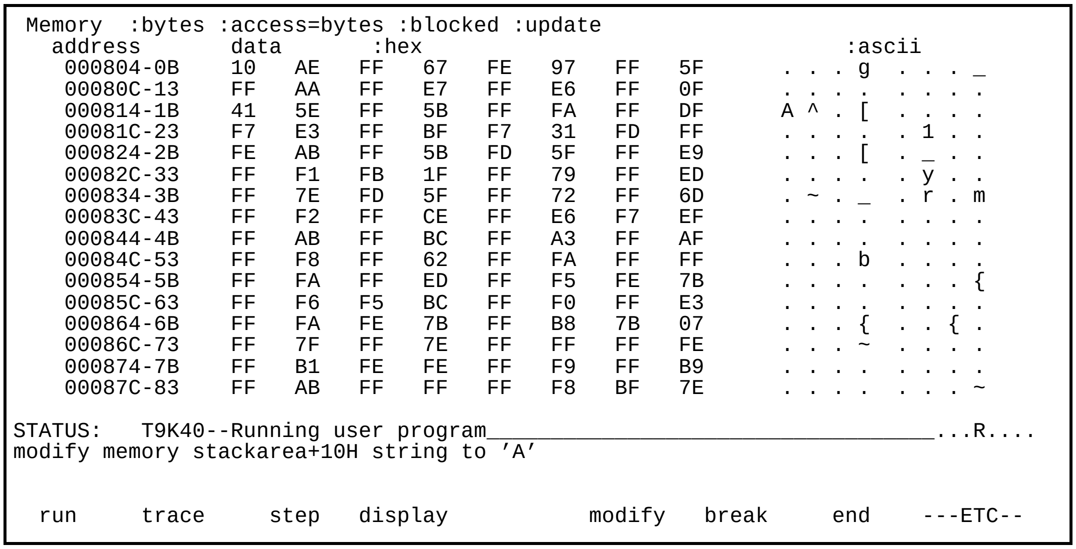
```
 Memory  :bytes :access=bytes :blocked :update
   address       data       :hex                                    :ascii
    000990-97    07   00   03   00   FF   49   03   00      . . . .  . I . .
    000998-9F    BF   55   FB   86   FF   E1   FF   FE      . U . .  . . . .
    0009A0-A7    BF   7E   FD   7C   FF   C0   FF   8B      . ~ . |  . . . .
    0009A8-AF    FF   F4   FF   EF   FF   D7   FB   CD      . . . .  . . . .
    0009B0-B7    F7   79   FF   AB   FF   66   FF   0F      . y . .  . f . .
    0009B8-BF    FF   75   F7   4F   FF   E6   FF   FF      . u . O  . . . .
    0009C0-C7    FF   6C   FF   7F   FF   7F   FF   ED      . l . .  . . . .
    0009C8-CF    FF   F4   BF   F7   DF   E8   FF   CF      . . . .  . . . .
    0009D0-D7    FF   ED   FF   7F   FF   7E   FD   D9      . . . .  . ~ . .
    0009D8-DF    FF   EC   FF   65   FF   FC   FF   29      . . . e  . . . )
    0009E0-E7    FF   FE   F7   A7   FF   FF   FF   1F      . . . .  . . . .
    0009E8-EF    FF   C6   F7   FF   FF   CE   FF   BF      . . . .  . . . .
    0009F0-F7    FF   D0   FF   CF   FF   EC   FF   FF      . . . .  . . . .
    0009F8-FF    FF   E1   FF   3F   FF   EA   FF   EF      . . . ?  . . . .
    000A00-07    31   0B   00   08   00   00   39   0B      1 . . .  . . 9 .
    000A08-0F    00   08   00   00   FF   DF   00   7F      . . . .  . . . .

STATUS:   T9K40--Running user program___________________________________...R....
display memory data thru +7fh blocked bytes


  run     trace     step   display          modify   break     end    ---ETC--
```

Modifying Memory

You can use the modify memory command to send commands to the sample program. Memory locations **stackarea** and **stackarea+10h** correspond to memory address 804 hex and 814 hex respectively. For example, to enter the ’10h’ at address 804 and enter ’A’ at address 814 : use the following commands.

```
display memory stackarea <RETURN>
modify memory stackarea to 10h <RETURN>
modify memory stackarea+10h string to 'A'
<RETURN>
```

After the memory location are modified, the memory display shows the following



```
 Memory  :bytes :access=bytes :blocked :update
   address       data       :hex                                  :ascii
    000804-0B    10   AE   FF   67   FE   97   FF   5F      . . . g  . . . _
    00080C-13    FF   AA   FF   E7   FF   E6   FF   0F      . . . .  . . . .
    000814-1B    41   5E   FF   5B   FF   FA   FF   DF      A ^ . [  . . . .
    00081C-23    F7   E3   FF   BF   F7   31   FD   FF      . . . .  . 1 . .
    000824-2B    FE   AB   FF   5B   FD   5F   FF   E9      . . . [  . _ . .
    00082C-33    FF   F1   FB   1F   FF   79   FF   ED      . . . .  . y . .
    000834-3B    FF   7E   FD   5F   FF   72   FF   6D      . ~ . _  . r . m
    00083C-43    FF   F2   FF   CE   FF   E6   F7   EF      . . . .  . . . .
    000844-4B    FF   AB   FF   BC   FF   A3   FF   AF      . . . .  . . . .
    00084C-53    FF   F8   FF   62   FF   FA   FF   FF      . . . b  . . . .
    000854-5B    FF   FA   FF   ED   FF   F5   FE   7B      . . . .  . . . {
    00085C-63    FF   F6   F5   BC   FF   F0   FF   E3      . . . .  . . . .
    000864-6B    FF   FA   FE   7B   FF   B8   7B   07      . . . {  . . { .
    00086C-73    FF   7F   FF   7E   FF   FF   FF   FE      . . . ~  . . . .
    000874-7B    FF   B1   FE   FE   FF   F9   FF   B9      . . . .  . . . .
    00087C-83    FF   AB   FF   FF   FF   F8   BF   7E      . . . .  . . . ~

STATUS:   T9K40--Running user program___________________________________...R....
modify memory stackarea+10H string to 'A'


  run     trace     step   display           modify   break     end    ---ETC--
```

Breaking into the Monitor

The "break" command allows you to divert emulator execution from the user program to the monitor. You can continue user program execution with the "run" command. To break emulator execution from the demo program to the monitor, enter the following command.

break <RETURN>

Notice that the current address is pointed out with inverse video in displaying memory when the execution breaks to the monitor.

Using Software Breakpoints

Software breakpoints are handled by the TLCS-9000 undefined instruction (breakpoint interrupt instruction:7F9Fh). When you define or enable a software breakpoint, the emulator will replace the opcode at the software breakpoint address with a breakpoint interrupt instruction.

Caution

Software breakpoints should not be set, cleared, enabled, or disabled while the emulator is running user code. If any of these commands are entered while the emulator is running user code and the emulator is executing code in the area where the breakpoint is being modified, program execution may be unreliable.

Note

You must only set software breakpoints at memory locations which contain instruction opcodes (not operands or data). If a software breakpoint is set at a memory location which is not an instruction opcode, the software breakpoint instruction will never be executed. Further, your program won’t work correctly.

Note

NMI will be ignored, when software breakpoint and NMI occur at the same time.

Note

Because software breakpoints are implemented by replacing opcodes with the breakpoint interrupt instruction, you cannot define software breakpoints in target ROM. Them you can use software breakpoints.

When software breakpoints are enabled and the emulator detects the breakpoint interrupt instruction, it generates a break into the monitor. Since the system controller knows the locations of defined software breakpoints, it can determine whether the breakpoint instruction in your target program.

If the breakpoint interrupt was generated by a software breakpoint, execution breaks to the monitor, and the breakpoint interrupt instruction is replaced by original opcode. A subsequent run or step command will execute from this address.

If the breakpoint interrupt was generated by a undefined instruction(7F9Fh) in the target program, execution still breaks to the monitor, and an "undefined breakpoint" status message is displayed. To continue program execution, you must run or step from the target program's breakpoint interrupt vector address.

Enabling/Disabling Software Breakpoints

When you initially enter the Softkey Interface, software breakpoints are disabled. To enable the software breakpoints feature, enter the following command.

```
modify software_breakpoints enable <RETURN>
```

When software breakpoints are enabled and you set a software breakpoint, the TLCS-9000 breakpoint interrupt instruction (7F9Fh) will be placed at the address specified. When the breakpoint interrupt instruction is executed, program execution will break into the monitor.

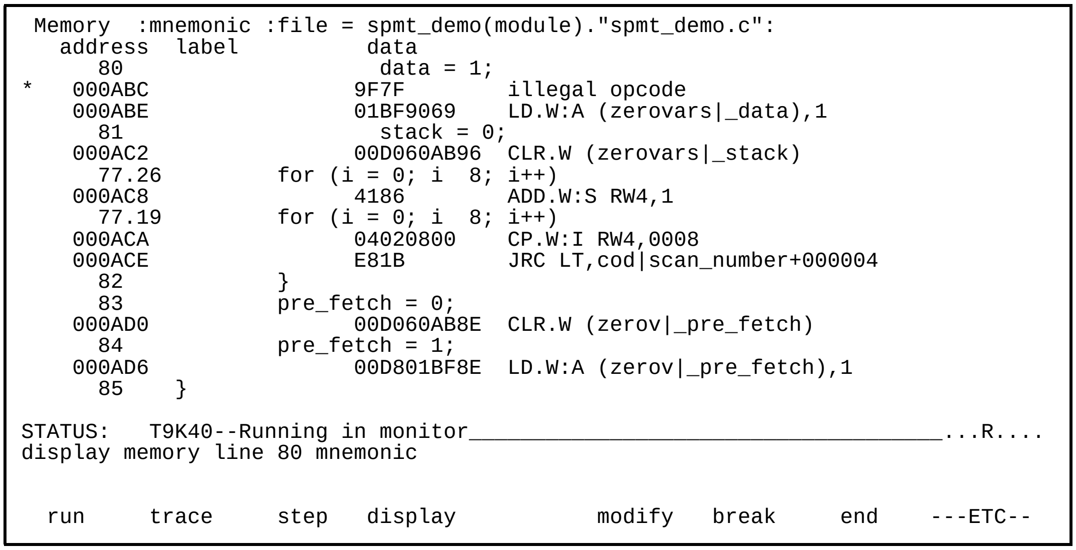
Setting a Software Breakpoint

To set a software breakpoint at line 80 of "spmt_demo.c", enter the following command.

```
modify software_breakpoints set line 80
<RETURN>
```

To see the address where the software breakpoint has been set, enter the following command:

```
display memory line 80 mnemonic <RETURN>
set source on inverse_video on <RETURN>
```



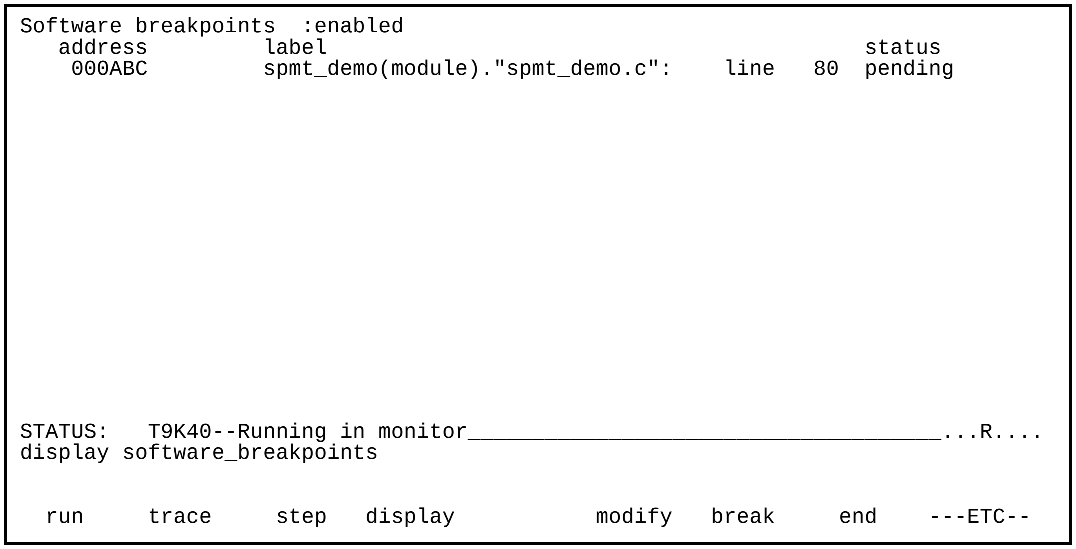
```
 Memory  :mnemonic :file = spmt_demo(module)."spmt_demo.c":
   address  label          data
      80                     data = 1;
*   000ABC                9F7F        illegal opcode
    000ABE                01BF9069    LD.W:A (zerovars|_data),1
      81                     stack = 0;
    000AC2                00D060AB96  CLR.W (zerovars|_stack)
      77.26         for (i = 0; i  8; i++)
    000AC8                4186        ADD.W:S RW4,1
      77.19         for (i = 0; i  8; i++)
    000ACA                04020800    CP.W:I RW4,0008
    000ACE                E81B        JRC LT,cod|scan_number+000004
      82            }
      83            pre_fetch = 0;
    000AD0                00D060AB8E  CLR.W (zerov|_pre_fetch)
      84            pre_fetch = 1;
    000AD6                00D801BF8E  LD.W:A (zerov|_pre_fetch),1
      85    }

STATUS:   T9K40--Running in monitor_____________________________________...R....
display memory line 80 mnemonic


  run     trace     step   display           modify   break     end    ---ETC--
```

The asterisk (*) in left side of the address lists points out that the software breakpoint has been set. The opcode at the software breakpoint address was replaced to the software breakpoint instruction.

Displaying Software Breakpoints

To display software breakpoints, enter the following command.

display software_breakpoints <RETURN>



```
Software breakpoints  :enabled
   address         label                                                status
    000ABC         spmt_demo(module)."spmt_demo.c":    line   80  pending











STATUS:   T9K40--Running in monitor_____________________________________...R....
display software_breakpoints


  run     trace     step   display           modify   break     end    ---ETC--
```

The software breakpoints display shows that the breakpoint is pending. When breakpoints are hit they become inactivated. To reactivate the breakpoint so that is "pending", you must re-enter the "modify software_breakpoints set" command.

After the software breakpoint has been set, enter the following command to cause the emulator to continue executing the demo program.

run `<RETURN>`

A message on the status line shows that the software breakpoint has been hit. The status line also shows that the emulator is now executing in the monitor.

The software breakpoint address is pointed out with inverse video in displaying memory in mnemonic format. To see the software breakpoint with memory, enter the following command.

display memory `line 80` ***mnemonic*** `<RETURN>`

Notice that the original opcode was replaced at the address that the software breakpoint has been set.

Clearing a Software Breakpoint

To remove software breakpoint defined above, enter the following command.

modify software_breakpoints clear line 80 <RETURN>

The breakpoint is removed from the list, and the original opcode is restored if the breakpoint was pending.

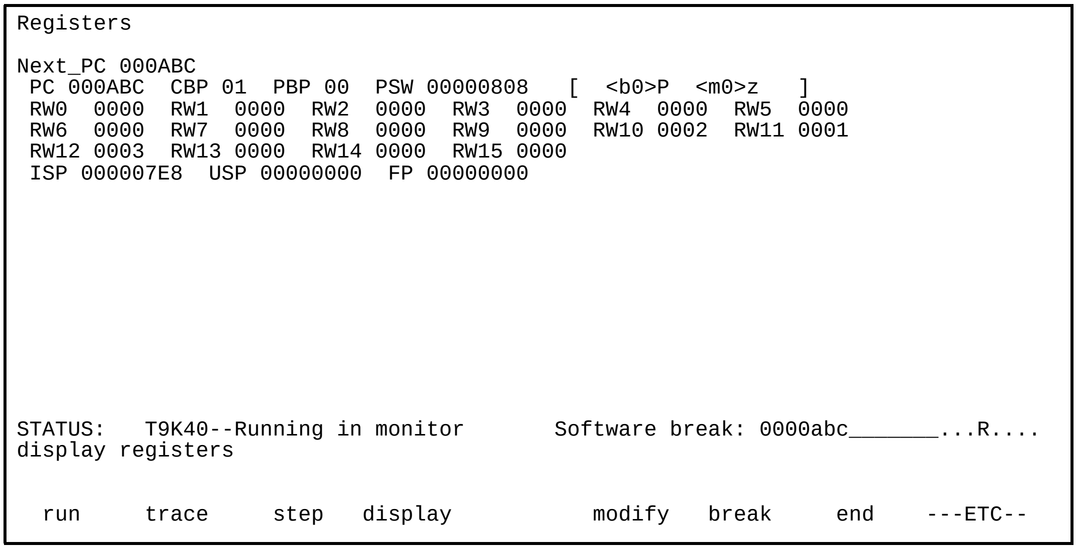
To clear all software breakpoints, you can enter the following command.

modify software_breakpoints clear <RETURN>

Displaying Registers

Enter the following command to display registers. You can display the basic registers, or an individual register. Refer to "REGISTER CLASS and NAME" section in "Using the Emulator" chapter .

display registers <RETURN>



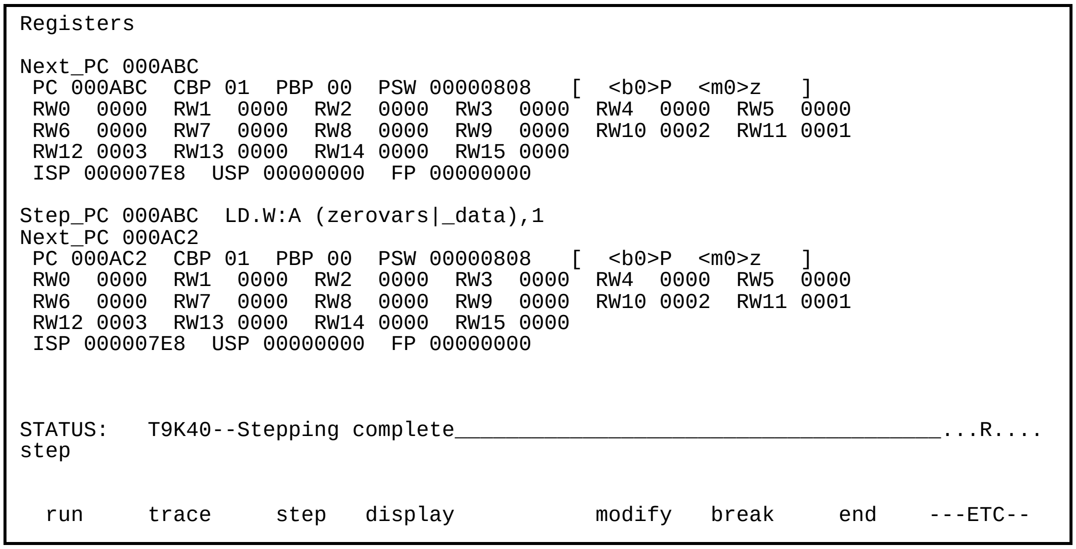
```
Registers

Next_PC 000ABC
 PC 000ABC  CBP 01  PBP 00  PSW 00000808   [  <b0>P  <m0>z   ]
 RW0  0000  RW1  0000  RW2  0000  RW3  0000  RW4  0000  RW5  0000
 RW6  0000  RW7  0000  RW8  0000  RW9  0000  RW10 0002  RW11 0001
 RW12 0003  RW13 0000  RW14 0000  RW15 0000
 ISP 000007E8  USP 00000000  FP 00000000










STATUS:   T9K40--Running in monitor       Software break: 0000abc_______...R....
display registers


  run     trace     step   display          modify   break     end    ---ETC--
```

Stepping Through the Program

The step command allows you to step through program execution an instruction or a number of instructions at a time. Also, you can step from the current program counter or from a specific address. To step through the example program from the address of the software breakpoint set earlier, enter the following command.

step <RETURN>, <RETURN>, <RETURN>, ...

You will see the inverse-video moves according to the step execution. You can continue to step through the program just by pressing the <RETURN> key.



```
Registers

Next_PC 000ABC
 PC 000ABC  CBP 01  PBP 00  PSW 00000808   [  <b0>P  <m0>z   ]
 RW0  0000  RW1  0000  RW2  0000  RW3  0000  RW4  0000  RW5  0000
 RW6  0000  RW7  0000  RW8  0000  RW9  0000  RW10 0002  RW11 0001
 RW12 0003  RW13 0000  RW14 0000  RW15 0000
 ISP 000007E8  USP 00000000  FP 00000000

Step_PC 000ABC  LD.W:A (zerovars|_data),1
Next_PC 000AC2
 PC 000AC2  CBP 01  PBP 00  PSW 00000808   [  <b0>P  <m0>z   ]
 RW0  0000  RW1  0000  RW2  0000  RW3  0000  RW4  0000  RW5  0000
 RW6  0000  RW7  0000  RW8  0000  RW9  0000  RW10 0002  RW11 0001
 RW12 0003  RW13 0000  RW14 0000  RW15 0000
 ISP 000007E8  USP 00000000  FP 00000000



STATUS:   T9K40--Stepping complete______________________________________...R....
step


  run     trace     step   display          modify   break     end     ---ETC--
```

You can step program execution by source lines, enter:

step source <RETURN>

Source line stepping is implemented by single stepping assembly instructions until the next PC is outside of the address range of the current source line. When source line stepping is attempted on assembly code, stepping will complete when a source line is found. To terminate stepping type <Ctrl>-C.

Using the Analyzer

HP 64700 emulators contain an emulation analyzer. The emulation analyzer monitors the internal emulation lines (address, data, and status).

Source Line Referencing

A trace may be taken and displayed using source line referencing. Also, lines of the source program can be displayed with the trace list where the trace occurred.

To display the trace with source code in inverse video, enter the following command:

```
set source on inverse_video on <RETURN>
```

Specifying a Simple Trigger

Suppose you want you trace program execution after the point at address **semantic_check**. The following command make this trace specification.

```
trace after semantic_check <RETURN>
```

The STATUS message shows "Emulation trace started.".

Enter the following command to cause sample program execution to continue from the current program counter.

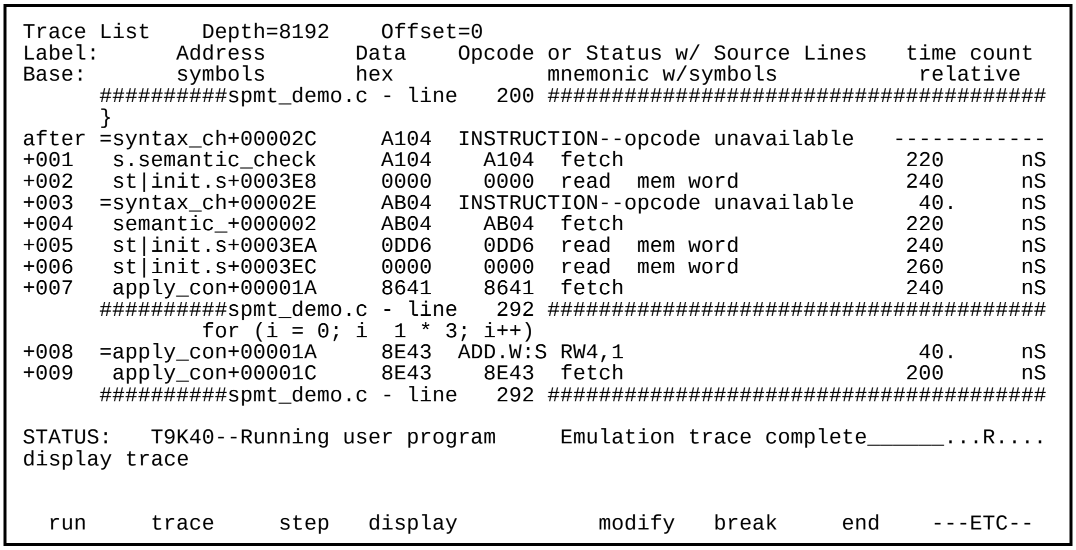
```
run <RETURN>
```

The STATUS message shows "Emulation trace complete.".

Display the Trace

The trace listings which following are of program execution on the TLCS-9000 emulator. To see the trace list, enter the following command:

display trace <RETURN>



```
Trace List    Depth=8192    Offset=0
Label:      Address       Data    Opcode or Status w/ Source Lines   time count
Base:       symbols       hex           mnemonic w/symbols          relative
      ##########spmt_demo.c - line   200 #######################################
      }
after =syntax_ch+00002C     A104  INSTRUCTION--opcode unavailable   ------------
+001   s.semantic_check     A104    A104  fetch                      220      nS
+002   st|init.s+0003E8     0000    0000  read  mem word             240      nS
+003  =syntax_ch+00002E     AB04  INSTRUCTION--opcode unavailable     40.     nS
+004   semantic_+000002     AB04    AB04  fetch                      220      nS
+005   st|init.s+0003EA     0DD6    0DD6  read  mem word             240      nS
+006   st|init.s+0003EC     0000    0000  read  mem word             260      nS
+007   apply_con+00001A     8641    8641  fetch                      240      nS
      ##########spmt_demo.c - line   292 #######################################
              for (i = 0; i  1 * 3; i++)
+008  =apply_con+00001A     8E43  ADD.W:S RW4,1                       40.     nS
+009   apply_con+00001C     8E43    8E43  fetch                      200      nS
      ##########spmt_demo.c - line   292 #######################################

STATUS:   T9K40--Running user program     Emulation trace complete______...R....
display trace


  run     trace     step   display           modify   break     end    ---ETC--
```

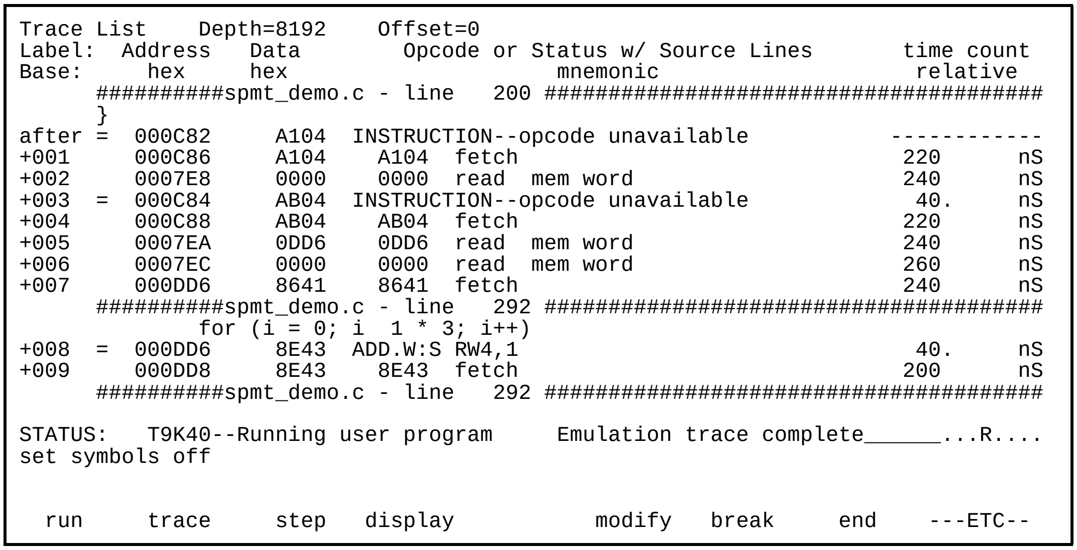
The trace list shows the trace after line (semantic_check()).

To list the next lines of the trace, press the <PGDN> or <NEXT> key.

Displaying Trace with No Symbol

The trace listing shown above has symbol information because of the "**set symbols on**" setting before in this chapter. To see the trace listing with no symbol information, enter the following command.

set symbols off <RETURN>



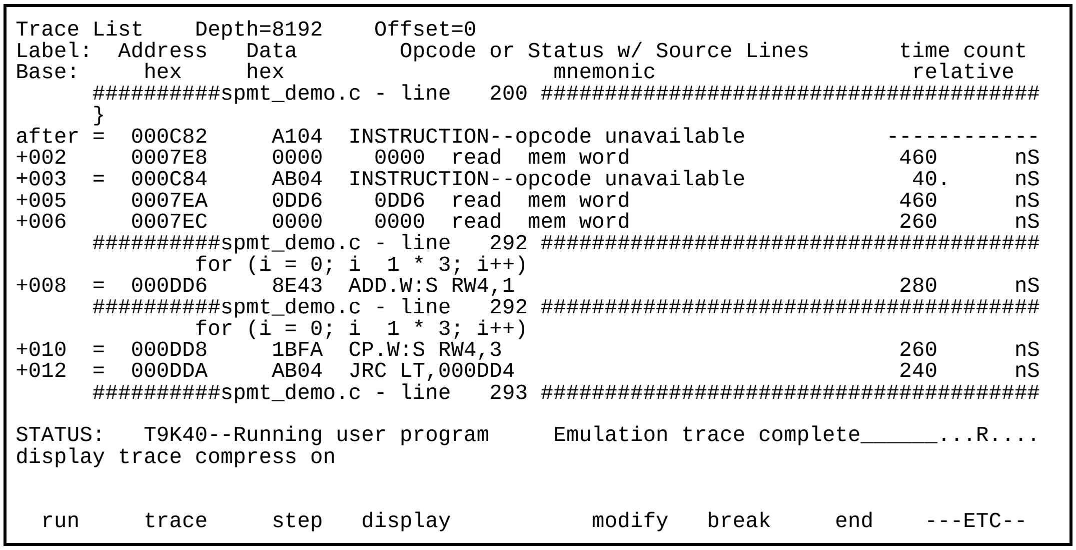
```
Trace List    Depth=8192    Offset=0
Label:  Address   Data        Opcode or Status w/ Source Lines       time count
Base:     hex     hex                     mnemonic                    relative
      ##########spmt_demo.c - line   200 #######################################
      }
after =  000C82     A104  INSTRUCTION--opcode unavailable          ------------
+001     000C86     A104    A104  fetch                              220      nS
+002     0007E8     0000    0000  read  mem word                     240      nS
+003  =  000C84     AB04  INSTRUCTION--opcode unavailable             40.     nS
+004     000C88     AB04    AB04  fetch                              220      nS
+005     0007EA     0DD6    0DD6  read  mem word                     240      nS
+006     0007EC     0000    0000  read  mem word                     260      nS
+007     000DD6     8641    8641  fetch                              240      nS
      ##########spmt_demo.c - line   292 #######################################
              for (i = 0; i  1 * 3; i++)
+008  =  000DD6     8E43  ADD.W:S RW4,1                               40.     nS
+009     000DD8     8E43    8E43  fetch                              200      nS
      ##########spmt_demo.c - line   292 #######################################

STATUS:   T9K40--Running user program     Emulation trace complete______...R....
set symbols off


  run     trace     step   display          modify   break     end    ---ETC--
```

As you can see, the analysis trace display shows the trace list without symbol information.

Displaying Trace with Compress Mode

If you want to see more executed instructions on a display, the TLCS-9000 emulator Softkey Interface provides **compress mode** for analysis display. To see trace display with compress mode, enter the following command:

display trace compress on <RETURN>



```
Trace List    Depth=8192    Offset=0
Label:  Address   Data        Opcode or Status w/ Source Lines       time count
Base:     hex     hex                     mnemonic                    relative
      ##########spmt_demo.c - line   200 #######################################
      }
after =  000C82     A104  INSTRUCTION--opcode unavailable          ------------
+002     0007E8     0000    0000  read  mem word                     460      nS
+003  =  000C84     AB04  INSTRUCTION--opcode unavailable             40.     nS
+005     0007EA     0DD6    0DD6  read  mem word                     460      nS
+006     0007EC     0000    0000  read  mem word                     260      nS
      ##########spmt_demo.c - line   292 #######################################
              for (i = 0; i  1 * 3; i++)
+008  =  000DD6     8E43  ADD.W:S RW4,1                              280      nS
      ##########spmt_demo.c - line   292 #######################################
              for (i = 0; i  1 * 3; i++)
+010  =  000DD8     1BFA  CP.W:S RW4,3                               260      nS
+012  =  000DDA     AB04  JRC LT,000DD4                              240      nS
      ##########spmt_demo.c - line   293 #######################################

STATUS:   T9K40--Running user program     Emulation trace complete______...R....
display trace compress on


  run     trace     step   display           modify   break     end    ---ETC--
```

As you can see, the analysis trace display shows the analysis trace lists without fetch cycles. With this command you can examine program execution easily.

If you want to see all of cycles including fetch cycles, enter following command:

display trace compress off <RETURN>

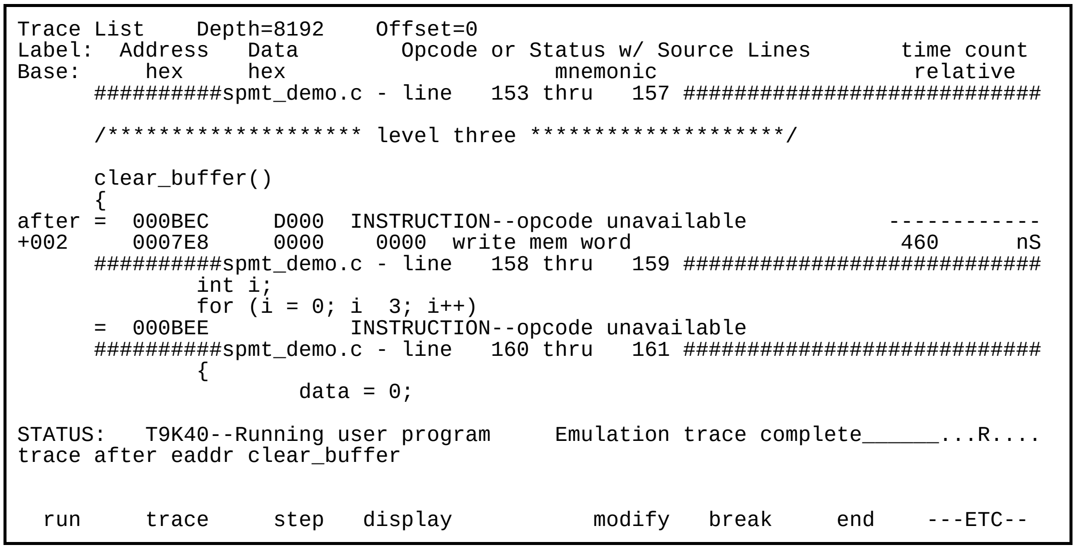
The trace display shows you all of the cycles the emulation analyzer have captured.

Trigger the Analyzer at an Instruction Execution State

The emulator analyzer can capture states of instruction execution. If you want to trigger the analyzer when an instruction at a desired address is executed, you should not set up the analyzer trigger condition to detect the address. If you do so, the analyzer will be also triggered in case that the address is accessed to fetch the instruction, or read the data from address. You should use the execution address(eaddr) qualifier.Suppose that you want to trace the states of the execution after the instruction at *clear_buffer*of the *spmt_demo.c* file, enter the following command.

```
trace after eaddr clear_buffer <RETURN>
```

The message "Emulation trace started" will appear on the status line, and the status line now shows "Emulation trace complete".



```
Trace List    Depth=8192    Offset=0
Label:  Address   Data        Opcode or Status w/ Source Lines       time count
Base:     hex     hex                     mnemonic                    relative
      ##########spmt_demo.c - line   153 thru   157 ############################

      /******************** level three ********************/

      clear_buffer()
      {
after =  000BEC     D000  INSTRUCTION--opcode unavailable           ------------
+002     0007E8     0000    0000  write mem word                     460      nS
      ##########spmt_demo.c - line   158 thru   159 ############################
              int i;
              for (i = 0; i  3; i++)
      =  000BEE           INSTRUCTION--opcode unavailable
      ##########spmt_demo.c - line   160 thru   161 ############################
              {
                      data = 0;

STATUS:   T9K40--Running user program     Emulation trace complete______...R....
trace after eaddr clear_buffer


  run     trace     step   display           modify   break     end    ---ETC--
```

The emulator has disassemble capability in trace listing. When the emulator disassembles instructions in stored trace information, the fetch cycles of each instruction are required. When you displayed the results of analyzer trace, some lines which include "INSTRUCTION--opcode unavailable" message may be displayed. Each line is instruction execution cycle at the address in the left side of the displayed because the fetch states for the instructions were not stored by the analyzer.

To display complete disassembles in the trace listing, you should modify location of trigger state in trace list, referred to as the "trigger position", to "**about**" instead of "**after**".

Displaying trace option

You can specify whether the emulator display only bus cycles, or only execution cycles, or both cycles. To specify, the TLCS-9000 emulator Softkey Interface provides display trace option. To display only bus cycles, enter the following command:

```
display trace mnemonic option
bus_cycles_only <RETURN>
```

If you want to display only execution cycles, enter the following command:

```
display trace mnemonic option
exec_cycles_only <RETURN>
```

If you want to display bus cycles and execution cycles, enter the following command:

```
display trace mnemonic option both_cycles
<RETURN>
```

Emulator Analysis Status Qualifiers

The following analysis status qualifiers may also be used with the TLCS-9000 emulator.

Qualifier	Status bits	Description
bus	0x0xxxxxxxxxxxy	bus cycle
byte	0x010xxxxxx1xxy	byte memory cycle
exec	00xxx1xxxxxxxxy	execute instruction
fetch	0x010x1xxxxx1xy	program fetch
halt	0x011xxxxxxxxxy	halt
intack	0x000xxxxxxxxxy	interrupt acknowledge
monitor	0x0xxxxxxxxxx0y	monitor cycle
read	0x010xxxxxxx1xy	read
user	0x0xxxxxxxxxx1y	user program cycle
word	0x010xxxxxx0xxy	word memory cycle
write	0x010x0xxxxx0xy	write

For a Complete Description

For a complete description of using the HP 64700 Series analyzer with the Softkey Interface, refer to the *Analyzer Softkey Interface User's Guide*.

Resetting the Emulator

To reset the emulator, enter the following command.

```
reset <RETURN>
```

Exiting the Softkey Interface

There are several options available when exiting the Softkey Interface: exiting and releasing the emulation system, exiting with the intent of re-entering (continuing), exiting locked from multiple emulation windows, and exiting (locked) and selecting the measurement system display or another module.

End Release System

To exit the Softkey Interface, releasing the emulator so that other users may use the emulator, enter the following command.

```
end release_system <RETURN>
```

Ending to Continue Later

You may also exit the Softkey Interface without specifying any options; this causes the emulator to be locked. When the emulator is locked, other users are prevented from using it and the emulator configuration is saved so that it can be restored the next time you enter (continue) the Softkey Interface.

```
end <RETURN>
```

Ending Locked from All Windows

When using the Softkey Interface from within window systems, the "end" command with no options causes an exit only in that window. To end locked from all windows, enter the following command.

```
end locked <RETURN>
```

This option only appears when you enter the Softkey Interface via the **emul700** command.

Refer to the *Softkey Interface Reference* manual for more information on using the Softkey Interface with window systems.

Notes

3

In-Circuit Emulation Topics

Introduction

Many of the topics described in this chapter involve the installation, and the commands which relate to using the emulator in-circuit, that is, connected to a target system or demo target board.

This chapter will:

- Show you how to install the emulation probe cable
- Show you how to install the emulation memory module.
- Show you how to install the emulation probe to demo target board.
- Describe the issues concerning the installation of the emulation probe into target systems.
- Describe how to execute program from target reset. This topics is related to program execution in general.
- Describe how to use software breakpoints with ROMed code, and how to test patches to ROMed code. These topics relate to the debugging of target system ROM.

Prerequisites

Before performing the tasks described in this chapter, you should be familiar with how the emulator operates in general. Refer to the *Concepts of Emulation and Analysis* manual and the "Getting Started" chapter of this manual.

Installing the Emulation Probe Cable

The probe cables consist of three ribbon cables. The longest cable connects to J3 of the emulation control card, and to J3 of the probe. The shortest cable connects to J1 of the emulation control card and J1 of the probe. The ribbon cables are held in place on the emulation control card by a cable clamp attached with two screws. No clamp holds the ribbon cables in the probe.

1. Secure the cable on the emulation control card with cable clamp and two screws.

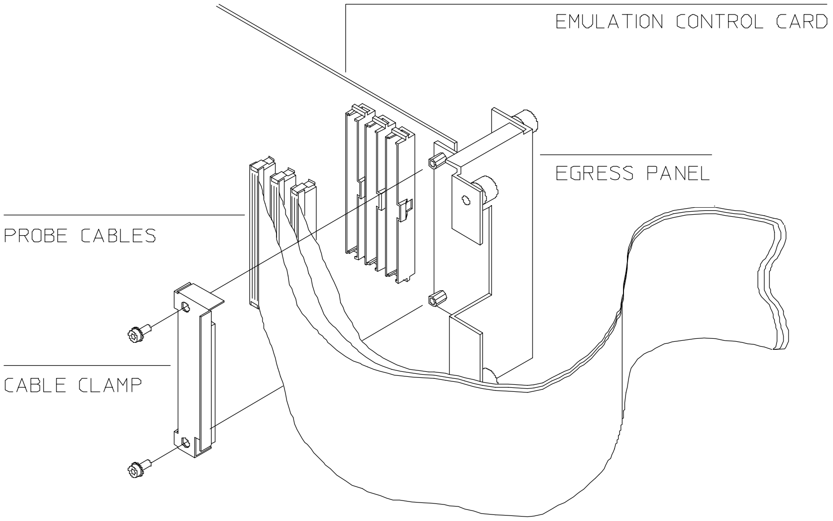


Figure 3-1 Installing cables to the control board

2. When insert the ribbon cables into the appropriate sockets, press inward on the connector clops so that they into the sockets as shown.

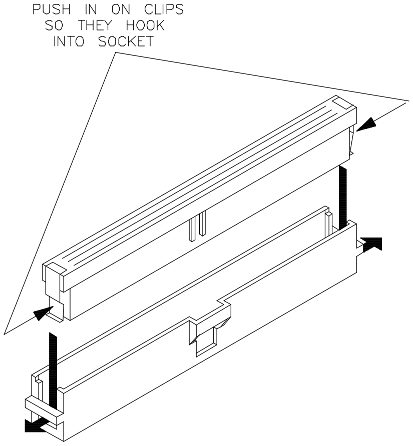


Figure 3-2 Installing cables into cable sockets

3. Connect the other ends of the cable s to the emulation probe.

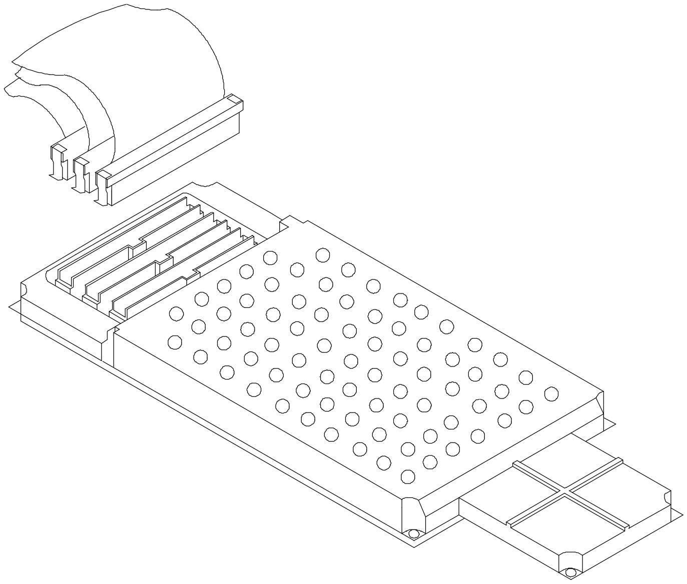


Figure 3-3 Installing cables to the emulation probe

Installing the Emulation Memory Module

There are four types of emulation memory modules that can be inserted into sockets on the probe.

1. Remove plastic rivets that secure the plastic cover on the top of the emulation probe, and remove the cover. The bottom cover is only removed when you need to replace a defective active probe on the exchange program.

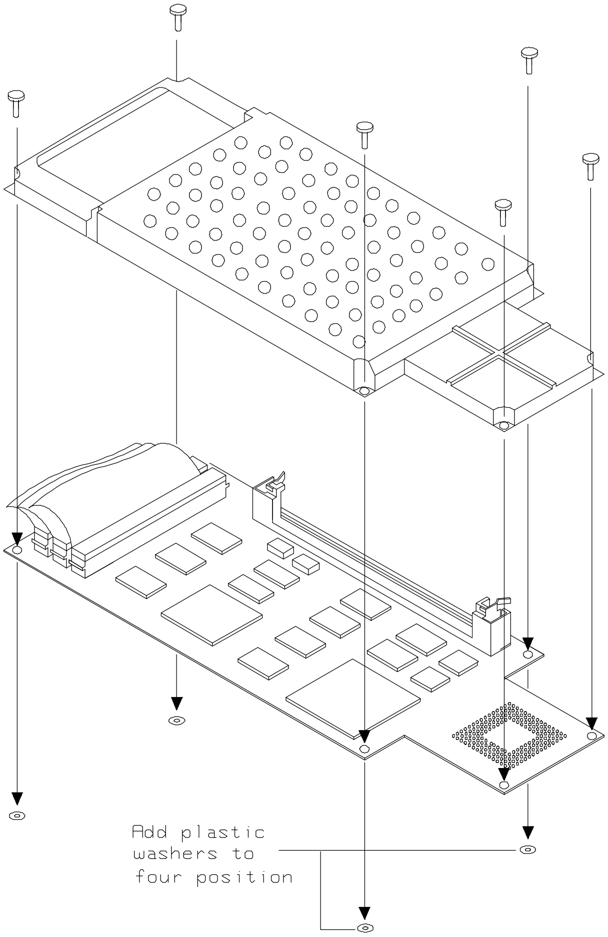


Figure 3-4 Opening the emulation probe cover

2. Insert emulation memory module on the emulation probe. There is a cutout on one side of the memory modules so that they can only be installed one way.

   To install memory modules, place the memory module into the socket groove at an angle. Firmly press the memory module into the socket to make sure it is completely seated. Once the memory module is seated in the connector groove, pull the memory module forward so that the notches on the socket fit into the holes on the memory module. There are two latches on the sides of the socket that hold the memory module in place.

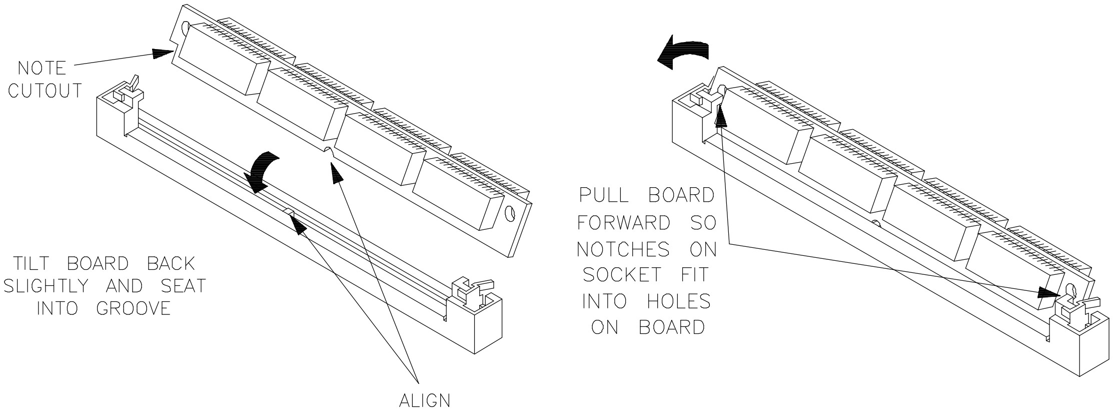

NOTE CUTOUT
TILT BOARD BACK SLIGHTLY AND SEAT INTO GROOVE
ALIGN
PULL BOARD FORWARD SO NOTCHES ON SOCKET FIT INTO HOLES ON BOARD

Figure 3-5 Installing the memory module

3. Replace the plastic cover, and insert new plastic rivets to secure the cover.

Installing into the Demo Target Board

To connect the microprocessor connector to the demo target board, proceeded with the following instructions.

1. Remove front bezel and connect the power cable to the connector of the HP 64700A front panel. Refer to the *HP 64700 Series Installation/Service* manual.

2. With HP 64700A power OFF, connect the emulation probe to the demo target board. When you install the emulation probe into the demo target board, be careful not to bend any of the pins.

   After connection the probe to the demo target board, set the TEST/TARGET MODE and SINGLE CHIP/EXTERNAL BUS MODE switches. Use TEST MODE position when you run performance verification test, and use TARGET MODE position when you run the emulator in "out-of-circuit" mode. You must set SINGLE CHIP/EXTERNAL BUS switch according to ’Processor operation mode?’ configuration.

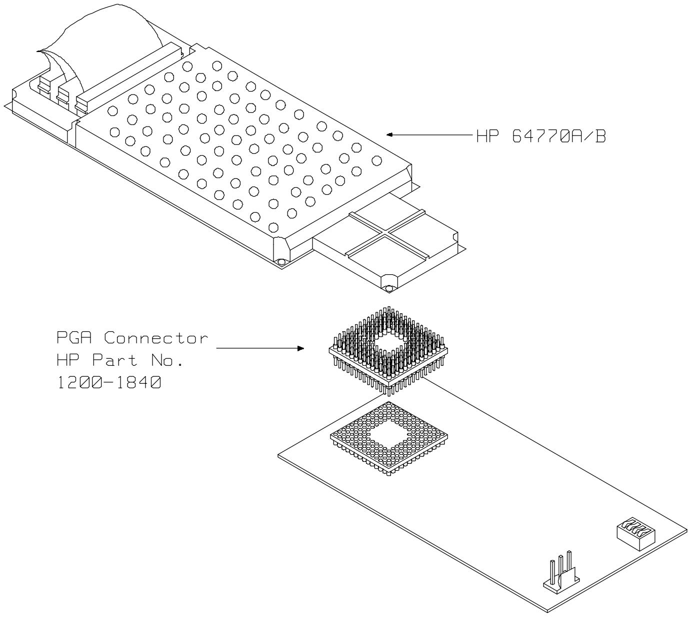


Figure 3-6 Installing the demo target board

3. Connect the power cable supply wires from the emulator to demo target board. When attaching the wire cable to the demo target board, make sure the connector is aligned properly so that all three pins are connected.

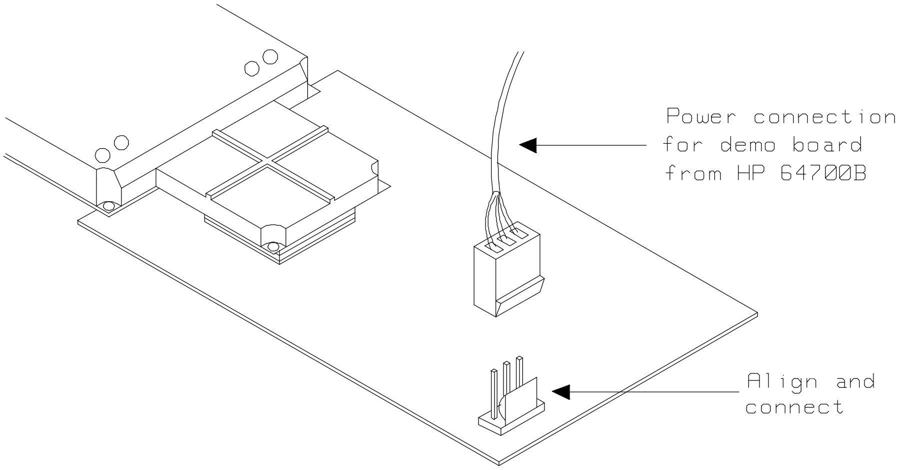


Figure 3-7 Installing the power cable

Installing into a Target System

The TLCS-9000 emulation probe has a 135-pin PGA connector;
The emulation probe is also provided with a conductive pin protector to protect the delicate gold-plated pins of the probe connector from damage due to impact.

Caution

Protect against electrostatic discharge. The emulator probe contains devices that are susceptible to damage by electrostatic discharge. Therefore, precautionary measures should be taken before handling the microprocessor connector attached to the end of the probe cable to avoid damaging the internal components of the probe by electrostatic electricity.

Caution

Make sure target system power is OFF. Do not install the emulation probe into the target system microprocessor socket with power applied to the target system. The emulator may be damaged if target system power is not removed before probe installation.

Caution

Make sure pin 1 of probe connector is aligned with pin 1 of the socket. When installing the emulation probe, be sure that probe is inserted into the processor socket so that pin 1 of the connector aligns with pin 1 of the socket. Damage to the emulator probe will result if the probe is incorrectly installed.

Caution

DO NOT use the microprocessor connector without using a pin protector. The pin protector prevents damage to the prove when inserting and removing the probe from the flexible adapter.

Caution ✋ **Compatibility of VOLTAGE/CURRENCY.** Please be sure to check that the voltage/currency of the emulator and target system being connected are compatible. If there is a discrepancy, damage may result.

Caution ✋ **Do not apply strong force to PGA-QFP probe,** as that might damage the probe.

Caution ✋ **Turn ON power.** When you start to use the 64770A/B emulator which is plugged into a target system, you must turn HP 64770A/B power ON at first, then turn target system power ON.

Caution ✋ **Turn OFF power** Do not turn HP 64770A/B power OFF while the emulator is plugged into a target system whose power is ON.

Installing into a QFP-PGA Adaptor

To connect the microprocessor connector to the target system, proceeded with the following instructions.

1. Attach the QFP socket/adapter(YAMAICHI IC149-120K13207-0B) on your target system.

2. Connect the PGA-QFP probe(64770-61602) to the emulation probe through PGA connector(1200-1840).

3. Install the PGA-QFP probe to the QFP socket/adaptor on your target system.

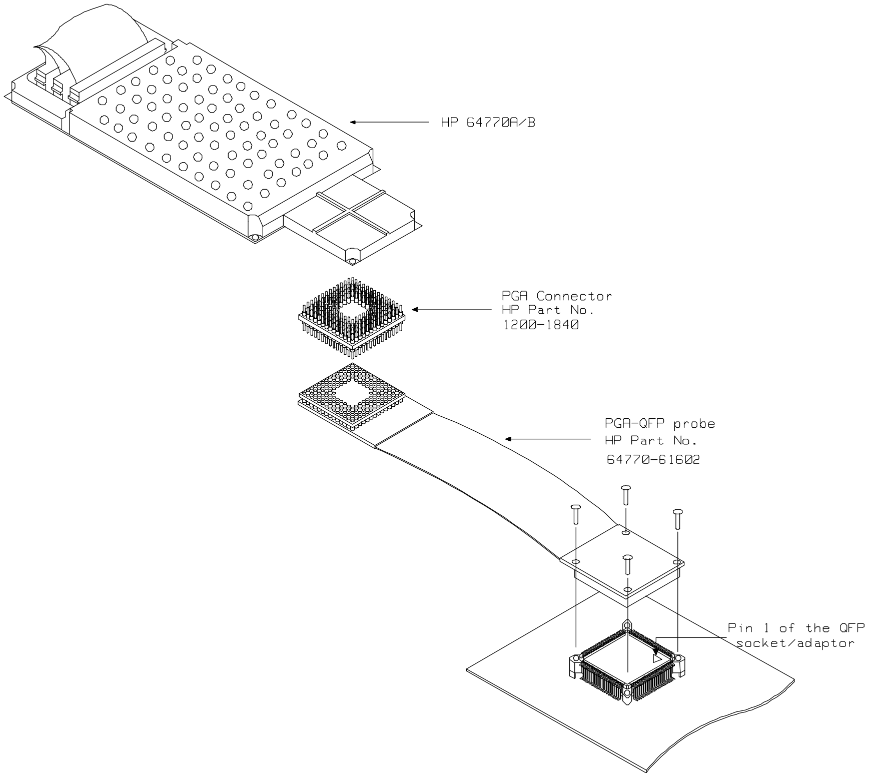


Figure 3-8 Installing into a taget system board

In-Circuit configuration Options

The TLCS-9000 emulation provides configuration options for the following in-circuit emulation issues. Refer to the "Configuring the Emulator" chapter for more information on these configuration option.

Enabling BUSRQ, NMI, RESET and INT0-7(for HP 64770A), IREQ(for HP 64770B) Input from the Target System

You can configure whether the emulator should accept or ignore the BUSRQ, NMI, RESET and INT0-7(for HP 64770A), IREQ(for HP 64770B) signals from the target system.

Running the Emulation from Target Reset

You can specify that the emulator begins execution from target system reset. When the target system RESET line becomes active and then inactive, the emulator will start reset sequence (option) as actual microprocessor.

At first, you must specify the emulator responds to RESET signal by the target system (see the "Enable RESET inputs from target system?" configuration in "Configuring the Emulator" chapter on this manual).

To Specify a run from target system reset, enter the following command:

```
    run from reset <RETURN>
```

The status now shows that the emulator is "Awaiting target reset". After the target system is reset,the status line message will change to show the appropriate emulator status.

Note

In the "Awaiting target reset" status, you can not break into the monitor. If you enter "run from reset" in the configuration that emulator ignores target system reset, you must reset the emulator.

Note

After you turn on the emulator, you must enter "reset" command and then "break" command to set the emulation stack pointer.

The TLCS-9000 emulator supports power on reset. If you want program to be executed by power on reset, execute the following process.

1) Enter "reset"

2) Enter "break"

3) Enter "run from reset"

4) Turn OFF your target system

4-1) If you see the p> system prompt, enter "run from reset" again.

5) Turn On your target system

Pin State in Background

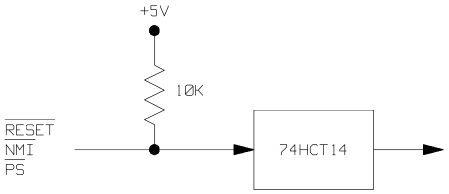
While the emulator is running in the monitor, the probe pins of the emulator are in the following state.

Address Bus	Same as running user’s program.
Data Bus	Same as running user’s program.
$\overline{\text{BS}}$ R/$\overline{\text{W}}$	Same as running user’s program.
$\overline{\text{UB}}$/$\overline{\text{WEH}}$ $\overline{\text{LB}}$/$\overline{\text{WEL}}$ $\overline{\text{CAS}}$/$\overline{\text{OE}}$ $\overline{\text{RAS0}}$/$\overline{\text{CE0}}$ $\overline{\text{RAS1}}$/$\overline{\text{CE1}}$ $\overline{\text{RAS2}}$/$\overline{\text{CE2}}$ $\overline{\text{RAS3}}$/$\overline{\text{CE3}}$	Same as running user’s program except accessing monitor area. When accessing monitor area, High level.
$\overline{\text{RFSH}}$/CE	Same as running user’s program except accessing monitor area. When accessing monitor area, Low level.

Target System Interface

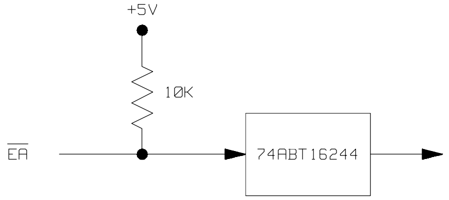
$\overline{\text{RESET}}$
$\overline{\text{PS}}$
$\overline{\text{NMI}}$

These signals are connected to 74HC14 through 10K ohm pull-up register.



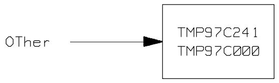
$\overline{\text{EA}}$

These signals are connected to 74ABT16244 through 10K ohm pull-up register.



Other signals

These signals are connected to TLCS-9000 emulation processor.



Notes

4

Configuring the Emulator

Introduction

Your TLCS-9000 emulator can be used in all stages of target system development. For instance, you can run the emulator out-of-circuit when developing target system software, or you can use the emulator in-circuit when integrating software with target system hardware. Emulation memory can be used in place of, or along with, target system memory. You can use the emulator’s internal clock or the target system clock. You can execute target programs in real-time or allow emulator execution to be diverted into the monitor when commands request access of target system resources (target system memory, register contents, etc.)

The emulator is a flexible instrument and it may be configured to suit your needs at any stage of the development process. This chapter describes the options available when configuring the TLCS-9000 emulator.

The configuration options are accessed with the following command.

```
    modify configuration <RETURN>
```

After entering the command above, you will be asked questions regarding the emulator configuration. The configuration questions are listed below and grouped into the following classes.

General Emulator Configuration:

- Restricting to real-time execution.
- Selecting processor type.
- Specifying processor operation mode.
- Specifying value of the monitor base address.
- Enabling emulation Vector Base Pointer.
- Specifying value of Vector base address.
- Specifying initial Current Bank Pointer value.

Memory Configuration:

- Mapping memory.

Emulator Pod Configuration:

- Enabling watch dog timer.
- Enabling BUSRQ input from target system.
- Enabling RESET input from target system.
- Enabling interrupt requests.
- Selecting target memory access size.

Debug/Trace Configuration:

- Enabling breaks on writes to ROM.
- Specifying tracing of user program/emulation monitor cycles.
- Selecting emulation analyzer speed.

Simulated I/O Configuration: Simulated I/O is described in the *Simulated I/O* reference manual.

Interactive Measurement Configuration: See the chapter on coordinated measurements in the *Softkey Interface Reference* manual.

General Emulator Configuration

The configuration questions described in this section involve general emulator operation.

Restrict to Real-Time Runs?

This configuration allows to you specify whether program execution should take place in real-time or whether commands should be allowed to cause breaks to the monitor during program execution.

no All commands, regardless of whether or not they require a break to the emulation monitor, are accepted by the emulator.

yes When runs are restricted to real-time and the emulator is running the user program, all commands that cause a break (except "reset", "break", "run", and "step") are refused. For example, the following commands are not allowed when runs are restricted to real-time:

- Display/modify registers.
- Display/modify memory.

Caution

If your target system circuitry is dependent on constant execution of program code, you should restrict the emulator to real-time runs. This will help insure that target system damage does not occur. However, remember that you can still execute the "run", "reset", "break", and "step" commands; you should use caution in executing these commands.

Processor type? (HP 64770A)

This question allows you to select which microprocessor to be emulated.

97PS40 The TMP97PS40F, TMP97CS40F, and TMP97C241F microprocessors are emulated. When you emulate TMP97C241F microprocessor, you must specify "Processor operation mode? external_bus".

97CM40 The TMP97CM40F microprocessor is emulated.

97PW40 The TMP97PW40F, TMP97CW40F microprocessors are emulated.

NONE no valid processor is selected. This is a power up default and can not break into monitor from reset until valid processor is selected.

Processor type? (HP 64770B)

This question allows you to select which microprocessor to be emulated.

97CS42 The TMP97CS42, TMP97PU42(64K mode) microprocessors is emulated.

97CU42 The TMP97CU42, TMP97PU42(96K mode) microprocessor are emulated.

97CW42 The TMP97PW42, TMP97CW42 microprocessors are emulated.

NONE no valid processor is selected. This is a power up default and can not break into monitor from reset until valid processor is selected.

Note

You must specify processor type before operation the emulator, Otherwise, you can not operate the emulator correctly.

Note

Changing this configuration setting will drive the emulator into a reset state and will reset the memory mapping. Monitor address and vector address configurations will be set to their default.

Processor operation mode?

This configuration allows to you specify whether operation mode is single chip mode or external bus mode.

single The emulator will operate in single chip mode.

external_bus The emulator will operate in external bus mode.

Note

TLCS-9000 emulator operates in accordance with this configuration instead of $\overline{\text{EA}}$ signal from target system.

But when the emulator breaks into the monitor from reset state, $\overline{\text{EA}}$ signal must accord with this configuration.

Note

Changing this configuration setting will drive the emulator into a reset state and will reset the memory mapping. Monitor address and vector address configurations will be set to their default.

Monitor base address?

This configuration allows you to specify the range of addresses that the monitor uses. The emulation monitor occupies 64K byte address space and the address of the monitor must be located on a 64K boundary. Valid address range is from 10000H through 0EF0000H.

Note

Changing this configuration setting will drive the emulator into a reset state and will reset the memory mapping. The vector address configuration will set to its default.

Enable emulation VBP?

This configuration allows you to specify whether or not the emulation VBP is used.

yes	The emulator supplies VBP value which determines the base address of the vector address. The emulator automatically initializes necessary vector entry to perform emulation tasks, emulation break, single stepping, and software breakpoint breaks.
no	VBP value is read from target system. The emulator does not do initializations for the vector entries to perform emulation tasks.

Note

Changing this configuration setting will drive the emulator into a reset state.

Vector base address?

This configuration allows you to specify the value for the VBP (Vector Base Pointer) to be calculated. Because this configuration is used whenever the emulator breaks into the monitor regardless "Enable emulation VBP?" configuration, you must specify address which accord with vector address.

Note

Changing this configuration setting will drive the emulator into a reset state and will reset the memory mapping.

Initial CBP value?

This configuration allows you to specify the value of CBP (Current Bank Pointer) when the emulator breaks into the monitor from reset state. When emulation VBP is enabled and first 256 byte of vector area is mapped as emulation ROM/RAM, this configuration is ignored and CBP is initialized along with the vector entry.

Memory Configuration

The memory configuration questions allows you to select the monitor type, to select the location of the monitor, and to map memory. To access the memory configuration questions, you must answer "yes" to the following question.

Modify memory configuration?

Mapping Memory

The emulation memory consists of 256k, 1M, or 4Mbytes. You can define up to 7 memory range.

The memory mapper allows you to characterize memory locations. It allows you specify whether a certain range of memory is present in the target system or whether you will be using emulation memory for that address range. You can also specify whether the target system memory is ROM or RAM, and you can specify that emulation memory be treated as ROM or RAM.

The internal RAM area(if you select the single chip mode, also internal ROM area) and emulation monitor area are mapped automatically. And you can not delete these map terms. External I/O area can not be mapped as emulation memory.

When you characterize memory ranges as emulation memory, note the following.

- When you characterize memory range which does not override 64K byte boundary as emulation memory, 64K byte is used.

  For example, when you characterize memory rang(1000h thru 010ffh), 64K byte of emulation memory is used.

- When you characterize memory range which override N block of 64K byte as emulation memory, 64K x 2^M (2^{M-1} < N =< 2^M) byte is used.

  For example, when you characterize memory range(0ff00h thru 200ffh) which overrides 3 block of 64K byte as emulation RAM, the 64K x 2^2 (2^1 < 3 =< 2^2:M=2) byte of emulation memory is used.

  For examle, when 192K byte emulation memroy is remained you can not characterize memory range(80000h thru 0affffh), which is 192K byte and override 3 block of 64K byte, as emulatoin RAM by one mapper term because the emulator needs 256K byte to map memory range(80000h thru 0affffh). In this case, you can characterize that memeoy range by two mapper term, the one is 128K byte(80000h thru 9ffffh) mapper term, the another is 64K byte(0a0000h thru 0affffh) mapper term.

Note

Target system accesses to emulation memory are not allowed. Target system devices that take control of the bus (for example, DMA controllers) cannot access emulation memory.

Blocks of memory can also be characterized as guarded memory. Guarded memory accesses will generate "break to monitor" requests. Writes to ROM will generate "break to monitor" requests if the "Enable breaks on writes to ROM?" configuration item is enabled (see the "Debug/Trace Configuration" section which follows).

Determining the Locations to be Mapped

Typically, assemblers generate relocatable files and linkers combine relocatable files to form the absolute file. The linker load map listing will show what locations your program will occupy in memory.

Emulator Pod Configuration

To access the emulator pod configuration questions, you must answer "yes" to the following question.

Modify emulator pod configuration?

Enable watch dog timer? (HP 64770A Only)

This question allows you to specify whether the watch dog timer is enabled or disabled when usr's program running.

yes The emulator enables watch dog timer.

no The emulator disables watch dog timer.

Respond to $\overline{\text{BUSRQ}}$ from target system?

This configuration allows you to specify whether or not the emulator accepts $\overline{\text{BUSRQ}}$(Bus Request) signal generated by the target system.

yes The emulator accepts $\overline{\text{BUSRQ}}$ signal. When the $\overline{\text{BUSRQ}}$ is accepted, the emulator will respond as actual microprocessor.

no The emulator ignore $\overline{\text{BUSRQ}}$ signal from target system completely.

Respond to $\overline{\text{RESET}}$ from target system?

The TLCS-9000 emulator can respond or ignore target system reset while running in user program or waiting for target system reset (refer to "run from reset" command in the *Softkey Interface Reference* manual). While running in monitor, the TLCS-9000 emulator ignores target system reset completely independent on this setting.

yes Specify that, this is a default configuration, make the emulator to respond to reset from target system. In this configuration, emulator will accept reset and execute from reset vector as same manner as actual microprocessor after reset is inactivated.

no The emulator ignores reset signal from target system completely, even while in foreground (executing user program).

Note

Changing this configuration option will drive the emulator into a reset state.

Respond to interrupts ?

This question allows you to specify whether or not the emulation processor accepts interrupts.

yes The emulator will respond to interrupt requests from target system(NMI and INT0-3 for HP 64770A, NMI and IREQ for 64770B) and an internal peripheral during user program execution.

no The emulator will always ignore interrupt requests.

Note

When target interrupts signal is enabled, it is in effect while the emulator is running the target program. While the emulator is running monitor, NMI, INT0-7(edge sense) for HP 64770A, IREQ for HP 64770B will be suspended until the emulator goes into user’s program.

Target memory access size

This configuration specifies the type of microprocessor cycles that are used by the monitor program to access target memory or I/O locations. When a command requests the monitor to read or write to target system memory or I/O, the monitor program will look at the access mode setting to determine whether byte or word instructions should be used.

bytes Selecting the byte access mode specifies that the emulator will access target memory using byte cycles (one byte at a time).

words

Selecting the word access mode specifies that the emulator will access target memory using word cycles (one word at a time).

any

Selecting the any access mode specifies that the emulator will access target memory using a display/modify target memory command option. If option "words" is specified, access size will be set to "words". Other target memory commands such as "load" and "store" will use an access size of "bytes".

Debug/Trace Configuration

The debug/trace configuration questions allows you to specify breaks on writes to ROM, enable/disable the software breakpoints feature, and specify that the analyzer trace foreground/background execution. To access the debug/trace configuration questions, you must answer "yes" to the following question.

Modify debug/trace options?

Break Processor on Write to ROM?

This question allows you to specify that the emulator break to the monitor upon attempts to write to memory space mapped as ROM. The emulator will prevent the processor from actually writing to memory mapped as emulation ROM; however, they cannot prevent writes to target system RAM locations which are mapped as ROM, even though the write to ROM break is enabled.

yes

Causes the emulator to break into the emulation monitor whenever the user program attempts to write to a memory region mapped as ROM.

no

The emulator will not break to the monitor upon a write to ROM. The emulator will not modify the memory location if it is in emulation ROM.

Note

The **wrrom** trace command status option allows you to use "write to ROM" cycles as trigger and storage qualifiers. For example, you could use the following command to trace about a write to ROM:

```
    trace about status wrrom <RETURN>
```

Trace monitor or user program operation?

This question allows you to specify whether the analyzer trace only user program emulation processor cycles, only monitor cycles, or both monitor or user program cycles.

user Specifies that the analyzer trace only user program cycles. This option is specified by the default emulator configuration.

monitor Specifies that the analyzer trace only emulation monitor cycles. (This is rarely a useful setting.)

both Specifies that the analyzer trace both user program and emulation monitor cycles. You may wish to specify this option so that all emulation processor cycles may be viewed in the trace display.

Emulation analyzer speed? (HP 64770A Only)

This question allows you specify the emulation processor clock speed. The analyzer capabilities of time and state count are affected by the processor clock speed. If you use 64794A/C/D Deep emulation analyzer, the trace state and time counter qualifiers can be used regardless of clock speed. You must answer this question, when you use HP 64770A emulator with HP 64704A emulation bus analyzer.

slow Specifies the processor clock speed is less than or equal to 16MHz. Both state and time counting are available.

fast Specifies the processor clock speed is greater than 16MHz. Only state counting are available.

Simulated I/O Configuration

The simulated I/O feature and configuration options are described in the *Simulated I/O* reference manual.

Interactive Measurement Configuration

The interactive measurement configuration questions are described in the chapter on coordinated measurements in the *Softkey Interface Reference* manual. Examples of coordinated measurements that can be performed between the emulator and the emulation analyzer are found in the "Using the Emulator" chapter.

Saving a Configuration

The last configuration question allows you to save the previous configuration specifications in a file which can be loaded back into the emulator at a later time.

Configuration file name? <FILE>

The name of the last configuration file is shown, or no filename is shown if you are modifying the default emulator configuration.

If you press <RETURN> without specifying a filename, the configuration is saved to a temporary file. This file is deleted when you exit the Softkey Interface with the "end release_system" command.

When you specify a filename, the configuration will be saved to two files; the filename specified with extensions of ".EA" and ".EB". The file with the ".EA" extension is the "source" copy of the file, and the file with the ".EB" extension is the "binary" or loadable copy of the file.

Ending out of emulation (with the "end" command) saves the current configuration, including the name of the most recently loaded configuration file, into a "continue" file. The continue file is not normally accessed.

Loading a Configuration

Configuration files which have been previously saved may be loaded with the following Softkey Interface command.

```
    load configuration <FILE> <RETURN>
```

This feature is especially useful after you have exited the Softkey Interface with the "end release_system" command; it saves you from having to modify the default configuration and answer all the questions again. To reload the current configuration, you can enter the following command.

```
    load configuration <RETURN>
```

5

Using the Emulator

Introduction

The "Getting Started" chapter shows you how to use the basic

This chapter discuss:

- Register names and classes
- Hardware breakpoint
- Vector area setting
- Analyzer topics
  - Specifying address and status for trigger or store condition
  - Specifying data for trigger or store condition
  - Specifying execute address for trigger or store condition
- Features available via "pod_command"

This chapter shows you how to:

- Store the contents of memory into absolute files
- Make coordinated measurements

REGISTER CLASS and NAME

Summary

70732 register designator. All available register class names and register names are listed below.

<REG_CLASS>

<REG_NAME>	Description
***(All basic registers)**	
PC	BASIC registers.
RW0	
RW1	
RW2	
RW3	
RW4	
RW5	
RW6	
RW7	
RW8	
RW9	
RW10	
RW11	
RW12	
RW13	
RW14	
RW15	
ISP	
USP	
FP	
CBP	
PBP	
PSW	

PBANK (Previous bank registers)

PPC	Saved PC
PPSW	Saved PSW
PPBP	Saved PBP
PR0	pw0 on previous bank
PR1	pw1 on previous bank
PR2	pw2 on previous bank
PR3	pw3 on previous bank
PR4	pw4 on previous bank
PR5	pw5 on previous bank
PR6	pw6 on previous bank
PR7	pw7 on previous bank
PR8	pw8 on previous bank
PR9	pw9 on previous bank
PR10	pw10 on previous bank
PR11	pw11 on previous bank
PR12	pw12 on previous bank
PR13	pw13 on previous bank
PR14	pw14 on previous bank
PR15	pw15 on previous bank

SYS (System control registers) (HP 64770A Only)

WDMOD	Watch dog timer mode	
WDCR	Watch dog timer control	(Write Only)
CH0CR	Memory controller channel 0	
CH1CR	Memory controller channel 1	
CH2CR	Memory controller channel 2	
CH3CR	Memory controller channel 3	
REFHREG	Refresh control	

SYS (System control registers) (HP64770B Only)

OMR	Operation mode
PDMR	Power down mode
STBYMD	Stand-by mode
CH0CR	Memory controller channel 0
CH1CR	Memory controller channel 1
CH2CR	Memory controller channel 2
CH3CR	Memory controller channel 3
REFHREG	Refresh control

TMR (Timer registers) (HP 64770A Only)

TRUN0	Timer control (TRUN0123)	
TRUN4	Timer control (TRUN4567)	
TRDC0	Double buffer control (TRDC0123)	
TRDC4	Double buffer control (TRDC4567)	
TFFCR0	Timer flip-flop control (TFFCR0123)	
TFFCR4	Timer flip-flop control (TFFCR4567)	
T01MOD	Timer source clk and mode	(Write Only)
T23MOD	Timer source clk and mode	(Write Only)
T45MOD	Timer source clk and mode	(Write Only)
T67MOD	Timer source clk and mode	(Write Only)
TREG0	Timer register 0	(Write Only)
TREG1	Timer register 1	(Write Only)
TREG2	Timer register 2	(Write Only)
TREG3	Timer register 3	(Write Only)
TREG4	Timer register 4	(Write Only)
TREG5	Timer register 5	(Write Only)
TREG6	Timer register 6	(Write Only)
TREG7	Timer register 7	(Write Only)
TT0RUN	Timer control 0	
TT1RUN	Timer control 1	
TT0MOD	Timer source clk and mode	
TT1MOD	Timer source clk and mode	
TT0FFCR	Timer flip-flop control	
TT1FFCR	Timer flip-flop control	
TTREG0	Timer register 0	(Write Only)
TTREG1	Timer register 1	(Write Only)
TTREG2	Timer register 2	(Write Only)
TTREG3	Timer register 3	(Write Only)
CAP1	Capture register 1	(Read Only)
CAP2	Capture register 2	(Read Only)
CAP3	Capture register 3	(Read Only)
CAP4	Capture register 4	(Read Only)

GTO (General output timer registers)(HP 64770B Only)

GTR	General timer	
CPRS0	Compare reg for "Set ch0"	
CPRS1	Compare reg for "Set ch1"	
CPRS2	Compare reg for "Set ch2"	
CPRS3	Compare reg for "Set ch3"	
CPRS4	Compare reg for "Set ch4"	
CPRS5	Compare reg for "Set ch5"	
CPRS6	Compare reg for "Set ch6"	
CPRS7	Compare reg for "Set ch7"	
CPRR0	Compare reg for "Reset ch0"	
CPRR1	Compare reg for "Reset ch1"	
CPRR2	Compare reg for "Reset ch2"	
CPRR3	Compare reg for "Reset ch3"	
CPRR4	Compare reg for "Reset ch4"	
CPRR5	Compare reg for "Reset ch5"	
CPRR6	Compare reg for "Reset ch6"	
CPRR7	Compare reg for "Reset ch7"	
DOMR1	Digital output mode	
DOCR	Digital output control	
DOR1	Digital out	(Read Only)
LGTO	Output level of GTO	
GTOEN	GTO enable	

GTI (General input timer registers)(HP 64770B Only)

CPCL0	Pulse conter latch 0	(Read Only)
CPCL1	Pulse conter latch 1	(Read Only)
CPCL2	Pulse conter latch 2	(Read Only)
CPCL3	Pulse conter latch 3	(Read Only)
GTA0P	GTIA positive edge 0	(Read Only)
GTA1P	GTIA positive edge 1	(Read Only)
GTA2P	GTIA positive edge 2	(Read Only)
GTA3P	GTIA positive edge 3	(Read Only)
GTA0N	GTIA negative edge 0	(Read Only)
GTA1N	GTIA negative edge 1	(Read Only)
GTA2N	GTIA negative edge 2	(Read Only)
GTA3N	GTIA nagative edge 3	(Read Only)
GTB0	GTIB edge 0	(Read Only)
GTB1	GTIB edge 1	(Read Only)
GTB2	GTIB edge 2	(Read Only)
GTB3	GTIB edge 3	(Read Only)

POUT (Pulse timer output registers)(HP 64770B Only)

TIOC	TIO control	
LPOUT	Output level of POUT	(Read Only)
DOMR2	Digital output mode	
DOR2	Digital out	
CPRD0	Compare register for Pout 0	
CPRD1	Compare register for Pout 1	
CPRD2	Compare register for Pout 2	
CPRD3	Compare register for Pout 3	
CPRD4	Compare register for Pout 4	
CPRD5	Compare register for Pout 5	
CPRD6	Compare register for Pout 6	
CPRD7	Compare register for Pout 7	

POC (Pulse output down-counter registers)(HP 64770B Only)

CPOC0	Pulse output counter of ch0
CPOC1	Pulse output counter of ch1
CPOC2	Pulse output counter of ch2
CPOC3	Pulse output counter of ch3
CPOC4	Pulse output counter of ch4
CPOC5	Pulse output counter of ch5
CPOC6	Pulse output counter of ch6
CPOC7	Pulse output counter of ch7

SC (Serial communication registers) (HP 64770A Only)

SC0CR	Serial channel 0 control
SC0MOD	Serial channel 0 mode
BR0CR	Serial channel 0 baud rate control
SC0BUF	Serial channel 0 buffer
SC1CR	Serial channel 1 control
SC1MOD	Serial channel 1 mode
BR1CR	Serial channel 1 baud rate control
SC1BUF	Serial channel 1 buffer
SC2CR	Serial channel 2 control
SC2MOD	Serial channel 2 mode
BR2CR	Serial channel 2 baud rate control
SC2BUF	Serial channel 2 buffer
ODE	Port 8 open-drain enable

SCI (Serial interface registers) (HP 64770B Only)

SCATB	SCIA transmit buffer	(Write Only)
SCARB	SCIA receive buffer	(Read Only)
SCAMR	SCIA mode	
SCASR	SCIA status	(Read Only)
SCACR	SCIA control	
SC2TB	SCI2 transmit buffer	(Write Only)
SC2RB	SCI2 receive buffer	(Read Only)
SC2MR	SCI2 mode	
SC2SR	SCI2 status	(Read Only)
SC2CR	SCI2 control	
SCBTB	SCIB transmit buffer	(Write Only)
SCBRB	SCIB receive buffer	(Read Only)
SCBMR	SCIB mode	
SCBSR	SCIB status	(Read Only)
SCBCR	SCIB control	

SEI (Expansion serial interface registers) (HP 64770B Only)

ASCR	Asynchronous mode command	(Write Only)
ASBF	Asynchronous mode buffer	(Read Only)
AKCR	Synchronous mode command	(Write Only)
SKBF	Synchronous mode buffer	(Read Only)
SE2CR	SEI2 control & status	
SE3BO	SEI3 buffer register out	(Write Only)
SE3BI	SEI3 buffer register in	(Read Only)
SE3SFO	SEI3 shift register out	(Write Only)
SE3SFI	SEI3 shift register in	(Read Only)
SE3CR	SEI3 control	
SESR	SEI shif register	
SECR	SEI control & status	

SMP (Serial monitor port registers) (HP 64770B Only)

SMISR	SMP input shift register	(Read Only)
SMOSR	SMP output shift register	(Write Only)
SMFULL	SMP input full register	

AD (A/D converter registers) (HP 64770A Only)

ADMOD	A/D converter mode	
ADCCS	ADC channel selector	
ADREG04	AD result 04	(Read Only)
ADREG15	AD result 15	(Read Only)
ADREG26	AD result 26	(Read Only)
ADREG37	AD result 37	(Read Only)

DMA (DMA controler registers) (HP 64770B Only)

MAR0	Memory address 0	
DTCR0	Data transfer count 0	
MAR1	Memory address 1	
DTCR1	Data transfer count 1	
MAR2	Memory address 2	
DTCR2	Data transfer count 2	
MAR3	Memory address 3	
DTCR3	Data transfer count 3	
MAR4	Memory address 4	
DTCR4	Data transfer count 4	
MAR5	Memory address 5	
DTCR5	Data transfer count 5	
CHSR0	Channel status 0	(Read Only)
CHSR1	Channel status 1	(Read Only)
CHSR2	Channel status 2	(Read Only)

INT (Interrupt control registers) (HP 64770A Only)

INTE0	Interrupt enable 0
INTE1	Interrupt enable 1
INTE2	Interrupt enable 2
INTE3	Interrupt enable 3
INTE4	Interrupt enable 4
INTE5	Interrupt enable 5
INTE6	Interrupt enable 6
INTE7	Interrupt enable 7
INTET0	Interrupt enable 8 bit timer 0
INTET1	Interrupt enable 8 bit timer 1
INTET2	Interrupt enable 8 bit timer 2
INTET3	Interrupt enable 8 bit timer 3
INTET4	Interrupt enable 8 bit timer 4
INTET5	Interrupt enable 8 bit timer 5
INTET6	Interrupt enable 8 bit timer 6
INTET7	Interrupt enable 8 bit timer 7
INTETT0	Interrupt enable 16 bit timer TTREG0
INTETT1	Interrupt enable 16 bit timer TTREG1
INTETT2	Interrupt enable 16 bit timer TTREG2
INTETT3	Interrupt enable 16 bit timer TTREG3
INTES0R	Interrupt enable serial 0 receive
INTES0T	Interrupt enable serial 0 transmit
INTES1R	Interrupt enable serial 1 receive
INTES1T	Interrupt enable serial 1 transmit
INTES2R	Interrupt enable serial 2 receive
INTES2T	Interrupt enable serial 2 transmit
INTEAD	Interrupt enable A/D
INTETASK	Interrupt enable TASK
INMIMC	Interrupt NMI input mode control

PIC (Interrupt control registers) (HP 64770B Only)

GTICR	General timer interrupt control
TIICR0	GTI interrupt control 0
TIICR1	GTI interrupt control 1
TIICR2	GTI interuput control 2
TOICR0	GTO interrupt control 0
TOICR1	GTO interrupt control 1
TOICR2	GTO interrupt control 2
TOICR3	GTO interrupt control 3
POICR0	POUT interrupt control 0
POICR1	POUT interrupt control 1
SIOICR0	SCI interupt control
DMAICR0	SCI2 interupt control
DMAICR1	SCI3 interupt control
SWICR0	SOFTWARE interrupt control 0
SWICR1	SOFTWARE interrupt control 1
SWICR2	SOFTWARE interrupt control 2
NMIRQ	NMI interrupt request flag
GTIRQ	GT interrupt request flag
TIIRQ	Timer input interrupt request flag
TOISRQ	Timer output set interrupt request
TOIRRQ	Timer output reset interrupt request
POIRQ	Pout interrupt flag
DMAIRQ	DMA interrupt flag
SWIRQ	SWI interrupt flag

PRT (Port registers) (HP 64770A Only)

PT0	Port 0	
PT1	Port 1	
PT2	Port 2	
PT3	Port 3	
PT4	Port 4	
PT5	Port 5	
PT6	Port 6	
PT7	Port 7	
PT8	Port 8	
PT9	Port 9	
PTA	Port A	
PTB	Port B	
PTC	Port C	(Read Only)
P0CR	Port 0 control	(Write Only)
P0FC	Port 0 function	(Write Only)
P1CR	Port 1 control	(Write Only)
P1FC	Port 1 function	(Write Only)
P2CR	Port 2 control	(Write Only)
P2FC	Port 2 function	(Write Only)
P3CR	Port 3 control	(Write Only)
P3FC	Port 3 function	(Write Only)
P4CR	Port 4 control	(Write Only)
P4FC	Port 4 function	(Write Only)
P5CR	Port 5 control	(Write Only)
P5FC	Port 5 function	(Write Only)
P6CR	Port 6 control	(Write Only)
P6FC	Port 6 function	(Write Only)
P7CR	Port 7 control	(Write Only)
P7FC	Port 7 function	(Write Only)
P8CR	Port 8 control	(Write Only)
P8FC	Port 8 function	(Write Only)
P9CR	Port 9 control	(Write Only)
P9FC	Port 9 function	(Write Only)
PACR	Port A control	(Write Only)
PAFC	Port A function	(Write Only)
PBCR	Port B control	(Write Only)
PBFC	Port B function	(Write Only)

PRT (Port registers) (HP 64770B Only)

P0	Port 0 data	
P1	Port 1 data	
P2	Port 2 data	
P3	Port 3 data	
P4	Port 4 data	
P5	Port 5 data	
P6	Port 6 data	
P9	Port 9 data	
PJ	Port J data	
PF	Port F data	
PG	Port G data	
PM	Port M data	
PH	Port H data	
PS	Port S data	
P0CR	Port 0 control	
P0FC	Port 0 function	
P1CR	Port 1 control	
P1FC	Port 1 function	
P2CR	Port 2 control	
P2FC	Port 2 function	
P3CR	Port 3 control	
P3FC	Port 3 function	
P4CR	Port 4 control	
P4FC	Port 4 function	
P5CR	Port 5 control	
P5FC	Port 5 function	
P6CR	Port 6 control	
P6FC	Port 6 function	
P9CR	Port 9 control	
PJCR	Port J control	(Write Only)
PFCR	Port F control	
PGCR	Port G control	
PMCR	Port M control	
PHCR	Port H control	
PSCR	Port S control	

PFSR0	Port function select 0	
PFSR1	Port function select 1	
PFSR2	Port function select 2	
PFSR3	Port function select 3	
PFSR4	Port function select 4	(Write Only)
PFSR5	Port function select 5	
PFSR6	Port function select 6	

OTHER (Other registers)

RB0	RB0
RB1	RB1
RB2	RB2
RB3	RB3
RB4	RB4
RB5	RB5
RB6	RB6
RB7	RB7
RB8	RB8
RB9	RB9
RB10	RB10
RB11	RB11
RB12	RB12
RB13	RB13
RB14	RB14
RB15	RB15
RD0	RD0
RD2	RD2
RD4	RD4
RD6	RD6
RD8	RD8
RD10	RD10
RD12	RD12
RD14	RD14
USPL	lower 16 bits of USP
USPH	upper 16 bits of USP
FPL	lower 16 bits of FP
FPH	upper 16 bits of FP

Hardware Breakpoints

The analyzer may generate a break request to the emulation processor. To break when the analyzer trigger condition is satisfied, use the "break_on_trigger" trace option.

Additionally, you can see the program states before the breakpoint in trace listing. Specify the trigger position at the end of trace listing by using "before" option.

When the trigger condition is found. the emulator execution will break into the emulation monitor. Then you can also see the trace listing mentioned above, enter the following commands.

```
trace before <QUALIFIER>
break_on_trigger<RETURN>
```

Without the trigger condition, the trigger will never occur and will never break.

Vector Area Setting

TLCS-9000 microprocessor has vector area(2k bytes). TLCS-9000 emulator uses three vector entry in vector area to realize the following emulator features.

- Break
- Single-Step
- Software Break Point

Single Chip Mode

If you use the TLCS-9000 emulator in single chip mode, you do not need to set the vector entry since the emulator set automatically. The values of PC, PSW, and CBP are set bye vector entry when the emulator breaks into the monitor from reset state.

External Bus Mode

IF you use the TLCS-9000 emulator in external bus mode, the way of the emulator’s operations differ according to "Enable emulation VBP?" configuration and memory mapping.

"Enable emulation VBP? yes"

The emulator read Vector Base Pointer(VBP) from emulation VBP. When the emulator breaks into the monitor from reset state, the value of emulation VBP is specified by "Vector base address?" configuration.

If vector area are mapped as emulation memory, the emulator sets the vector entry when the emulator breaks into the monitor from reset state. When the emulator breaks into the monitor from reset state, the values of PC, PSW, and CBP are set by vector entry.

If vector area are mapped as target memory, the emulator uses copy of vector area. The emulator copies data of vector ares when the emulator breaks into the monitor from reset state, and then sets the vector entry. When the emulator breaks into the monitor from reset, the value of PC, and PSW are set by vector entry and the value of CBP is specified by "Initial CBP value?" configuration.

"Enable emulation VBP? no"

In this case, the emulator does not set the vector entry. So you must set up the vector entry to realize the emulator features. If you do not set the vector entry, the emulator can not operate correctly.

Even if you specify that "Enable emulation VBP? no", the value of PC, PSW, and CBP are specified in the same way as you specify that "Enable emulation VBP? yes" when the emulator breaks into the monitor from reset state.

Set the vector area as following.

Vector number	Offset	value	Purpose
12	60H	0000H	Break
	62H	0202H	
	64H	00xxH	
	66H	0000H	
Vector number	**Offset**	**value**	**Purpose**

14	70H	0000H	Step
	72H	0204H	
	74H	00xxH	
	76H	0000H	
31	F8H	0000H	Software break point
	FAH	0200H	
	FCH	00xxH	
	FEH	0000H	

xx: Upper 8 bits of monitor area

Analyzer Topics

Specifying Address and Status for Trigger or Store Condition

The analyzer captures the actual bus states and execute states by exclusive bus. In some case, bus state and execute state are captured simultaneously. To specify actual bus states for trigger or store condition by "address", "status", and "data", you should add "status bus" condition to trigger/store condition as following.

```
trace after address 1000h status bus<RETURN>
trace after status write and bus
```

Specifying Data for Trigger or Store Condition

The analyzer captures the actual bus states of the TLCS-9000 microprocessor. When you specify a data in the analyzer trigger or store condition, the ways of analyzer data specification differ according to the address and the data size.

To trigger analyzer when the TLCS-9000 microprocessor accesses the byte data 12h at address 1000h(even address), enter the following,

```
trace after address 1000h data 0xx12h status
bus and byte
```

To trigger analyzer when the TLCS-9000 microprocessor accesses the byte data 12h at address 1001h(odd address), enter the following,

```
    trace after address 1001h data 012xxh status
    bus and byte
```

To trigger analyzer when the TLCS-9000 microprocessor accesses the word data 1234h at address 1001h(odd address), the data bus activity of cycles will be as follows.

```
Sequencer level  Address bus  Data bus
            1       1001        34xx
            2       1002        xx12
```

In this case, you need to use the analyzer sequential trigger capabilities. We do not describe the detail about the sequential trigger feature. Only how to trigger the analyzer at this example is described. To specify the condition, enter;

```
    trace find_sequence 1001h data 34xxh status
    bus trigger after 1002h data 0xx12h status
    bus <RETURN>
```

Specifying Execute Address for Trigger or Store Condition

To specify "eaddr" for trigger or store condition, you must specify even addresses as execute address. To trigger analyzer when TLCS-9000 microprocessor executes instruction at address 2000h, enter the following,

```
    trace after eaddr 2000h <RETURN>
```

Features Available via Pod Commands

Several emulation features available in the Terminal Interface but not in the Softkey Interface may be accessed via the following emulation commands.

```
display pod_command <RETURN>
pod_command '<Terminal Interface command>'
<RETURN>
```

Some of the most notable Terminal Interface features not available in the Softkey Interface are:

- Searching memory for strings or numeric expressions.
- Sequencing in the analyzer.

Refer to your Terminal Interface documentation for information on how to perform these tasks.

Note

Be careful when using the "pod_command". The Softkey Interface, and the configuration files in particular, assume that the configuration of the HP 64700 pod is NOT changed except by the Softkey Interface. Be aware that what you see in "modify configuration" will NOT reflect the HP 64700 pod's configuration if you change the pod's configuration with this command. Also, commands which affect the communications channel should NOT be used at all. Other commands may confuse the protocol depending upon how they are used. The following commands are not recommended for use with "pod_command":

stty, **po**, **xp -** Do not use, will change channel operation and hang.
echo, **mac -** Usage may confuse the protocol in use on the channel.
wait - Do not use, will tie up the pod, blocking access.
init, **pv -** Will reset pod and force end release_system.
t - Do not use, will confuse trace status polling and unload.0h)0

Storing Memory Contents to an Absolute File

The "Getting Started" chapter shows you how to load absolute files into emulation or target system memory. You can also store emulation or target system memory to an absolute file with the following command.

```
store memory 800h thru 84fh to absfile
<RETURN>
```

The command above causes the contents of memory locations 800H-84FH to be stored in the absolute file "absfile.X". Notice that the ".X" extension is appended to the specified filename.

Coordinated Measurements

For information on coordinated measurements and how to use them, refer to the "Coordinated Measurements" chapter in the *Softkey Interface Reference* manual.

Index