About this Manual

We’ve added this manual to the Agilent website in an effort to help you support your product. This manual is the best copy we could find; it may be incomplete or contain dated information. If we find a more recent copy in the future, we will add it to the Agilent website.

Support for Your Product

Agilent no longer sells or supports this product. Our service centers may be able to perform calibration if no repair parts are needed, but no other support from Agilent is available. You will find any other available product information on the Agilent Test & Measurement website, www.tm.agilent.com.

HP References in this Manual

This manual may contain references to HP or Hewlett-Packard. Please note that Hewlett-Packard's former test and measurement, semiconductor products and chemical analysis businesses are now part of Agilent Technologies. We have made no changes to this manual copy. In other documentation, to reduce potential confusion, the only change to product numbers and names has been in the company name prefix: where a product number/name was HP XXXX the current name/number is now Agilent XXXX. For example, model number HP8648A is now model number Agilent 8648A.

HP 64785

SH-7000 Emulator Softkey Interface

User’s Guide

HEWLETT
PACKARD

HP Part No. 64785-97001
Printed in Japan
June 1994

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 merchantability 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.

© Copyright 1994, Hewlett-Packard Company.

This document contains proprietary information, which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another language without the prior written consent of Hewlett-Packard Company. The information contained in this document is subject to change without notice.

HP is a trademark of Hewlett-Packard Company.

UNIX is a registered trademark of UNIX System Laboratories Inc. in the U.S.A. and other countries.

SH-7000™ is trademark of Hitachi Ltd.

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 **64785-97001, June 1994**

Using this Manual

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

- HP 64785A SH-7000 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 demo board/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.

Organization

Chapter 1 **Introduction to the SH-7000 Emulator.** This chapter briefly introduces you to the concept of emulation and lists the basic features of the SH-7000 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 breakpoins, 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 SH-7000 Emulator

2 Getting Started

3 In-Circuit Emulation Topics

4 Configuring the Emulator

Illustrations

Tables

1

Introduction to the SH-7000 Emulator

Introduction

The topics in this chapter include:

- Purpose of the emulator
- Features of the emulator
- Limitations and Restrictions of the SH-7000 emulator

Purpose of the Emulator

The SH-7000 emulator is designed to replace the SH-7000 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.

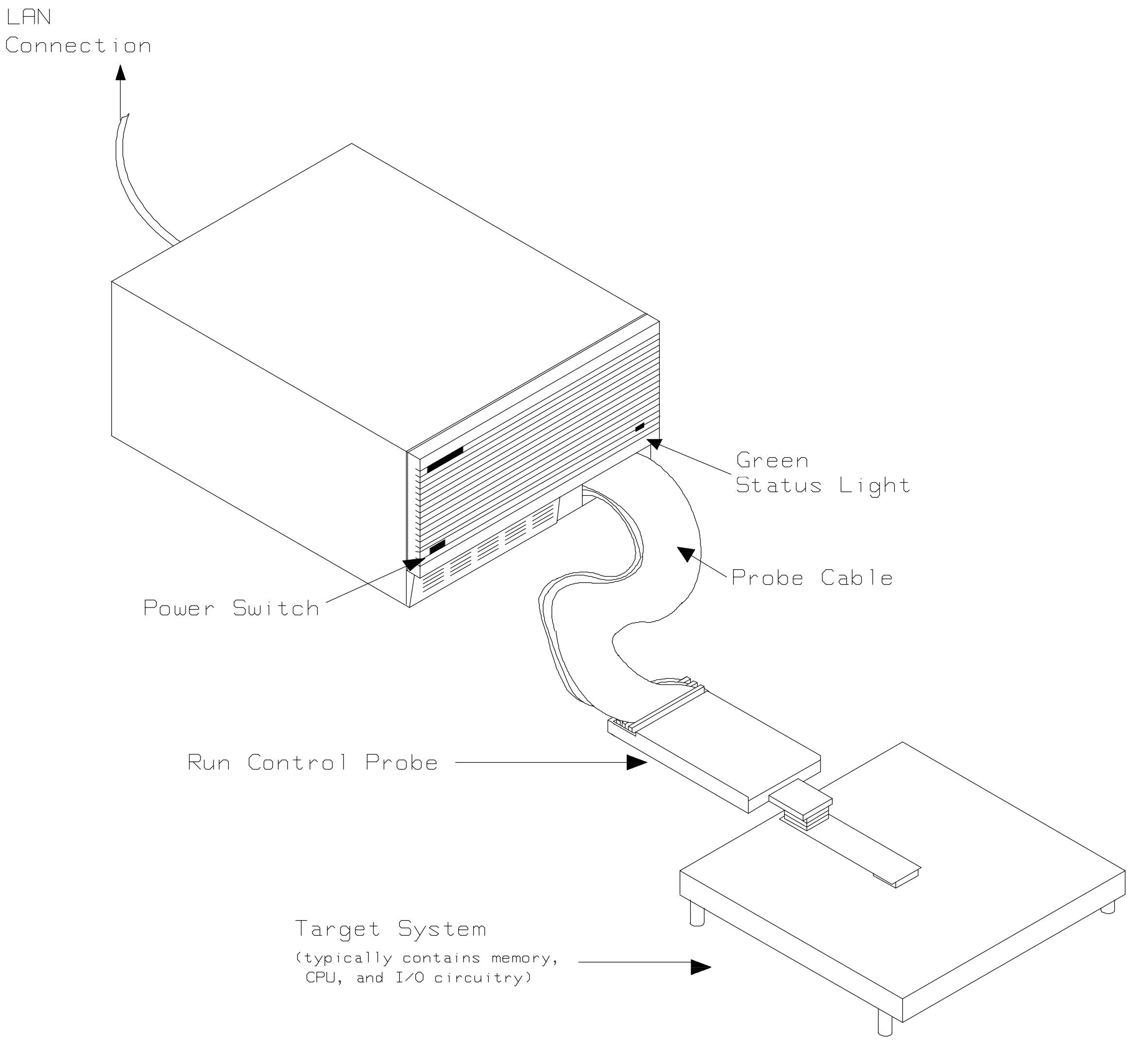


Figure 1-1 HP 64785A Emulator for SH-7000

Features of the SH-7000 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 SH-7000 emulator supports the microprocessors listed in Table 1-1.

Table 1-1 Supported Microprocessors

Supported Microprocessors	Reffered to as
HD6417032F	SH-7032
HD6477034F HD6437034F	SH-7034

Clock Speeds

The SH-7000 emulator runs with a target system clock from 2.0 to 20.0 MHz.

Emulation memory

The SH-7000 emulator can be used with one of the following Emulation Memory Module.

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

You can define up to 16 memory ranges. The minimum amount of emulation memory that can be allocated to a range is 16K byte. 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 SH-7000 emulator is used with one of the following analyzers which allows you to trace code execution and processor activity.

- HP64704 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 SH-7000 internal register contents. This includes the ability to modify the program counter(PC) value so you can control where the emulator starts program run.

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 SH-7000 instructions which read the target memory locations and send their contents to the emulation controller.

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 the emulator break to the monitor program when the analyzer finds a specific state or states, allowing you to perform post-mortem analysis of the program execution. You can also set software breakpoints in your program. This feature is realized by inserting a special instruction into user program. One of undefined opcodes (0000 hex) is used as software breakpoint instruction. Refer to the "Using Software Breakpoints" section of "Getting Started" chapter for more information.

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 needs to break to the monitor to perform an action you requested, such as displaying target system memory.) The Emulator features performed in real-time include: running and analyzer tracing.

The emulator features not performed in real-time includes: display or modification of target system memory, load/dump of target memory, display or modification of registers.

Coverage and Memory Copy

The SH-7000 emulator does not support coverage test and momory copy from target memory.

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

Interrupts While in the Monitor

The SH-7000 emulator does not accept any interrupts in the monitor program. Edge sensed interrupts are suspended while running the monitor program, and such interrupts will occur when context is changed to the user program. Level sensed interrupts are ignored during the monitor program.

$\overline{\text{BREQ}}$ signal is always accepted by the SH-7000 emulator.

Watchdog Timer

The watchdog timer is suspended count up while the emulator is running the monitor program.

Monitor Break at Sleep/Standby Mode

When the SH-7000 emulator breaks into the monitor program, sleep or software standby mode is released. Then, PC indicates next address of "SLEEP" instruction.

Memory Module

One state access and DRAM short pitch access are not allowed, when you operate the emulator using 25ns memory module with the clock faster than 16.6MHz.

One state access is not allowed, when you operate the emulator using 20ns memory module with the target system which uses $\overline{\text{BREQ}}$ signal and the clock faster than 16.6MHz.

DMA support

Direct memory access to the emulation memory by external DMAC is not allowed.

Single address mode transfer to the emulation memory by internal DMAC is not allowed.

Warp Mode

SH-7000 emulator does not support Warp mode.

Evaluation Chip

Hewlett-Packard makes no warranty of the problem caused by the SH-7000 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 64785A SH-7000 emulator (for the SH-7032/34 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 SH-7000 emulator.

A Look at the Demo Program

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

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/hp64785

Compiling the Demo Program

The demo program is written for and compiled/linked with the Hitachi SH7000 C Compiler Package. The demo program was compiled with the following command.

```
$ shc -debug spmt_demo.c <RETURN>
$ asmsh -debug init.src <RETURN>
```

Linking the Demo Program

The following command was used to generate the absolute file. The contents of "spmt_demo.k" linkage editor subcommand file is shown in figure 2-1.

```
$ lnk -subcommand=spmt_demo.k<RETURN>
```

```
debug
input spmt_demo,init
library shclib.lib
start P(1000),B(0F000000)
output spmt_demo
print spmt_demo
exit
```

Figure 2-1 Linkage Editor Subcommand File

Generate HP Absolute file

To generate HP Absolute file for the Softkey Interface, you need to use "**shcnvhp**" absolute file format converter program. The shcnvhp converter is provided with HP 64785 Softkey Interface. To generate HP Absolute file, enter following command:

```
$ shcnvhp spmt_demo <RETURN>
```

You will see that spmt_demo.X, spmt_demo.L, and spmt_demo.A are generated. These are sufficient throughout this chapter.

Note

You need to specify "debug" command line option to compiler, assembler and linker command to generate local symbol information.

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

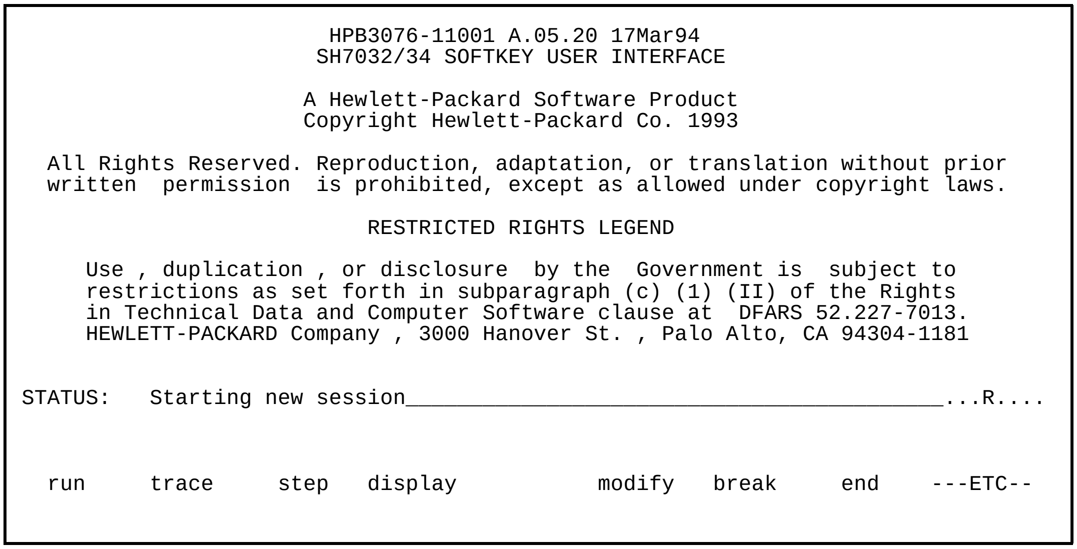
If **/usr/hp64000/bin** is specified in your PATH environment variable, you can also enter the Softkey Interface with the following command.

```
$ 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:      sh70        sh7034     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.



```
                        HPB3076-11001 A.05.20 17Mar94
                       SH7032/34 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>
Processor type?  7032 <RETURN>
Processor operation mode?  mode_0 <RETURN>
Area 1 memory type?  other <RETURN>
Modify memory configuration?  yes <RETURN>
```

Now you should be facing memory mapping screen. One mapper term must be specified for the demo program. Enter the following line to map the program code.

```
    0h thru 3fffh 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.

Note

The internal RAM/ROM area and emulation monitor area are mapped automatically. And the emulation memory system does not introduce internal RAM/ROM area in memory mapping display.

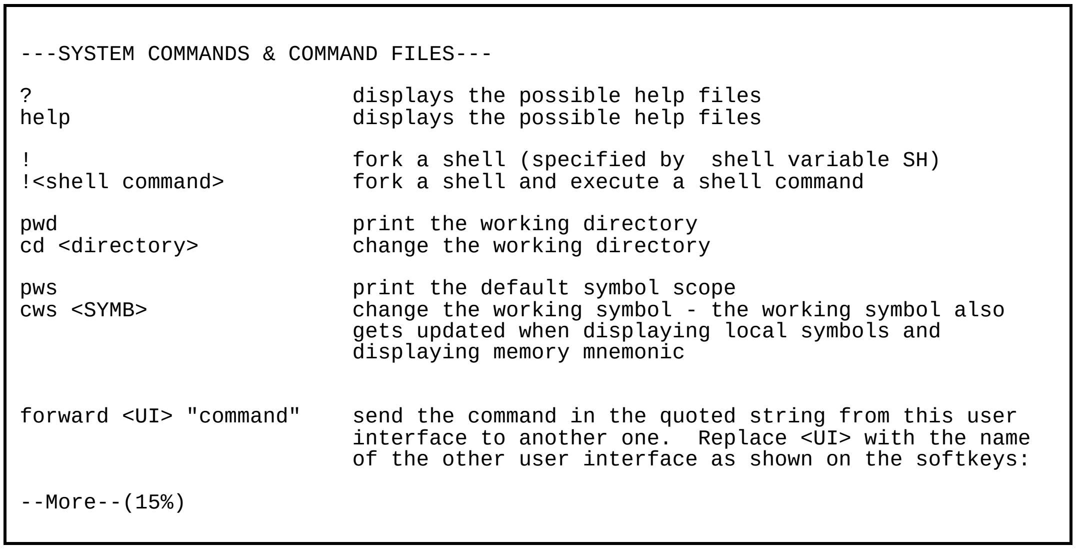
On-Line Help

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>
```



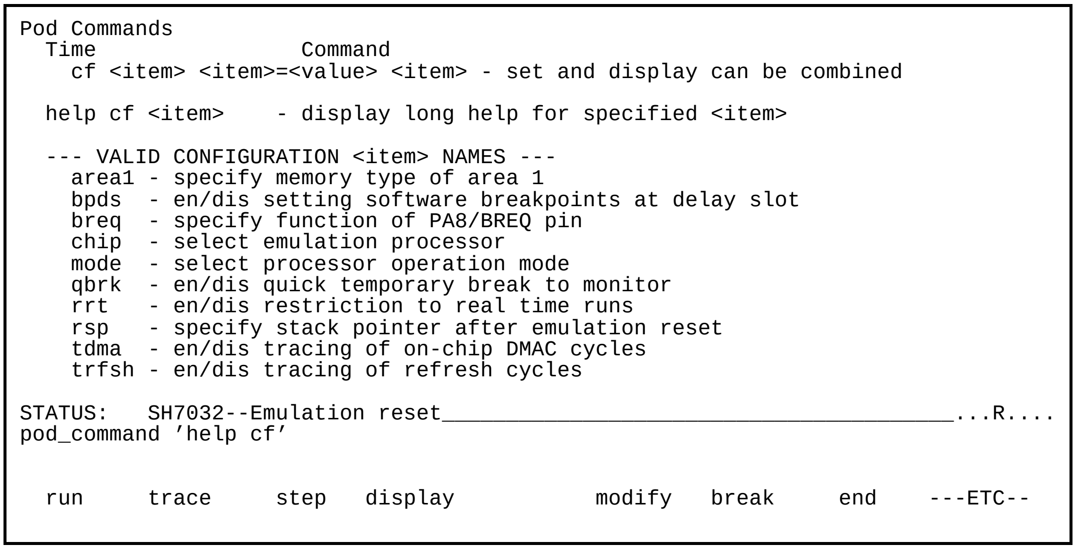
```
---SYSTEM COMMANDS & COMMAND FILES---

?                          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 ---
    area1 - specify memory type of area 1
    bpds  - en/dis setting software breakpoints at delay slot
    breq  - specify function of PA8/BREQ pin
    chip  - select emulation processor
    mode  - select processor operation mode
    qbrk  - en/dis quick temporary break to monitor
    rrt   - en/dis restriction to real time runs
    rsp   - specify stack pointer after emulation reset
    tdma  - en/dis tracing of on-chip DMAC cycles
    trfsh - en/dis tracing of refresh cycles

STATUS:   SH7032--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 format:

- HP absolute

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 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/hp64785/spmt_demo
<RETURN>
```

Displaying Symbols

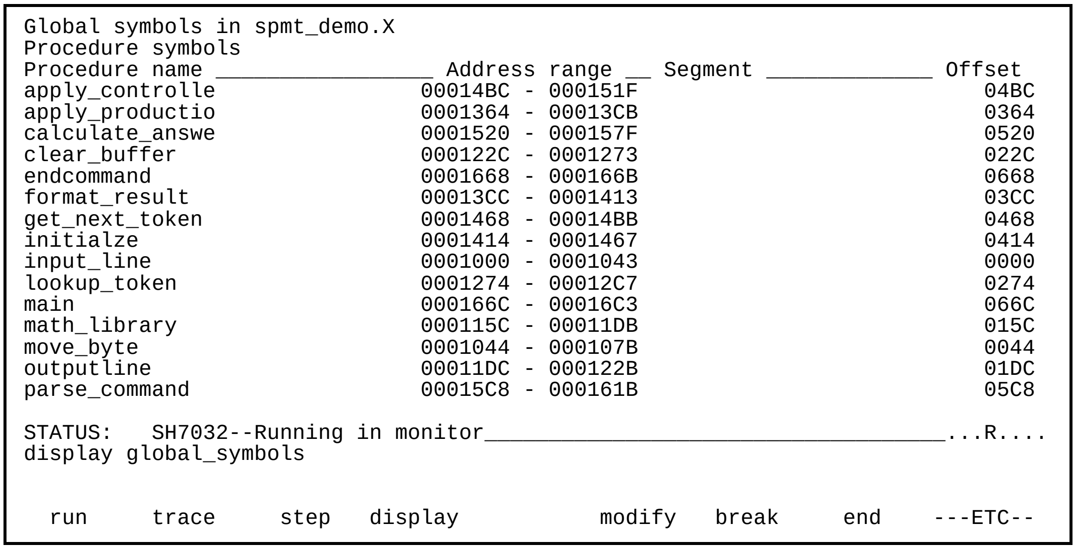
When you load an absolute file into memory (unless you use the "nosymbols" syntax), symbol information is also loaded. Both global symbols and symbols that are local to a source file can be displayed.

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_controlle                00014BC - 000151F                           04BC
apply_productio                0001364 - 00013CB                           0364
calculate_answe                0001520 - 000157F                           0520
clear_buffer                   000122C - 0001273                           022C
endcommand                     0001668 - 000166B                           0668
format_result                  00013CC - 0001413                           03CC
get_next_token                 0001468 - 00014BB                           0468
initialze                      0001414 - 0001467                           0414
input_line                     0001000 - 0001043                           0000
lookup_token                   0001274 - 00012C7                           0274
main                           000166C - 00016C3                           066C
math_library                   000115C - 00011DB                           015C
move_byte                      0001044 - 000107B                           0044
outputline                     00011DC - 000122B                           01DC
parse_command                  00015C8 - 000161B                           05C8

STATUS:   SH7032--Running in monitor____________________________________...R....
display global_symbols


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

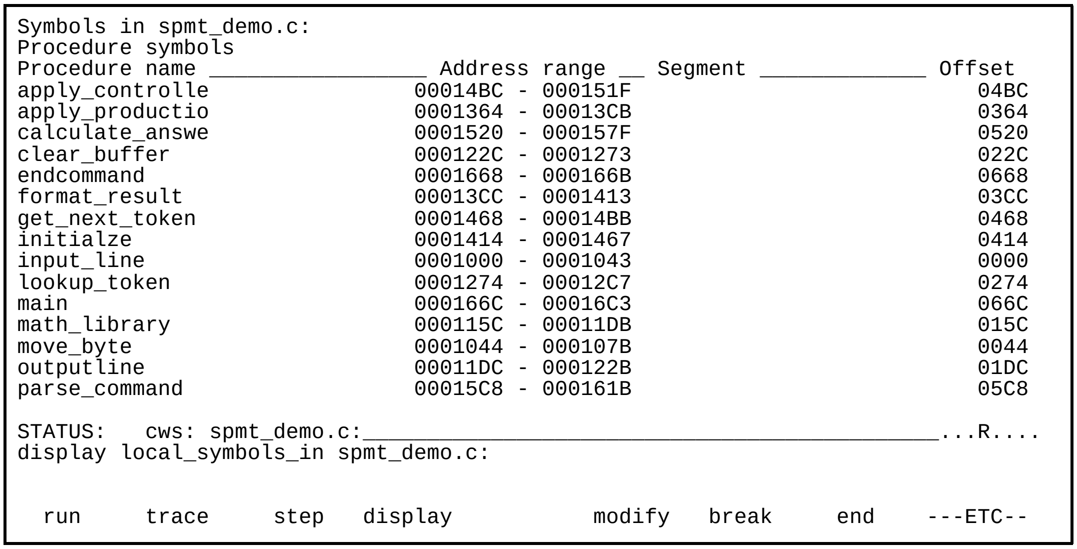
Local

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.



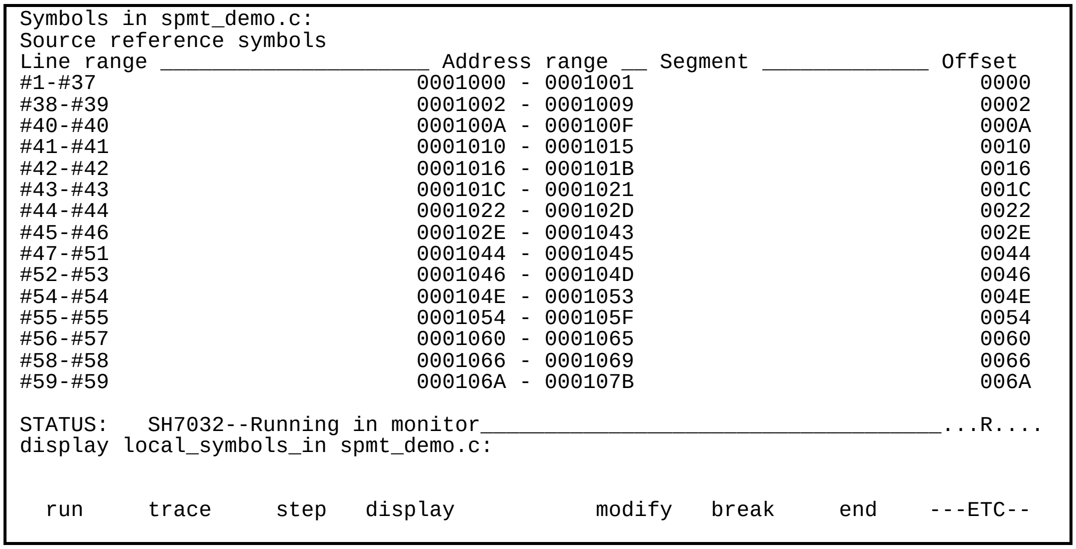
```
Symbols in spmt_demo.c:
Procedure symbols
Procedure name _________________ Address range __ Segment _____________ Offset
apply_controlle                00014BC - 000151F                           04BC
apply_productio                0001364 - 00013CB                           0364
calculate_answe                0001520 - 000157F                           0520
clear_buffer                   000122C - 0001273                           022C
endcommand                     0001668 - 000166B                           0668
format_result                  00013CC - 0001413                           03CC
get_next_token                 0001468 - 00014BB                           0468
initialze                      0001414 - 0001467                           0414
input_line                     0001000 - 0001043                           0000
lookup_token                   0001274 - 00012C7                           0274
main                           000166C - 00016C3                           066C
math_library                   000115C - 00011DB                           015C
move_byte                      0001044 - 000107B                           0044
outputline                     00011DC - 000122B                           01DC
parse_command                  00015C8 - 000161B                           05C8

STATUS:   cws: spmt_demo.c:_____________________________________________...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.

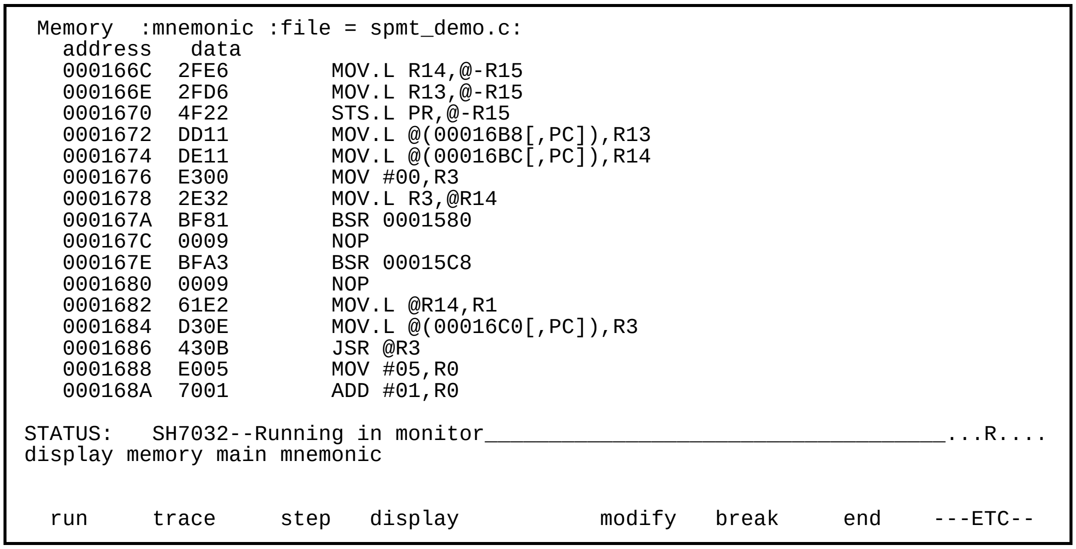


```
Symbols in spmt_demo.c:
Source reference symbols
Line range ______________________ Address range __ Segment _____________ Offset
#1-#37                          0001000 - 0001001                          0000
#38-#39                         0001002 - 0001009                          0002
#40-#40                         000100A - 000100F                          000A
#41-#41                         0001010 - 0001015                          0010
#42-#42                         0001016 - 000101B                          0016
#43-#43                         000101C - 0001021                          001C
#44-#44                         0001022 - 000102D                          0022
#45-#46                         000102E - 0001043                          002E
#47-#51                         0001044 - 0001045                          0044
#52-#53                         0001046 - 000104D                          0046
#54-#54                         000104E - 0001053                          004E
#55-#55                         0001054 - 000105F                          0054
#56-#57                         0001060 - 0001065                          0060
#58-#58                         0001066 - 0001069                          0066
#59-#59                         000106A - 000107B                          006A

STATUS:   SH7032--Running in monitor____________________________________...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>



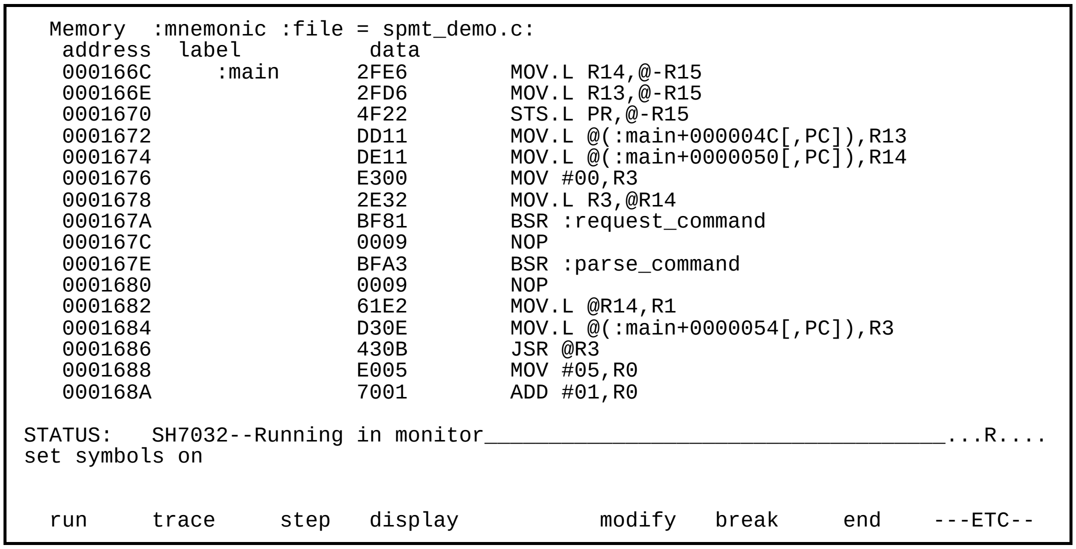
```
 Memory  :mnemonic :file = spmt_demo.c:
   address   data
   000166C  2FE6        MOV.L R14,@-R15
   000166E  2FD6        MOV.L R13,@-R15
   0001670  4F22        STS.L PR,@-R15
   0001672  DD11        MOV.L @(00016B8[,PC]),R13
   0001674  DE11        MOV.L @(00016BC[,PC]),R14
   0001676  E300        MOV #00,R3
   0001678  2E32        MOV.L R3,@R14
   000167A  BF81        BSR 0001580
   000167C  0009        NOP
   000167E  BFA3        BSR 00015C8
   0001680  0009        NOP
   0001682  61E2        MOV.L @R14,R1
   0001684  D30E        MOV.L @(00016C0[,PC]),R3
   0001686  430B        JSR @R3
   0001688  E005        MOV #05,R0
   000168A  7001        ADD #01,R0

STATUS:   SH7032--Running in monitor____________________________________...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>



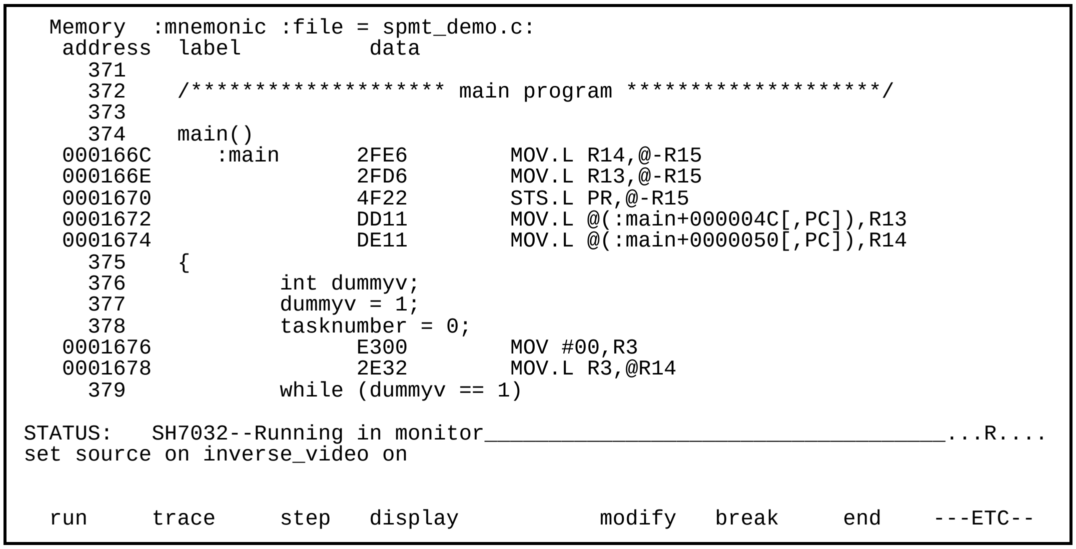
```
  Memory  :mnemonic :file = spmt_demo.c:
   address  label          data
   000166C     :main      2FE6        MOV.L R14,@-R15
   000166E                2FD6        MOV.L R13,@-R15
   0001670                4F22        STS.L PR,@-R15
   0001672                DD11        MOV.L @(:main+000004C[,PC]),R13
   0001674                DE11        MOV.L @(:main+0000050[,PC]),R14
   0001676                E300        MOV #00,R3
   0001678                2E32        MOV.L R3,@R14
   000167A                BF81        BSR :request_command
   000167C                0009        NOP
   000167E                BFA3        BSR :parse_command
   0001680                0009        NOP
   0001682                61E2        MOV.L @R14,R1
   0001684                D30E        MOV.L @(:main+0000054[,PC]),R3
   0001686                430B        JSR @R3
   0001688                E005        MOV #05,R0
   000168A                7001        ADD #01,R0

STATUS:   SH7032--Running in monitor____________________________________...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.c:
   address  label          data
     371
     372    /******************** main program ********************/
     373
     374    main()
   000166C     :main      2FE6        MOV.L R14,@-R15
   000166E                2FD6        MOV.L R13,@-R15
   0001670                4F22        STS.L PR,@-R15
   0001672                DD11        MOV.L @(:main+000004C[,PC]),R13
   0001674                DE11        MOV.L @(:main+0000050[,PC]),R14
     375    {
     376            int dummyv;
     377            dummyv = 1;
     378            tasknumber = 0;
   0001676                E300        MOV #00,R3
   0001678                2E32        MOV.L R3,@R14
     379            while (dummyv == 1)

STATUS:   SH7032--Running in monitor____________________________________...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 the Emulation from Target Reset" section in the "In-Circuit Emulation" chapter).

Note

Run and step commands from odd address are not allowed. Always you must perform run and step commands from even address.

Note

When you perform step command for delayed branch instruction, the emulator steps an instruction in delay slot too.

Displaying Memory

The demo program "spmt_demo.c" alters memory.

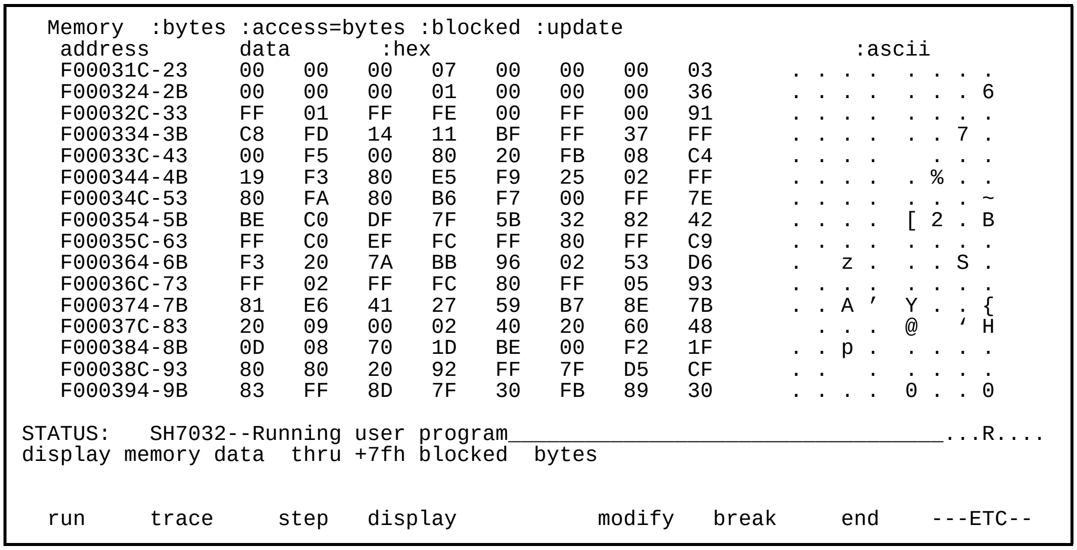
Using Symbolic Addresses

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**.



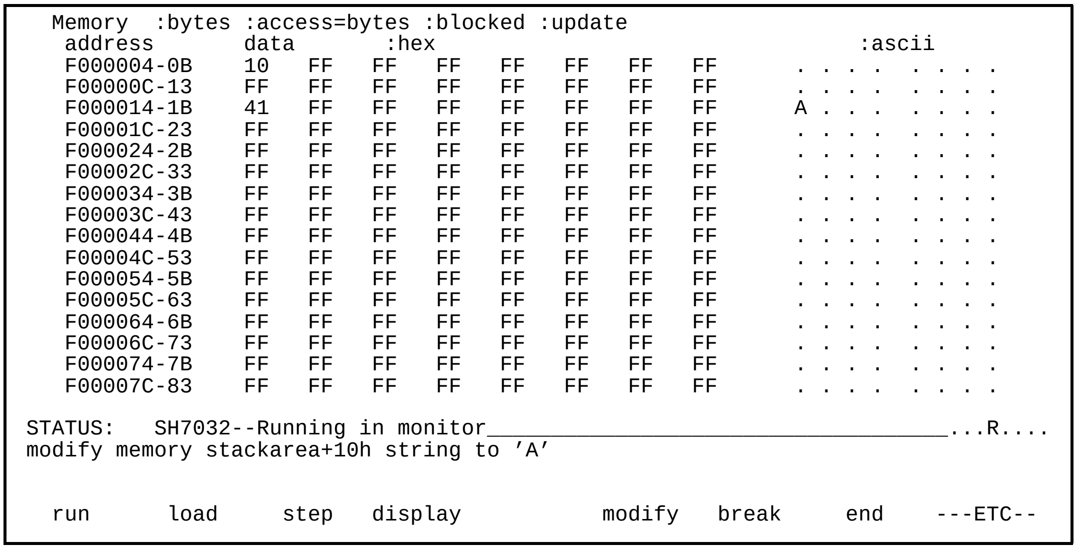
```
  Memory  :bytes :access=bytes :blocked :update
   address       data       :hex                                 :ascii
   F00031C-23    00   00   00   07   00   00   00   03      . . . .  . . . .
   F000324-2B    00   00   00   01   00   00   00   36      . . . .  . . . 6
   F00032C-33    FF   01   FF   FE   00   FF   00   91      . . . .  . . . .
   F000334-3B    C8   FD   14   11   BF   FF   37   FF      . . . .  . . 7 .
   F00033C-43    00   F5   00   80   20   FB   08   C4      . . . .    . . .
   F000344-4B    19   F3   80   E5   F9   25   02   FF      . . . .  . % . .
   F00034C-53    80   FA   80   B6   F7   00   FF   7E      . . . .  . . . ~
   F000354-5B    BE   C0   DF   7F   5B   32   82   42      . . . .  [ 2 . B
   F00035C-63    FF   C0   EF   FC   FF   80   FF   C9      . . . .  . . . .
   F000364-6B    F3   20   7A   BB   96   02   53   D6      .   z .  . . S .
   F00036C-73    FF   02   FF   FC   80   FF   05   93      . . . .  . . . .
   F000374-7B    81   E6   41   27   59   B7   8E   7B      . . A ’  Y . . {
   F00037C-83    20   09   00   02   40   20   60   48        . . .  @   ‘ H
   F000384-8B    0D   08   70   1D   BE   00   F2   1F      . . p .  . . . .
   F00038C-93    80   80   20   92   FF   7F   D5   CF      . .   .  . . . .
   F000394-9B    83   FF   8D   7F   30   FB   89   30      . . . .  0 . . 0

STATUS:   SH7032--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 f000004 hex and f000014 hex respectivity. For example, to enter the ’10h’ at address f000004 and enter ’A’ at address f000014 : 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
   F000004-0B    10   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
   F00000C-13    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
   F000014-1B    41   FF   FF   FF   FF   FF   FF   FF      A . . .  . . . .
   F00001C-23    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
   F000024-2B    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
   F00002C-33    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
   F000034-3B    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
   F00003C-43    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
   F000044-4B    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
   F00004C-53    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
   F000054-5B    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
   F00005C-63    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
   F000064-6B    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
   F00006C-73    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
   F000074-7B    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
   F00007C-83    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .

STATUS:   SH7032--Running in monitor____________________________________...R....
modify memory stackarea+10h string to ’A’


  run      load     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.

Note

If DMA transfer by internal DMAC is in progress with BURST transfer mode, **break** command is suspended and occurs after DMA transfer is completed.

Using Software Breakpoints

Software breakpoints are handled by the SH-7000 undefined instruction (breakpoint interrupt instruction:0000h). 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, enabled, disabled, or removed 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

A software breakpoint at delay slot causes slot invalid instruction exception in your program.

Note

You must only set software breakpoints at even address. If you set a software breakpoint at odd address, the emulator generates a error.

Note

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

When software breakpoints are enabled and the emulator detects the breakpoint interrupt instruction(0000h), it generates a break into the monitor.

If the breakpoint interrupt instruction(0000h) was generated by a software breakpoint, execution breaks to the monitor, and the breakpoint interrupt instruction is replaced by the original opcode. A subsequent run or step command will execute from this address.

If the breakpoint interrupt was generated by a undefined instruction (0000h) 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 SH-7000 breakpoint interrupt instruction (0000h) will be placed at the address specified. When the breakpoint interrupt instruction is executed, program execution will break into the monitor.

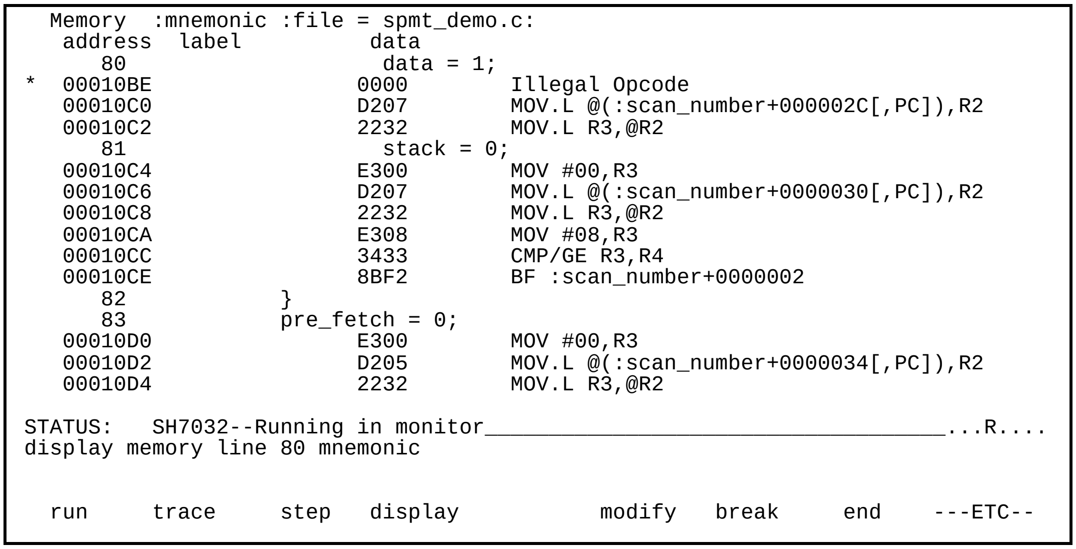
Setting a Software Breakpoint

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>



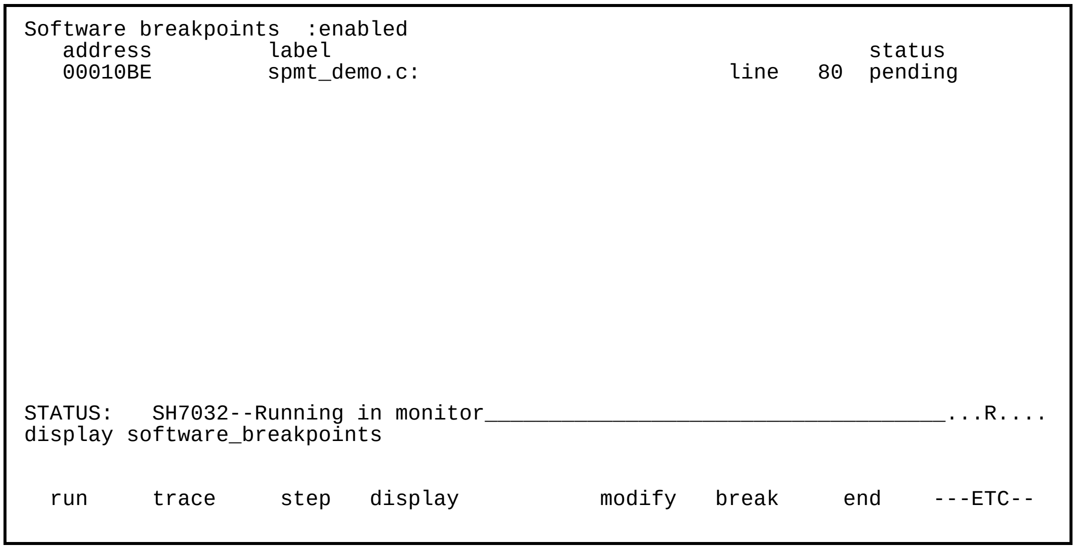
```
  Memory  :mnemonic :file = spmt_demo.c:
   address  label          data
      80                    data = 1;
*  00010BE                0000        Illegal Opcode
   00010C0                D207        MOV.L @(:scan_number+000002C[,PC]),R2
   00010C2                2232        MOV.L R3,@R2
      81                    stack = 0;
   00010C4                E300        MOV #00,R3
   00010C6                D207        MOV.L @(:scan_number+0000030[,PC]),R2
   00010C8                2232        MOV.L R3,@R2
   00010CA                E308        MOV #08,R3
   00010CC                3433        CMP/GE R3,R4
   00010CE                8BF2        BF :scan_number+0000002
      82            }
      83            pre_fetch = 0;
   00010D0                E300        MOV #00,R3
   00010D2                D205        MOV.L @(:scan_number+0000034[,PC]),R2
   00010D4                2232        MOV.L R3,@R2

STATUS:   SH7032--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
   00010BE         spmt_demo.c:                        line   80  pending










STATUS:   SH7032--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 reenter 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.

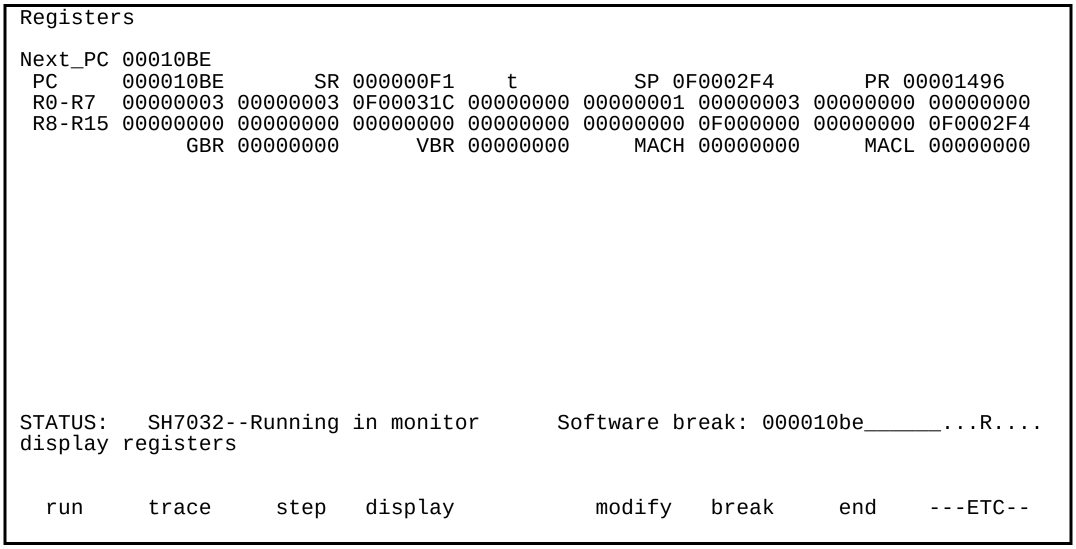
To clear all software breakpoints, you can enter the following command.

```
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>
```



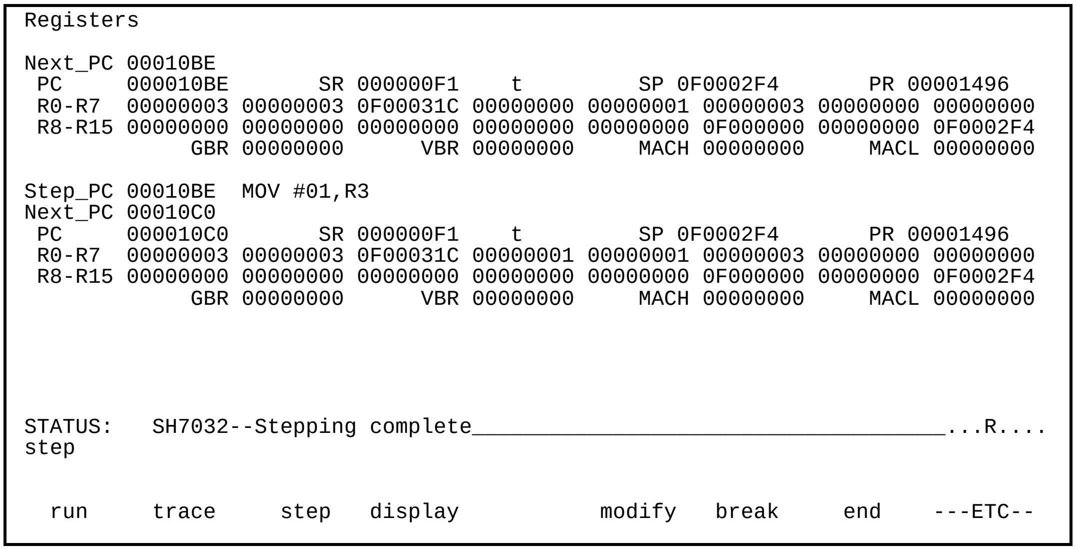
```
Registers

Next_PC 00010BE
 PC     000010BE       SR 000000F1    t         SP 0F0002F4       PR 00001496
 R0-R7  00000003 00000003 0F00031C 00000000 00000001 00000003 00000000 00000000
 R8-R15 00000000 00000000 00000000 00000000 00000000 0F000000 00000000 0F0002F4
             GBR 00000000      VBR 00000000     MACH 00000000     MACL 00000000











STATUS:   SH7032--Running in monitor      Software break: 000010be______...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 00010BE
 PC     000010BE       SR 000000F1    t         SP 0F0002F4       PR 00001496
 R0-R7  00000003 00000003 0F00031C 00000000 00000001 00000003 00000000 00000000
 R8-R15 00000000 00000000 00000000 00000000 00000000 0F000000 00000000 0F0002F4
             GBR 00000000      VBR 00000000     MACH 00000000     MACL 00000000

Step_PC 00010BE  MOV #01,R3
Next_PC 00010C0
 PC     000010C0       SR 000000F1    t         SP 0F0002F4       PR 00001496
 R0-R7  00000003 00000003 0F00031C 00000001 00000001 00000003 00000000 00000000
 R8-R15 00000000 00000000 00000000 00000000 00000000 0F000000 00000000 0F0002F4
             GBR 00000000      VBR 00000000     MACH 00000000     MACL 00000000




STATUS:   SH7032--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.

Note

Step and run commands from odd address are not allowed. Always you must perform step and run commands from even address.

Note

When you perform step command for delayed branch instruction, the emulator steps an instruction in delay slot too.

Using the Analyzer

HP 64700 emulators contain an emulation analyzer. The emulation analyzer monitors the internal emulation lines (address, data, and status). Optionally, you may have an additional 16 trace signals which monitor external input lines. The analyzer collects data at each pulse of a clock signal, and saves the data (a trace state) if it meets a "storage qualification" condition.

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.

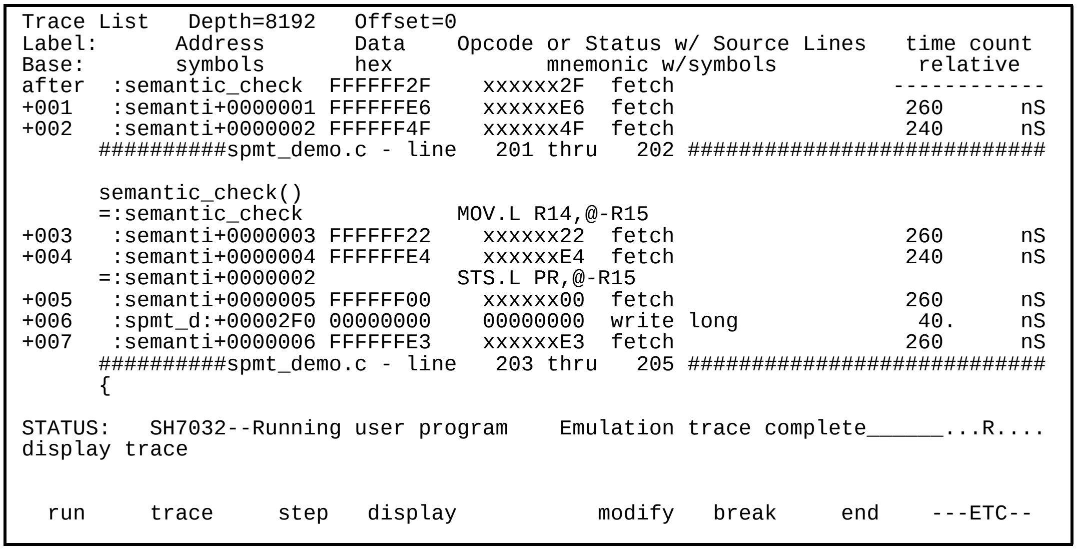
```
run <RETURN>
```

The STATUS message shows "Emulation trace complete.".

Display the Trace

The trace listings which following are of program execution on the SH-7000 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
after  :semantic_check  FFFFFF2F    xxxxxx2F  fetch                 ------------
+001   :semanti+0000001 FFFFFFE6    xxxxxxE6  fetch                  260      nS
+002   :semanti+0000002 FFFFFF4F    xxxxxx4F  fetch                  240      nS
      ##########spmt_demo.c - line   201 thru   202 ############################

      semantic_check()
      =:semantic_check            MOV.L R14,@-R15
+003   :semanti+0000003 FFFFFF22    xxxxxx22  fetch                  260      nS
+004   :semanti+0000004 FFFFFFE4    xxxxxxE4  fetch                  240      nS
      =:semanti+0000002           STS.L PR,@-R15
+005   :semanti+0000005 FFFFFF00    xxxxxx00  fetch                  260      nS
+006   :spmt_d:+00002F0 00000000    00000000  write long              40.     nS
+007   :semanti+0000006 FFFFFFE3    xxxxxxE3  fetch                  260      nS
      ##########spmt_demo.c - line   203 thru   205 ############################
      {

STATUS:   SH7032--Running user program    Emulation trace complete______...R....
display trace


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

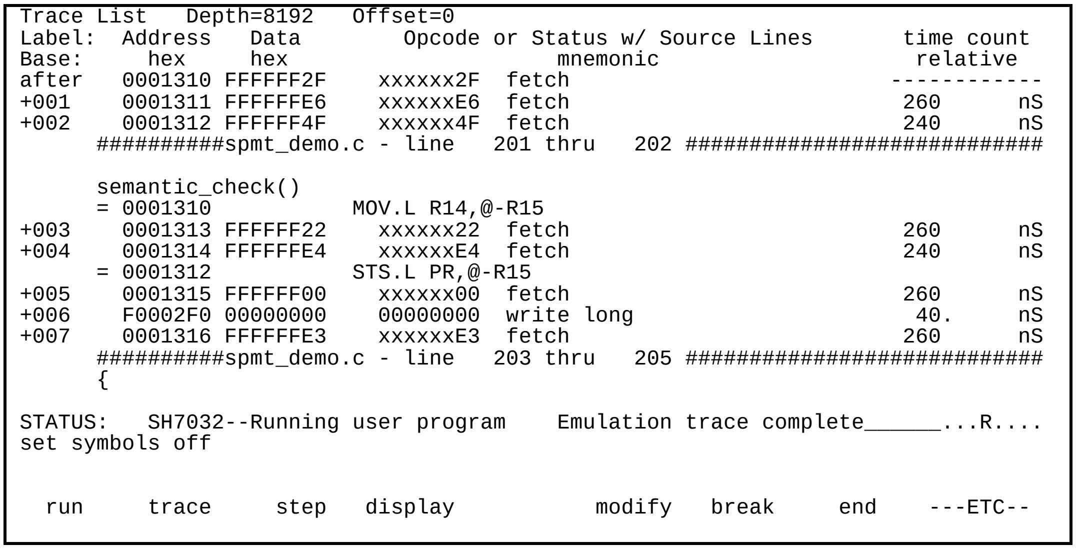
The trace list shows the trace after line (semantic_check()).

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>



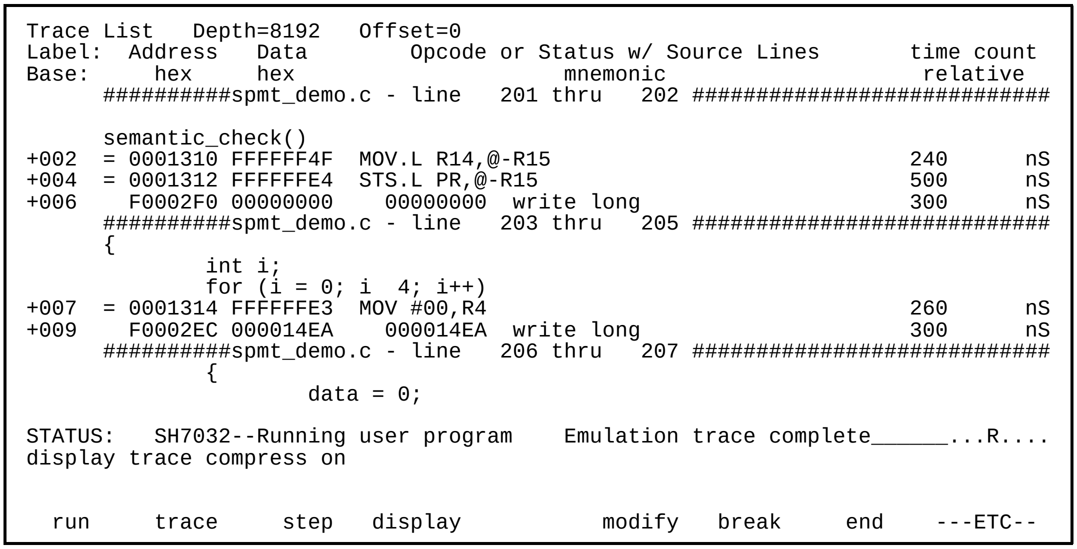
```
Trace List   Depth=8192   Offset=0
Label:  Address   Data        Opcode or Status w/ Source Lines       time count
Base:     hex     hex                   mnemonic                      relative
after   0001310 FFFFFF2F    xxxxxx2F  fetch                         ------------
+001    0001311 FFFFFFE6    xxxxxxE6  fetch                          260      nS
+002    0001312 FFFFFF4F    xxxxxx4F  fetch                          240      nS
      ##########spmt_demo.c - line   201 thru   202 ############################

      semantic_check()
      = 0001310           MOV.L R14,@-R15
+003    0001313 FFFFFF22    xxxxxx22  fetch                          260      nS
+004    0001314 FFFFFFE4    xxxxxxE4  fetch                          240      nS
      = 0001312           STS.L PR,@-R15
+005    0001315 FFFFFF00    xxxxxx00  fetch                          260      nS
+006    F0002F0 00000000    00000000  write long                      40.     nS
+007    0001316 FFFFFFE3    xxxxxxE3  fetch                          260      nS
      ##########spmt_demo.c - line   203 thru   205 ############################
      {

STATUS:   SH7032--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 SH-7000 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   201 thru   202 ############################

      semantic_check()
+002  = 0001310 FFFFFF4F  MOV.L R14,@-R15                            240      nS
+004  = 0001312 FFFFFFE4  STS.L PR,@-R15                             500      nS
+006    F0002F0 00000000    00000000  write long                     300      nS
      ##########spmt_demo.c - line   203 thru   205 ############################
      {
              int i;
              for (i = 0; i  4; i++)
+007  = 0001314 FFFFFFE3  MOV #00,R4                                 260      nS
+009    F0002EC 000014EA    000014EA  write long                     300      nS
      ##########spmt_demo.c - line   206 thru   207 ############################
              {
                      data = 0;

STATUS:   SH7032--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.

Emulator Analysis Status Qualifiers

The following analysis status qualifiers may also be used with the SH-7000 emulator.

Qualifier	Status bits	Description
bg	0xxxxxxxxxxxxxxx0y	Background cycle
byte	0xxxxxxxxxx00x0xxy	Byte memory cycle
cpu	0xxxxxxxxxxxx1xxxy	CPU cycle
data	0xxxxxxxxxxxxx0xxy	Data cycle
dma	0xxxxxxxxxxxx00xxy	DMA cycle
fetch	0xxxxxxxxxxxx111xy	Fetch cycle
fg	0xxxxxxxxxxxxxxx1y	Foreground cycle
grd	00xxxxxxxxxxxxxxxy	Guarded memory access
intack	0xx0xxxxxxxxx111xy	Interrupt acknowledge cycle
long	0xxxxxxxxxx101xxxy	Long word access
read	0xxxxxxxxxxxxxx1xy	Read cycle
refresh	0xxxxxxxxxxxx01xxy	Refresh cycle
word	0xxxxxxxxxx01xxxxy	Word access
write	0xxxxxxxxxxxxx00xy	Write cycle
wrrom	0x0xxxxxxxxxxx00xy	Write to ROM cycle

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 reentering (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. When you enter the Softkey Interface via **pmon** and **MEAS_SYS**, only one window is permitted.

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

Selecting the Measurement System Display or Another Module

When you enter the Softkey Interface via **pmon** and **MEAS_SYS**, you have the option to select the measurement system display or another module in the measurement system when exiting the Softkey Interface. This type of exit is also "locked"; that is, you can continue the emulation session later. For example, to exit and select the measurement system display, enter the following command.

```
end select measurement_system <RETURN>
```

This option is not available if you have entered the Softkey Interface via the **emul700** command.

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.

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.

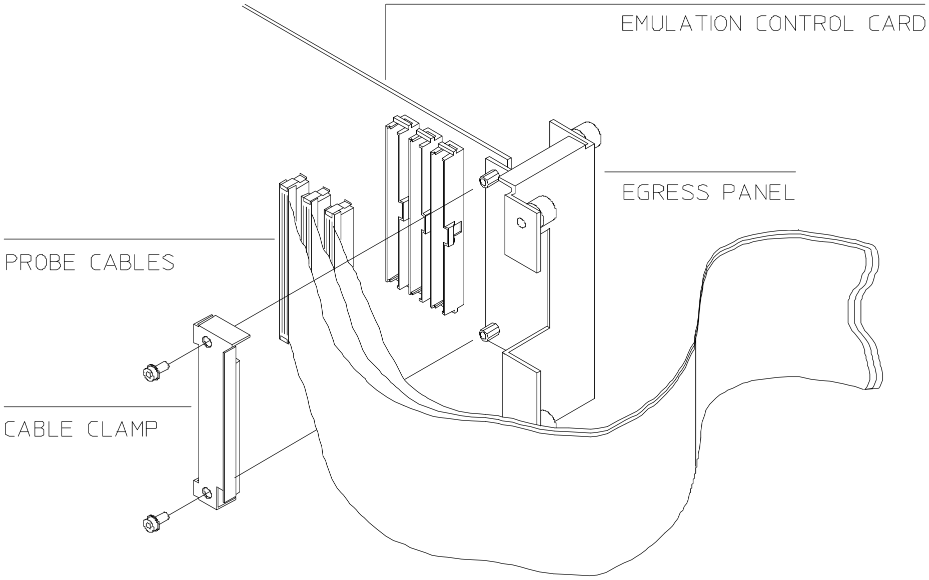


Figure 3-1 Installing cables to the control board

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

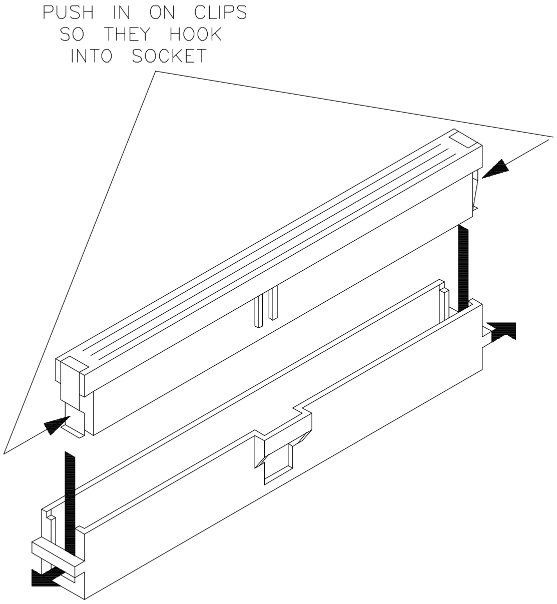


Figure 3-2 Installing cables into cable sockets

3. Connect the other ends of the cables to the emulation probe.

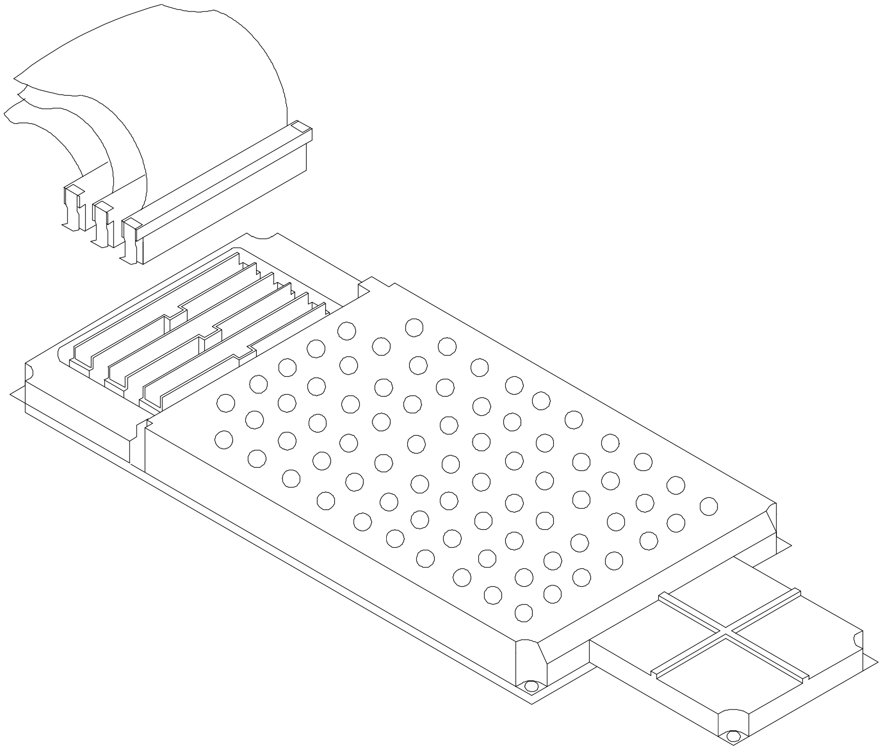


Figure 3-3 Installing cables to the emulation probe

Installing the Emulation Memory Module

There are three 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 emulator probe, and remove the cover. The bottom cover is only removed when you need to replace a defective active probe on the exchange program.

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.

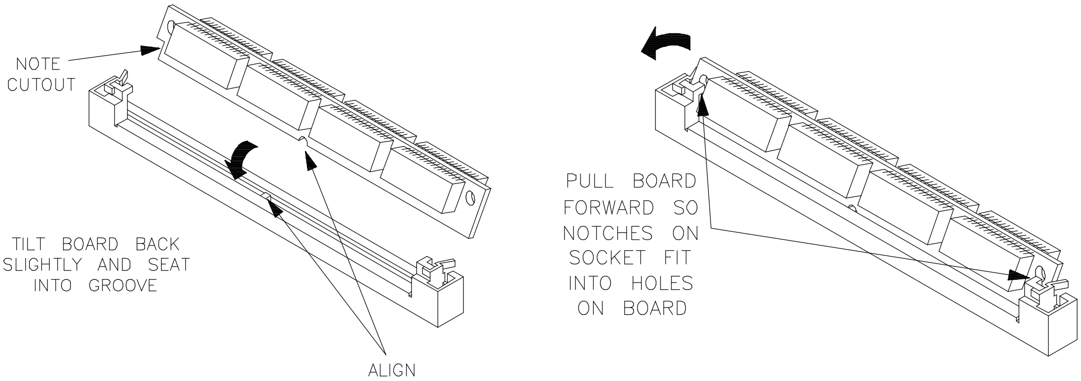

NOTE CUTOUT
TILT BOARD BACK SLIGHTLY AND SEAT INTO GROOVE
ALIGN
PULL BOARD FORWARD SO NOTCHES ON SOCKET FIT INTO HOLES ON BOARD

Figure 3-4 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 64700B front panel. Refer to the *HP 64700 Series Installation/Service* manual.

2. Set up the processor mode switches on the demo target board. You need to set up switches to proper mode which you set up in the emulator configuration.

3. With HP 64700B power OFF, connect the emulation probe to the demo target board as shown in the Figure 4-5. When you install the probe into the demo target board, be careful not to bend any of the pins.

4. 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.

Note

Set up the processor mode switches equal to the processor mode set up in the emulator configuration.

Note

You need to attach the demo target board to the SH-7000 emulator, when you test the SH-7000 emulator using **pv** command.

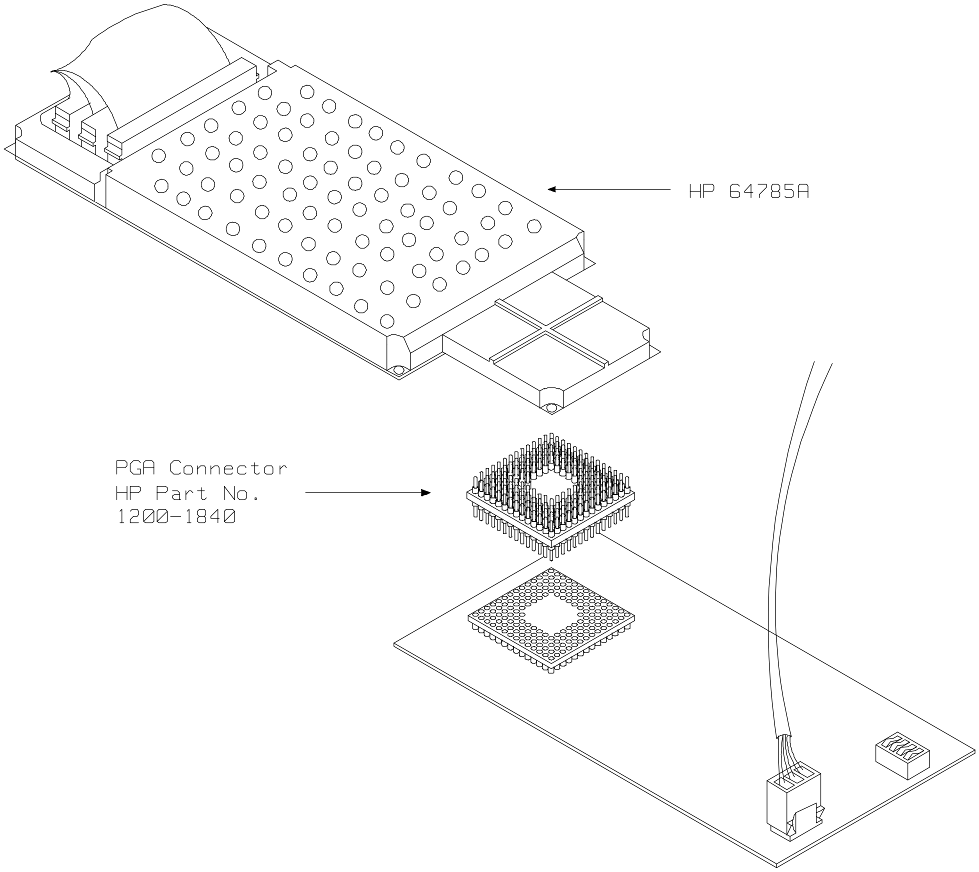


Figure 3-5 Installing the demo target board

Installing into a Target System

The SH-7000 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 emulation 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 emulation 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.

QFP socket/adaptor

The QFP socket/adaptor is provided with the SH-7000 emulator. QFP socket/adaptor is designed for SH-7000 QFP microprocessor. To do in-circuit emulation, you must attach the QFP socket/adaptor to your target system and connect with the SH-7000 emulation probe.

Note

You can order additional QFP socket/adaptor with part No. HP 64784-61611. Contact your local HP sales representative to purchase additional parts.

Installing the emulation probe into your target system

1. Attach the QFP socket/adaptor to your target system.

2. With HP 64700B power OFF, connect the PGA-QFP probe to the emulation probe through the PGA connector.

3. Power OFF your target system, and install the PGA-QFP probe to the QFP socket/adaptor as shown in Figure 4-6.

4. Power ON the emulator first, then power ON your target system.

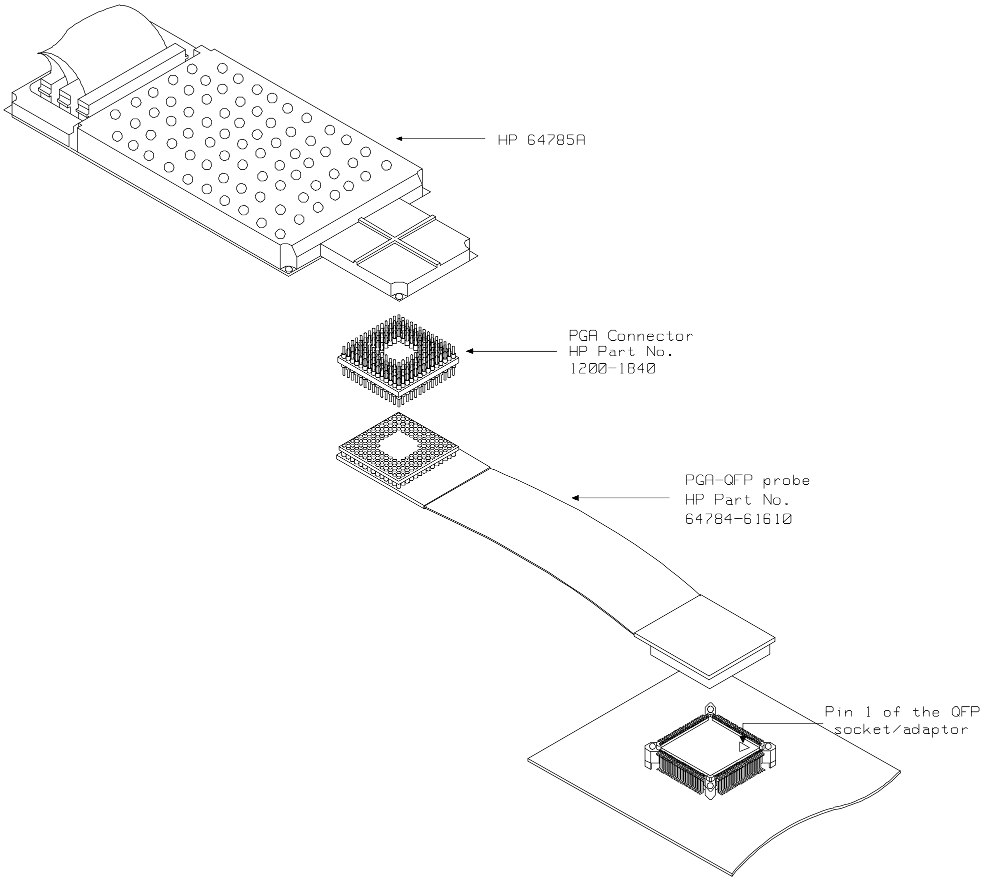


Figure 3-6 Installing into a target system board

In-Circuit configuration

The SH-7000 emulator provides configuration options for the following in-circuit emulation issues. Refer to the "Configuring the Emulator" chapter for more information.

Specifying the pin function of PA8/$\overline{\text{BREQ}}$.

You need to specify whether your target system uses PA8 or $\overline{\text{BREQ}}$ for PA8/BREQ pin. By default, this configuration is set to "PA8".

Runnig the emulation from Target Reset

You can specify that the SH-7000 emulator begins execution from target system reset. When the target system RES line becomes active and then inactive, the SH-7000 emulator will start reset sequence as actual microprocessor.

To specify a run from target system reset, enter the following commnad:

```
run from reset
```

The status now shows that the SH-7000 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 exit this status, you need to enter "rst" command.

Note

You need to break into monitor before running from reset, when you configure ’processor type’ in situations where the emulator can not break.

Reset Types

SH-7000 has two types of resets: power-on reset and manual reset. As Table 4-1 shows, to power OFF the target system always drives the SH-7000 emulator into the power-on reset state. Also, when power ON the target system, a high input at the NMI pin drives the SH-7000 emulator into power-on reset state and a low input at the NMI pin drives the emulator into manual reset state.

Table 4-1 Reset Types

Reset Types	Target System Power		
	OFF	ON	
		NMI	
		High	Low
Power-on reset	O	O	X
Manual reset	X	X	O

Target System Interface and Timing Specification

Refer to the *SH-7000 Terminal Interface User’s Guide* for information on the target system interface and timing specification of the SH-7000 emulator.

4

Configuring the Emulator

Introduction

Your SH-7000 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 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 SH-7000 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 Area 1 memory type.

Memory Configuration:

- Mapping memory.

Emulator Pod Configuration:

- Enabling quick-break mode.
- Specifying reset value for stack pointer.
- Selecting memory access size.
- Selecting PA8/BREQ pin function.

Debug/Trace Configuration:

- Enabling breaks on writes to ROM.
- Enabling setting breakpoints at delay slot.
- Specifying tracing of user program/emulation monitor cycles.
- Enabling tracing on-chip DMAC cycles.
- Enabling tracing refresh 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 "reset", "break", and "step" commands; you should use caution in executing these commands.

Processor type?

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

7032	The SH-7032 microprocessor is emulated.
7034	The SH-7034 microprocessor is emulated.

Note

If the emulation processor without on-chip ROM is selected and the processor operation mode is configured as **mode_2**, the emulator will ignore the mode configuration option and the emulation processor will be operated in **mode_0**.

Note

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

Note

When you change this configuration, you need to break into monitor once. Usually, changing this configuration will drive the emulator into monitor automatically, then drive it into a reset state. In situations without clock source, you need to break it, explicitly.

Processor operation mode?

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

mode_0	The emulator will operate in mode 0.
mode_1	The emulator will operate in mode 1.
mode_2	The emulator will operate in mode 2.

Note

If **mode_2** and the processor which has no on-chip ROM are selected, the emulator will ignore this mode configuration option and the emulation processor will be operated in **mode_0**.

Note

When you configure to emulate **7032** microprocessor, **mode_2** does not appear in this configuration item.

Note

You need to supply operation mode signal same as this configuration from the target system.

Note

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

Area 1 memory type?

This configuration allows you to select the memory type of the area1.

dram If the area 1 is used as dynamic RAM space in your target system, set ’dram’ to this configuration option. The memory mapper will treat the area 1 as 16-Mbyte address space.

other If the area 1 is used as other memory space, set ’other’ to this configuration option. The memory mapper will treat the area 1 as 4-Mbyte address space.

Note

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

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 16 memory range (at 16K byte boundaries and at least 16K byte length).

The memory mapper allows you to characterize memory locations. It allows you to 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.

Note

The internal RAM/ROM area and all registers of on-chip peripheral modules are mapped automatically. And the emulation memory system does not introduce these areas in memory mapping display.

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

- When you use 1M byte memory module and characterize memory range which does not override 32K byte boundary as emulation memory, 64K byte is used.

Also when you use 4M byte memory module and characterize memory range which does not override 128K byte boundary, 128K byte is used.

Note ☝

Direct memory access to the emulation memory by external DMAC is not allowed. Also, single address mode transfer to the emulation memory by internal DMAC is not allowed.

Note ☝

The emulation memory has no parity bit. You can not check and generate parity for emulation memory.

Note ☝

The SH-7000 emualtor ignores memory mapping for address/data multiplexed I/O space. Address/data multiplexed I/O space is always accessed as target RAM. However, when you map this area as guarded memory, you can not access this area by commands.

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 quick-break mode?

This question allows you to specify whether the quick-break is enabled or disabled.

yes The emulator enables quick-break mode. In this configuration, temporary break to the monitor for an operation such as display registers will spend a very short time in the monitor. The CMB does not work in this setting.

no The emulator disables quick-break mode. In this configuration, temporary break to the monitor will spend more time in the monitor.

Note

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

Reset value for stack pointer?

Specify the value that the stack pointer will be set to when the monitor is entered after an emulation reset. This configuration option has no effect when a run from reset command is given.

The value of the stack pointer must be long word aligned.

Memory access size?

This configuration specifies the type of microprocessor cycles that are used by the monitor program to access memory locations. When a command requests the monitor to read or write to memory location, 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 memory using byte cycles (one byte at a time).

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

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

Note

When the access size is **words**, modifying memory will fail if you try to modify memory from odd address or with data which byte count is odd. Also, you can’t load file which byte count is odd. Therefore, it is recommended to use the emulator with default **any** or **bytes** in this configuration.

PA8/BREQ pin function?

This configuration option specifies the function of PA8/BREQ pin.

breq If the PA8/BREQ pin is used as /BREQ input in your target system, set ’breq’ to this configuration option.

pa8 If the PA8/BREQ pin is used as PA8 input/output or is not used in your target system, set ’pa8’ to this configuration option.

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>
```

Enable setting breakpoints at delay slot?

A software breakpoint at delay slot causes slot invalid instruction exception in your program.

yes When setting software breakpoints at delay slot is enabled, you can set software breakpoints at any location.

no The breakpoint command will check if the instruction before the requested breakpoint address is a delayed branch or not. And, if the instruction is a delayed branch, the command will fail.

Trace background or foreground operation?

This configuration option allows you to specify whether the analyzer trace only user program, only monitor program or both monitor and user program.

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

background Specifies that the analyzer trace only monitor cycles. (This is rarely 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.

Trace on-chip DMAC cycles?

This configuration option allows you to specify whether the analyzer trace on-chip DMAC cycles or not.

yes Specifies that the analyzer traces on-chip DMAC cycles.

no Specifies that the analyzer does not trace on-chip DMAC cycles.

Note

Address error by internal DMAC in monitor is suspended and occurs after when context is changed to user program.

Note

When trace on-chip DMAC cycles is no, the emulator will not break to monitor upon a write to ROM or guarded memory by internal DMAC.

Trace refresh cycles?

This configuration option allows you to specify whether the analyzer trace refresh cycles or not.

yes Specifies that the analyzer traces refresh cycles.

no Specifies that the analyzer does not trace refresh cycles.

Emulation analyzer speed?

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. You must answer this question, when you use HP 64704A emulation bus analyzer.

slow Specifies the processor clock speed is less than or equal to 16.6MHz. Both state and time counting are available.

fast Specifies the processor clock speed is greater than 16.6MHz. 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
- Analyzer topics
  - Specifying data for trigger or store condition
- Features available via "pod_command"

This chapter shows you how to:

Emulation memory access

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

REGISTER CLASS and NAME

Summary

SH-7000 register designator. All available register class names and register names are listed below.

<REG_CLASS>

<REG_NAME>	Description
***(All basic registers)**	
PC	Program counter
SR	Status register
R0	General register R0
R1	General register R1
R2	General register R2
R3	General register R3
R4	General register R4
R5	General register R5
R6	General register R6
R7	General register R7
R8	General register R8
R9	General register R9
R10	General register R10
R11	General register R11
R12	General register R12
R13	General register R13
R14	General register R14
R15	General register R15
SP	Stack pointer
GBR	Global base register
VBR	Vector base register
PR	Procedure register
MACH	Multiply and accumulate register high
MACL	Multiply and accumulate register low

INTC(Interrupt controller)

IPRA	Interrupt priority register A
IPRB	Interrupt priority register B
IPRC	Interrupt priority register C
IPRD	Interrupt priority register D
IPRE	Interrupt priority register E
ICR	Interrupt control register

UBC(User break controller)

BAR	Break address register
BAMR	Break address mask register
BBR	Break bus cycle register

BSC(Bus state controller)

BCR	Bus control register
WCR1	Wait state control register 1
WCR2	Wait state control register 2
WCR3	Wait state control register 3
DCR	DRAM area control register
PCR	Parity control register
RCR	Refresh control register
RTCSR	Refresh timer control/status register
RTCNT	Refresh timer counter
RTCOR	Refresh time constant register

DMAC0(Direct memory access controller 0)

SAR0	DMA source address register 0
DAR0	DMA destination register 0
DMATCR0	DMA transfer count register 0
CHCR0	DMA channel control register 0
DMAOR	DMA operation register

DMAC1(Direct memory access controller 1)

SAR1	DMA source address register 1
DAR1	DMA destination register 1
DMATCR1	DMA transfer count register 1
CHCR1	DMA channel control register 1

DMAC2(Direct memory access controller 2)

SAR2	DMA source address register 2
DAR2	DMA destination register 2
DMATCR2	DMA transfer count register 2
CHCR2	DMA channel control register 2

DMAC3(Direct memory access controller 3)

SAR3	DMA source address register 3
DAR3	DMA destination register 3
DMATCR3	DMA transfer count register 3
CHCR3	DMA channel control register 3

ITUG(Integrated-timer pulse unit general)

TSTR	Timer start register
TSNC	Timer synchro register
TMDR	Timer mode register
TFCR	Timer function control register
TOCR	Timer output control register

ITU0(Integrated-timer pulse unit 0)

TCR0	Timer control register 0
TIOR0	Timer I/O register 0
TIER0	Timer interrupt enable register 0
TSR0	Timer status register 0
TCNT0	Timer counter 0
GRA0	General register A0
GRB0	General register B0

ITU1(Integrated-timer pulse unit 1)

TCR1	Timer control register 1
TIOR1	Timer I/O register 1
TIER1	Timer interrupt enable register 1
TSR1	Timer status register 1
TCNT1	Timer counter 1
GRA1	General register A1
GRB1	General register B1

ITU2(Integrated-timer pulse unit 2)

TCR2	Timer control register 2
TIOR2	Timer I/O register 2
TIER2	Timer interrupt enable register 2
TSR2	Timer status register 2
TCNT2	Timer counter 2
GRA2	General register A2
GRB2	General register B2

ITU3(Integrated-timer pulse unit 3)

TCR3	Timer control register 3
TIOR3	Timer I/O register 3
TIER3	Timer interrupt enable register 3
TSR3	Timer status register 3
TCNT3	Timer counter 3
GRA3	General register A3
GRB3	General register B3

ITU4(Integrated-timer pulse unit 4)

TCR4	Timer control register 4
TIOR4	Timer I/O register 4
TIER4	Timer interrupt enable register 4
TSR4	Timer status register 4
TCNT4	Timer counter 4
GRA4	General register A4
GRB4	General register B4

TPC(Programmable timing pattern controller)

TPMR	TPC output mode register
TPCR	TPC output control register
NDERA	Next data enable register A
NDERB	Next data enable register B
NDRA	Next data register A (address 5fffff5H)
NDRA0	Next data register A (address 5fffff7H)
NDRB	Next data register B (address 5fffff4H)
NDRB2	Next data register B (address 5fffff6H)

WDT(Watchdog timer)

WDTCSR	Timer control/status register
WDTCNT	Timer counter
RSTCSR	Reset control/status register

SCI0(Serial communication interface 0)

SMR0	Serial mode register 0	
BRR0	Bit rate register 0	
SCR0	Serial control register 0	
TDR0	Transmit data register 0	
SSR0	Serial status register 0	
RDR0	Receive data register 0	(Read Only)

SCI1(Serial communication interface 1)

SMR1	Serial mode register 1	
BRR1	Bit rate register 1	
SCR1	Serial control register 1	
TDR1	Transmit data register 1	
SSR1	Serial status register 1	
RDR1	Receive data register 1	(Read Only)

ADC(A/D converter)

ADDRA	A/D data register A	(Read Only)
ADDRB	A/D data register B	(Read Only)
ADDRC	A/D data register C	(Read Only)
ADDRD	A/D data register D	(Read Only)
ADDSR	A/D control/status register D	
ADCR	A/D control register	

PFC(Pin function controller)

PAIOR	Port A I/O register
PBIOR	Port B I/O register
PACR1	Port A control register 1
PACR2	Port A control register 2
PBCR1	Port B control register 1
PBCR2	Port B control register 2
CASCR	Column address strobe pin control register

PORT(Parallel I/O port)

PADR	Port A data register
PBDR	Port B data register
PCDR	Port C data register

SYS(System control)

SBYCR	System control register

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.

Analyzer Topics

Specifying Data for Trigger or Store Condition

You may want to trigger the emulation analyzer when specific data appears on the data bus. You can accomplish this with the following command.

```
trace after data <data>
```

There are some points to be noticed when you trigger the analyzer to 32 bits bus area in this way. You need to specify the <data> with 32 bits value shown in Table 5-1. This is because the analyzer is designed so that it can capture data on internal data bus (which has 32 bits width).

Table 5-1 Trigger for 32 bit bus area

Address Value	Byte Access	Word Access
4N [*1]	ddxxxxxx [*2]	ddddxxxx [*2]
4N+1 [*1]	0xxddxxxx [*2]	-
4N+2 [*1]	0xxxxddxx [*2]	0xxxxdddd [*2]
4N+3 [*1]	0xxxxxxdd [*2]	-

```
*1 N means random value
*2 dd and dddd mean data value
```

Note that you always need to specify "xx" value to identify byte/word values on the 32 bit data bus. Be careful to trigger the analyzer by data.

When you trigger the analyzer to 8/16 bits bus area, you can capture same way as the SH-7000 microprocessor.

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

Accessing Emulation Memory

Usually, the emulation memory is accessed by monitor program. Even if the emulation is reset state, the emulation memory can be accessed.

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.

Notes

Index