HP 64785

SH-7000 Emulator Terminal Interface

User's Guide

HEWLETT PACKARD

HP Part No. 64785-97000
Printed in Japan
June 1994

Edition 1

Notice

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

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 **64785-97000, June 1994**

Using this Manual

This manual will show you how to use HP 64785A SH-7000 emulator with the Terminal Interface.

This manual will:

- Show you how to use emulation commands by executing them on a sample program and describing their results.
- Show you how to configure the emulator for your development needs.
- Show you how to use the emulator in-circuit (connected to a demo board and target system).
- Describe the command syntax which is specific to the SH-7000 emulator.

This manual will not:

- Describe every available option to the emulation commands; this is done in the *HP 64700 Emulators Terminal Interface: User's Reference*.

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 programs, run programs, use software breakpoints, and search memory for data.

Chapter 3 **Using the Emulator**. This chapter shows you how to: restrict the emulator to real-time execution, use the analyzer, and run the emulator from target system reset.

Chapter 4 **In-Circuit Emulation Topics**. This chapter shows you how to: install the emulator probe into a demo board and target system.

Appendix A **SH-7000 Emulator Specific Command Syntax**. This appendix describes the command syntax which is specific to the SH-7000 emulator. Included are: emulator configuration items, display and access modes, register class and name.

Contents

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.

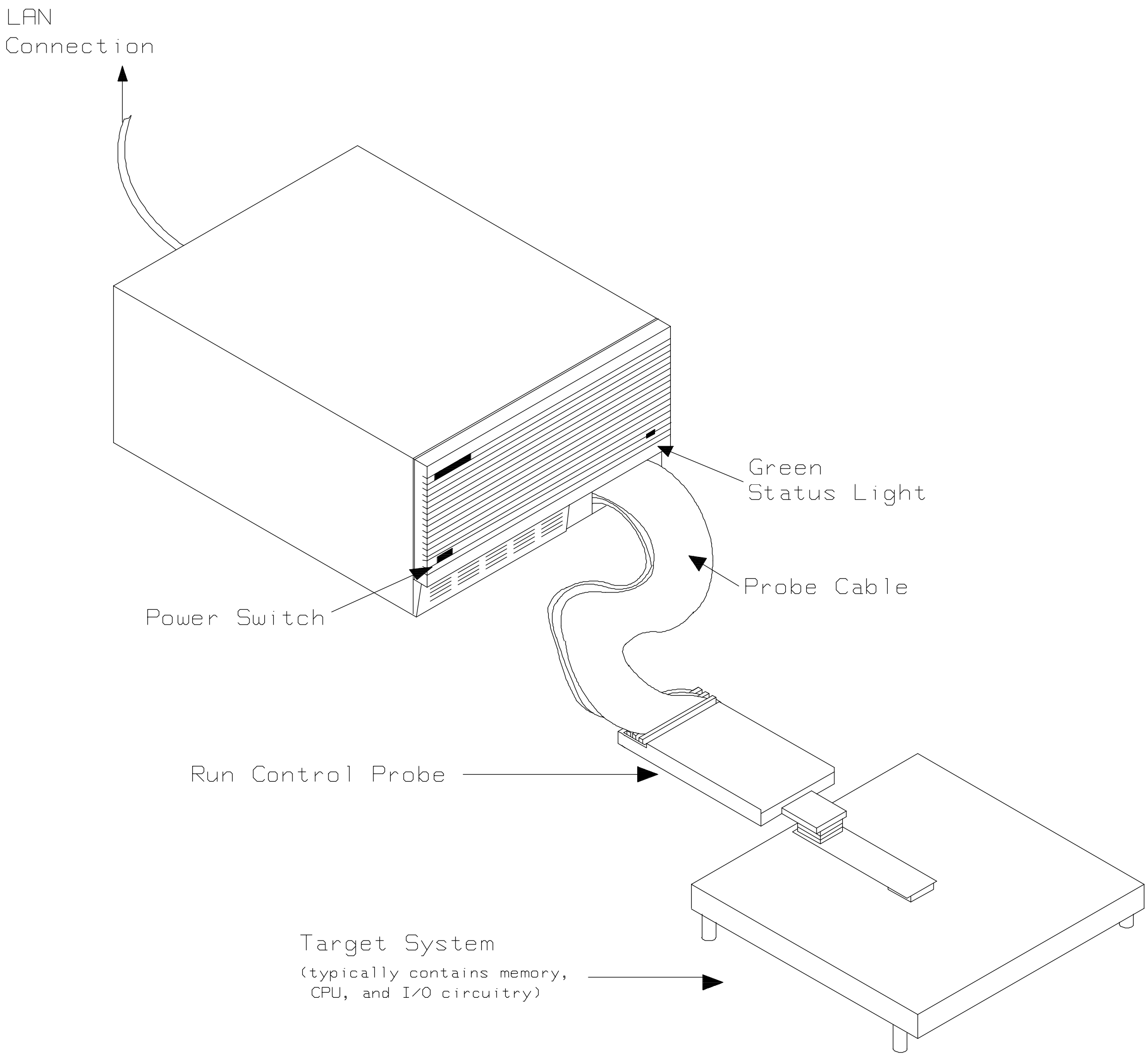


Figure 1-1 HP 64785A Emulator for SH-7000

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 emulator for the SH-7000 microprocessor.

This chapter will:

- Describe the sample program used for this chapter's examples.
- Show you how to use the "help" facility.
- Show you how to use the memory mapper.
- Show you how to enter emulation commands to view execution of the sample program. The commands described in this chapter include:
  - Displaying and modifying memory
  - Stepping
  - Displaying registers
  - Defining macros
  - Searching memory
  - Running
  - Breaking
  - Using software breakpoints
  - Using the Analyzer
- Show you how to reset the emulator.

Before You Begin

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

1. Completed hardware installation of the HP64700 emulator in the configuration you intend to use for your work:
   - Standalone configuration
   - Transparent configuration
   - Remote configuration
   - Local Area Network configuration

   References: *HP 64700 Series Installation/Service* manual

2. If you are using the Remote configuration, you must have completed installation and configuration of a terminal emulator program which will allow your host to act as a terminal connected to the emulator. In addition, you must start the terminal emulator program before you can work the examples in this chapter.

3. If you have properly completed steps 1 and 2 above, you should be able to hit <RETURN> (or <ENTER> on some keyboards) and get one of the following command prompts on your terminal screen:

```
U>
R>
M>
```

   If you do not see one of these command prompts, retrace your steps through the hardware and software installation procedures outlined in the manuals above, verifying all connections and procedural steps.

   In any case, you **must** have a command prompt on your terminal screen before proceeding with the tutorial.

A Look at the Sample Program

The sample program used in this chapter is listed in figure 2-1. The program emulates a primitive command interpreter.

```
                .GLOBAL         Init,Msgs,Cmd_Input
                .GLOBAL         Msg_Dest

                .SECTION        Table,DATA
Msgs
Msg_A           .SDATA          "THIS IS MESSAGE A"
Msg_B           .SDATA          "THIS IS MESSAGE B"
Msg_I           .SDATA          "INVALID COMMAND"
End_Msgs

                .SECTION        Prog,CODE
;*****************************************************
;* Set up the Stack Pointer.
;*****************************************************
Init            MOV             #0,R15
;*****************************************************
;* Clear previous command.
;*****************************************************
Clear           MOV             #H'00,R0
                MOV.L           @(cmd_input-$-4,PC),R1
                MOV.B           R0,@R1
;*****************************************************
;* Read command input byte.  If no command has been
;* entered, continue to scan for it.
;*****************************************************
Scan            MOV.B           @R1,R0
                CMP/EQ          #H'00,R0
                BT              Scan
;*****************************************************
;* A command has been entered.  Check if it is
;* command A, command B, or invalid command.
;*****************************************************
Exe_Cmd         CMP/EQ          #H'41,R0
                BT              Cmd_A
                CMP/EQ          #H'42,R0
                BT              Cmd_B
                BF              Cmd_I
;*****************************************************
;* Command A is entered.  R3 = the number of bytes
;* in message A.  R4 = location of the message.
;* Jump to the routine which writes the message.
;*****************************************************
Cmd_A           MOV.L           @(msg_a-$-4,PC),R4
                BRA             Write_Msg
                MOV             #Msg_B-Msg_A,R3
;*****************************************************
;* Command B is entered.
;*****************************************************
Cmd_B           MOV.L           @(msg_b-$-2,PC),R4
                BRA             Write_Msg
                MOV             #Msg_I-Msg_B,R3
;*****************************************************
;* An invalid command is entered.
;*****************************************************
Cmd_I           MOV             #End_Msgs-Msg_I,R3
                MOV.L           @(msg_i-$-2,PC),R4
;*****************************************************
```

Figure 2-1 Sample program listing

```
;* The destination area is cleared.
;****************************************************
Write_Msg       MOV.L           @(msg_dest-$-4,PC),R5
Clear_Old       MOV             R5,R6
                ADD             #h'20,R6
                MOV             #h'00,R0
Clear_Loop      MOV.B           R0,@R5
                ADD             #1,R5
                CMP/EQ          R5,R6
                BF              Clear_Loop
;****************************************************
;* Message is written to the destination.
;****************************************************
                MOV.L           @(msg_dest-$-4,PC),R5
                MOV             R5,R6
                ADD             R3,R6
Write_Loop      MOV.B           @R4+,R0
                MOV.B           R0,@R5
                ADD             #1,R5
                CMP/EQ          R5,R6
                BF              Write_Loop
;****************************************************
;* Go back and scan for next command.
;****************************************************
                BT              Clear

                .ALIGN          4
cmd_input       .DATA.L         Cmd_Input
msg_dest        .DATA.L         Msg_Dest
msg_a           .DATA.L         Msg_A
msg_b           .DATA.L         Msg_B
msg_i           .DATA.L         Msg_I

                .SECTION        Data,DATA
;****************************************************
;* Command input byte.
;****************************************************
Cmd_Input       .RES.B          1
                .RES.B          1
;****************************************************
;* Destination of the command messages.
;****************************************************
Msg_Dest        .RES.W          H'80

                .END            Init
```

Figure 2-1 Sample program listing (Cont'd)

Data Declarations

The area at Table section defines the messages used by the program to respond to various command inputs. These messages are labeled **Msg_A**, **Msg_B**, and **Msg_I**.

Initialization

The program instructions from the **Init** label to the **Clear** label perform initialization. The segment registers are loaded and the stack pointer is set up.

Reading Input

The instruction at the **Clear** label clears any random data or previous commands from the **Cmd_Input** byte. The **Scan** loop continually reads the **Cmd_Input** byte to see if a command is entered (a value other than 0H).

Processing Commands

When a command is entered, the instructions from **Exe_Cmd** to **Cmd_A** determine whether the command was "A", "B", or an invalid command.

If the command input byte is "A" (ASCII 41H), execution is transferred to the instructions at **Cmd_A**.

If the command input byte is "B" (ASCII 42H), execution is transferred to the instructions at **Cmd_B**.

If the command input byte is neither "A" nor "B", i.e. an invalid command has been entered, then execution is transferred to the instructions at **Cmd_I**.

The instructions at **Cmd_A**, **Cmd_B**, and **Cmd_I** load register R3 with the length location of the message to be displayed and register R4 with the starting location of the appropriate message. Then, execution transfers to **Write_Msg** where the appropriate message is written to the destination location, **Msg_Dest**. Then, the program jumps back to read the next command.

Destination Area

The area at Data section declares memory storage for the command input byte, and the destination area.

Using the "help" Facility

The HP 64700 Series emulator's Terminal Interface provides an excellent help facility to provide you with quick information about the various commands and their options. From any system prompt, you can enter "**help**" or "**?**" as shown below.

R>**help**

```
help  - display help information

   help <group>         - print help for desired group
   help -s <group>      - print short help for desired group
   help <command>       - print help for desired command
   help                 - print this help screen

 --- VALID <group> NAMES ---
   gram     - system grammar
   proc     - processor specific grammar

   sys      - system commands
   emul     - emulation commands
   hl       - highlevel commands (hp internal use only)
   trc      - analyzer trace commands
   *        - all command groups
```

Commands are grouped into various classes. To see the commands grouped into a particular class, you can use the help command with that group. Viewing the group help information in short form will cause the commands or the grammar to be listed without any description.

You can type **?** symbol instead of typing **help**. For example, if you want to get some information for group gram, enter "**? gram**". Following help information should be displayed.

R>**? gram**

```
gram - system grammar
-----------------------------------------------------------------------------
--- SPECIAL CHARACTERS ---
  #  - comment delimiter    ;  - command separator    Ctl C - abort signal
  {} - command grouping     "" - ascii string         ''    - ascii string
  Ctl R - command recall    Ctl B - recall backwards

--- EXPRESSION EVALUATOR ---
  number bases:  t-ten   y-binary   q-octal   o-octal   h-hex
  repetition and time counts default to decimal - all else default to hex
  operators:     ()  ~  *  /  %  +  -  <  <<  >  >>  &  ^  |  &&

--- PARAMETER SUBSTITUTION ---
  &token& - pseudo-parameter included in macro definition
          - cannot contain any white space between & pairs
          - performs positional substitution when macro is invoked
  Example
     Macro definition:  mac getfile={load -hbs"transfer -t &file&"}
     Macro invocation:  getfile MYFILE.o
     Expanded command:  load -hbs"transfer -t MYFILE.o"
```

Help information exists for each command. Additionally, there is help information for each of the emulator configuration items.

Becoming Familiar with the System Prompts

A number of prompts are used by the HP 64700 Series emulators. Each of them has a different meaning, and contains information about the status of the emulator before and after the commands execute. These prompts may seem cryptic at first, but there are two ways you can find out what a certain prompt means.

Using "? proc" to View Prompt Description

The first way you can find information on the various system prompts is to look at the **proc** help text.

```
R>? proc
```

```
--- Emulation Prompt Status Characters ---
  R - emulator in reset state           p - no target system power
  U - running user program              c - no target system clock
  M - running monitor program           r - target system reset active
  W - waiting for CMB to become ready   g - bus granted
  T - waiting for target system reset   b - no bus cycles
  ? - unknown state                     s - processor sleep or standby

--- Analyzer STATUS Field Equates ---
  fetch - instruction fetch        cpu     - CPU cycle
  data  - data access              dma     - on-chip DMAC cycle
  read  - read                     intack  - interrupt acknowledge
  write - write                    refresh - refresh cycle
  byte  - byte access              wrrom   - write to rom
  word  - word access              grd     - guarded memory access
  long  - long access              bg      - background
                                   fg      - foreground
```

Using the Emulation Status Command (es) for Description of Current Prompt

When using the emulator, you will notice that the prompt changes after entering certain commands. If you are not familiar with a new prompt and would like information about that prompt only, enter the **es** (emulation status) command for more information about the current status.

```
M>es
SH7032--Running in monitor
```

Initializing the Emulator

If you plan to follow this tutorial by entering commands on your emulator as shown in this chapter, verify that no one else is using the emulator. To initialize the emulator, enter the following command:

```
R>init
# Limited initialization completed
```

The **init** command with no options causes a limited initialization, also known as a warm start initialization. Warm start initialization does not affect system configuration. However, the **init** command will reset emulator and analyzer configurations. The **init** command:

- Resets the memory map.
- Resets the emulator configuration items.
- Resets the break conditions.
- Clears software breakpoints.

The **init** command does not:

- Clear any macros.
- Clear any emulation memory locations; mapper terms are deleted, but if you respecify the same mapper terms, you will find that the emulation memory contents are the same.

Set Up the Proper Emulation Configuration

Emulation configuration is needed to adapting to your specific development. As you have initialized the emulator, the emulation configuration items have default value.

Set Up Emulation Condition

The emulator allows you to set the emulator’s configuration setting with the **cf** command. Enter the **? cf** to view the information with the configuration command.

```
R>? cf
```

```
cf - display or set emulation configuration

  cf                - display current settings for all config items
  cf <item>         - display current setting for specified <item>
  cf <item>=<value> - set new <value> for specified <item>
  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
```

To view the current emulator configuration setting, enter the following command.

```
R>cf
```

```
cf area1=other
cf bpds=dis
cf breq=dis
cf chip=7032
cf mode=0
cf qbrk=dis
cf rrt=dis
cf rsp=0
cf tdma=en
cf trfsh=en
```

The individual configuration items won’t be explained in this section; refer to the "CONFIG_ITEMS" in the "SH-7000 Emulator Specific Command Syntax" appendix for details.

Mapping Memory

Depending on the memory module, emulation memory consists of 256K, 1M, or 4M bytes.

The memory mapper allows you to characterize memory locations. The minimum amount of emulation memory that can be allocated to a range is 16K byte. 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

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.

Blocks of memory can also be characterized as guarded memory. Guarded memory accesses will generate "break to monitor" requests. Writes to ROM will also generate "break to monitor" requests if the **rom** break condition is enabled. Memory is mapped with the **map** command. To view the memory mapping options, enter:

```
M>? map
```

```
map - display or modify the processor memory map

  map                        - display the current map structure
  map <addr>..<addr> <type>  - define address range as memory type
  map other <type>           - define all other ranges as memory type
  map -d <term#>             - delete specified map term
  map -d *                   - delete all map terms

--- VALID <type> OPTIONS ---
  eram  -  emulation ram
  erom  -  emulation rom
  tram  -  target ram
  trom  -  target rom
  grd   -  guarded memory
```

Enter the **map** command with no options to view the default map structure.

```
M>map
# remaining number of terms   : 16
# remaining emulation memory  : 100000h bytes
map  other tram
```

Which Memory Locations Should be Mapped?

Typically, assemblers generate relocatable files and linkers combine relocatable files to form the absolute file. A linker load map listing will show what memory locations your program will occupy. One for the sample program is shown below.

```
SECTION   NAME                             START      -      END        LENGTH

Prog                                    H'00001000    -   H'0000105f   H'00000060
Table                                   H'00001060    -   H'00001090   H'00000031
Data                                    H'0f000000    -   H'0f000101   H'00000102
```

From the load map listing, you can see that the sample program occupies two address ranges. The program and table area occupy locations 1000 through 1090 hex. The destination area, which contains the command input byte and the locations of the message destination, occupies locations 0f000000 through 0f000101 hex. For this sample program, map the address from 1000 through 3fff hex as emulation ROM. Since internal RAM/ROM area is automatically mapped by the emulator, you don't need to map these area. Enter the following commands to map sample program and display the memory map.

```
R>map 1000..3fff erom
R>map
# remaining number of terms   : 15
# remaining emulation memory  : f8000h bytes
map  00000000..00003fff  erom    # term 1
map  other tram
```

Note

You don't have to map internal RAM/ROM and all registers of the on-chip peripheral modules. The SH-7000 emulator has memory and maps them automatically. And the emulator memory system does not introduce them in memory mapping display.

When mapping memory for your target system programs, you should characterize emulation memory locations containing programs and constants (locations which should not be written) as ROM. This will prevent programs and constants from being written over accidentally.

Break will occur when instructions or commands attempt to do so(if the **rom** break condition is enabled).

Note

The defaults number base for address and data values within HP 64700 Terminal Interface is hexadecimal. Other number bases may be specified. Refer to the "Expressions" chapter or the *HP 64700 Terminal Interface Reference* manual for further details.

Getting the Sample Program into Emulation Memory

This section assumes you are using the emulator in one of the following three configurations:

1. Connected only to a terminal, which is called the *standalone* configuration. In the standalone configuration, you must modify memory to load the sample program.

2. Connected between a terminal and a host computer, which is called the *transparent* configuration. In the transparent configuration, you can load the sample program by downloading from the "other" port.

3. Connected to a host computer and accessed via a terminal emulation program. This configurations is called *remote* configurations. In the remote configuration, you can load the sample program by downloading from the same port.

Standalone Configuration

If you are operating the emulator in the standalone configuration, the only way to load the sample program into emulation memory is by modifying emulation memory locations with the **m** (memory display/modification) command.

You can enter both of program and data area of the sample program into memory with the **m** command as shown below.

```
R> m 00001000..0000100f=0ef,00,0e0,00,0d1,11,21,00,60,10,88,00,89,0fc,88,41
R> m 00001010..0000101f=89,02,88,42,89,03,8b,05,0d4,0e,0a0,05,0e3,11,0d4,0e
```

```
R> m 00001020..0000102f=0a0,02,0e3,11,0e3,0f,0d4,0d,0d5,09,66,53,76,20,0e0,00
R> m 00001030..0000103f=25,00,75,01,36,50,8b,0fb,0d5,05,66,53,36,3c,60,44
R> m 00001040..0000104f=25,00,75,01,36,50,8b,0fa,89,0db,00,09,0f,00,00,00
R> m 00001050..0000105f=0f,00,00,02,00,00,10,60,00,00,10,71,00,00,10,82
R> m 00001060..0000106f=54,48,49,53,20,49,53,20,4d,45,53,53,41,47,45,20
R> m 00001070..0000107f=41,54,48,49,53,20,49,53,20,4d,45,53,53,41,47,45
R> m 00001080..0000108f=20,42,49,4e,56,41,4c,49,44,20,43,4f,4d,4d,41,4e
R> m 00001090=44
```

(note the hex letters must be preceded by a digit)

You can also enter data area of sample program into memory by the following method.

```
R> m 00001060..00001090="THIS IS MESSAGE ATHIS IS MESSAGE BINVALID COMMAND"
```

After entering the opcodes and operands, you would typically display memory in mnemonic format to verify that the values entered are correct (see the example below). If any errors exist, you can modify individual locations.

Note

Be careful about using this method to enter programs from the listings of relocatable source files. If source files appear in relocatable sections, the address values of references to locations in other relocatable sections are not resolved until link-time. The correct values of these address operands will not appear in the assembler listing.

Transparent Configuration

If your emulator is connected between a terminal and a host computer, you can download programs into memory using the **load** command with the **-o** (from other port) option. The **load** command will accept absolute files in the following formats:

- HP absolute.
- Intel hexadecimal.
- Tektronix hexadecimal.
- Motorola S-records.

The examples which follow will show you the methods used to download HP absolute files and the other types of absolute files.

HP Absolutes

Downloading If you have a Softkey Interface, a file format converter is provided with it. The converter can convert Hitachi format files to HP Absolute files. (Refer to Softkey Interface User's Guide for more details) Downloading the HP Absolute requires the **transfer** protocol. The example below assumes that the **transfer** utility has been installed on the host computer (HP 64884 for HP 9000 Series 500, or HP 64885 for HP 9000 Series 300).

Note

Notice that the transfer command on the host computer is terminated with the <ESCAPE>**g** characters; by default, these are the characters which temporarily suspend the transparent mode to allow the emulator to receive data or commands.

```
R>load -hbo <RETURN> <RETURN>
$ transfer -rtb cmd_rds.X <ESCAPE>g
####
R>
```

Other Supported Absolute Files

The example which follows shows how to download Intel hexadecimal files by the same method (but different **load** options) can be used by load Tektronix hexadecimal and Motorola S-record files as well.

```
R>load -io <RETURN> <RETURN>
$ cat ihexfile <ESCAPE>g
#####
Data records = 00003 Checksum error = 00000
R>
```

Displaying Memory In Mnemonic Format

Once you have loaded a program into the emulator, you can verify that the program has indeed been loaded by displaying memory in mnemonic format.

```
R>m -dm 1000..1048
00001000  -    MOV #00,R15
00001002  -    MOV #00,R0
00001004  -    MOV.L @(000104c[,PC]),R1
00001006  -    MOV.B R0,@R1
00001008  -    MOV.B @R1,R0
0000100a  -    CMP/EQ #00,R0
0000100c  -    BT 0001008
0000100e  -    CMP/EQ #41,R0
00001010  -    BT 0001018
00001012  -    CMP/EQ #42,R0
00001014  -    BT 000101e
00001016  -    BF 0001024
00001018  -    MOV.L @(0001054[,PC]),R4
0000101a  -    BRA 0001028
0000101c  -    MOV #11,R3
0000101e  -    MOV.L @(0001058[,PC]),R4
00001020  -    BRA 0001028
00001022  -    MOV #11,R3
00001024  -    MOV #0f,R3
00001026  -    MOV.L @(000105c[,PC]),R4
00001028  -    MOV.L @(0001050[,PC]),R5
0000102a  -    MOV R5,R6
0000102c  -    ADD #20,R6
0000102e  -    MOV #00,R0
00001030  -    MOV.B R0,@R5
00001032  -    ADD #01,R5
00001034  -    CMP/EQ R5,R6
00001036  -    BF 0001030
00001038  -    MOV.L @(0001050[,PC]),R5
0000103a  -    MOV R5,R6
0000103c  -    ADD R3,R6
0000103e  -    MOV.B @R4+,R0
00001040  -    MOV.B R0,@R5
00001042  -    ADD #01,R5
00001044  -    CMP/EQ R5,R6
00001046  -    BF 000103e
00001048  -    BT 0001002
```

If you display memory in mnemonic format and do not recognize the instructions listed or see some illegal instructions or opcodes, go back and make sure the memory locations you have typed are mapped properly. If the memory map is not the problem, recheck the linker load map listing to verify that the absolute addresses of the program match with the locations you are trying to display.

Stepping Through the Program

The emulator allows you to execute one instruction or a number of instructions with the **s** (step) command. Enter the **? s** to view the options available with the step command.

R>**? s**

```
s - step emulation processor

   s                     - step one from current PC
   s <count>             - step <count> from current PC
   s <count> $           - step <count> from current PC
   s <count> <addr>      - step <count> from <addr>
   s -q <count> <addr>   - step <count> from <addr>, quiet mode
   s -w <count> <addr>   - step <count> from <addr>, whisper mode

 --- NOTES ---
   STEPCOUNT MUST BE SPECIFIED IF ADDRESS IS SPECIFIED!
   If <addr> is not specified, default is to step from current PC.
   A <count> of 0 implies step forever.
```

A step count of 0 will cause the stepping to continue "forever" (until some break condition, such as "write to ROM", is encountered, or until you enter <CTRL>**c**). The following command will step from the first address of the sample program.

R>**s 1 1000**

```
 00001000  -                   MOV #00,R15
 PC = 00001002
```

Note

Step(**s**) and run(**r**) commands from odd address are not allowed. Always you must perform step and run commands from even address.

Note

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

Displaying Registers

The step command shown above executed the "MOV #00,R15" instruction. Enter the following command to view the contents of the registers.

```
M>reg
```

```
reg pc=00001002 sr=000000f0 r0=00000000 r1=00000000 r2=00000000 r3=00000000
reg r4=00000000 r5=00000000 r6=00000000 r7=00000000 r8=00000000 r9=00000000
reg r10=00000000 r11=00000000 r12=00000000 r13=00000000 r14=00000000
reg r15=00000000 sp=00000000 gbr=00000000 vbr=00000000 pr=00000000
reg mach=00000000 macl=00000000
```

The register contents are displayed in a "register modify" command format. This allows you to save the output of the **reg** command to a command file which may later be used to restore the register contents. (Refer to the **po** (port options) command description in the *Terminal Interface: User's Reference* for more information on command files.)

Refer to the "REGISTER CLASS and NAME" section in the "SH-7000 Emulator Specific Command Syntax" appendix for more information on the register names and classes.

Combining Commands

More than one command may be entered in a single command line. The commands must be separated by semicolons (;). For example, you could execute the next instruction(s) and display the registers by entering the following.

```
M>s;reg
```

```
00001002  -              MOV #00,R0
PC = 00001004
reg pc=00001004 sr=000000f0 r0=00000000 r1=00000000 r2=00000000 r3=00000000
reg r4=00000000 r5=00000000 r6=00000000 r7=00000000 r8=00000000 r9=00000000
reg r10=00000000 r11=00000000 r12=00000000 r13=00000000 r14=00000000
reg r15=00000000 sp=00000000 gbr=00000000 vbr=00000000 pr=00000000
reg mach=00000000 macl=00000000
```

The sample above shows you that "MOV #00,R0" is executed by step command.

Using Macros

Suppose you want to continue stepping through the program and displaying registers after each step. You could continue entering **s** command followed by **reg** command, but you may find this tiresome. It is easier to use a macro to perform a sequence of commands which will be entered again and again.

Macros allow you to combine and store commands. For example, to define a macro which will display registers after every step, enter the following command.

```
M>mac st={s;reg}
```

Once the **st** macro has been defined, you can use it as you would use any other command.

```
M>st
```

```
# s ; reg
00001004  -                   MOV.L @(000104c[,PC]),R1
PC = 00001006
reg pc=00001006 sr=000000f0 r0=00000000 r1=0f000000 r2=00000000 r3=00000000
reg r4=00000000 r5=00000000 r6=00000000 r7=00000000 r8=00000000 r9=00000000
reg r10=00000000 r11=00000000 r12=00000000 r13=00000000 r14=00000000
reg r15=00000000 sp=00000000 gbr=00000000 vbr=00000000 pr=00000000
reg mach=00000000 macl=00000000
```

Command Recall

The command recall feature is yet another, easier way to enter commands again and again. You can press <CTRL>**r** to recall the commands which have just been entered. If you go past the command of interest, you can press <CTRL>**b** to move forward through the list of saved commands. To continue stepping through the sample program, you could repeatedly press <CTRL>**r** to recall and <RETURN> to execute the **st** macro.

Repeating Commands

The **rep** command is also helpful when entering commands repetitively. You can repeat the execution of macros as well as normal commands. For example, you could enter the following command to cause the **st** macro to be executed four times.

```
M>rep 4 st
```

```
# s ; reg
00001006  -                MOV.B R0,@R1
PC = 00001008
reg pc=00001008 sr=000000f0 r0=00000000 r1=0f000000 r2=00000000 r3=00000000
reg r4=00000000 r5=00000000 r6=00000000 r7=00000000 r8=00000000 r9=00000000
reg r10=00000000 r11=00000000 r12=00000000 r13=00000000 r14=00000000
reg r15=00000000 sp=00000000 gbr=00000000 vbr=00000000 pr=00000000
reg mach=00000000 macl=00000000
# s ; reg
00001008  -                MOV.B @R1,R0
PC = 0000100a
reg pc=0000100a sr=000000f0 r0=00000000 r1=0f000000 r2=00000000 r3=00000000
reg r4=00000000 r5=00000000 r6=00000000 r7=00000000 r8=00000000 r9=00000000
reg r10=00000000 r11=00000000 r12=00000000 r13=00000000 r14=00000000
reg r15=00000000 sp=00000000 gbr=00000000 vbr=00000000 pr=00000000
reg mach=00000000 macl=00000000
# s ; reg
0000100a  -                CMP/EQ #00,R0
PC = 0000100c
reg pc=0000100c sr=000000f1 r0=00000000 r1=0f000000 r2=00000000 r3=00000000
reg r4=00000000 r5=00000000 r6=00000000 r7=00000000 r8=00000000 r9=00000000
reg r10=00000000 r11=00000000 r12=00000000 r13=00000000 r14=00000000
reg r15=00000000 sp=00000000 gbr=00000000 vbr=00000000 pr=00000000
reg mach=00000000 macl=00000000
# s ; reg
0000100c  -                BT 0001008
PC = 00001008
reg pc=00001008 sr=000000f1 r0=00000000 r1=0f000000 r2=00000000 r3=00000000
reg r4=00000000 r5=00000000 r6=00000000 r7=00000000 r8=00000000 r9=00000000
reg r10=00000000 r11=00000000 r12=00000000 r13=00000000 r14=00000000
reg r15=00000000 sp=00000000 gbr=00000000 vbr=00000000 pr=00000000
reg mach=00000000 macl=00000000
```

Command Line Editing

The terminal interface supports the use of HP-UX **ksh(1)-**like editing of the command line. The default is for the command line editing feature to be disabled to be compatible with earlier versions of the interface. Use the **cl** command to enable command line editing.

```
M>cl -e
```

Refer to "Command Line Editing" in the *HP64700-Series Emulators Terminal Interface Reference* for information on using the command line editing feature.

Modifying Memory

The preceding step and register commands show the sample program is executing Scan loop, where it continually reads the command input byte to check if a command had been entered. Use the **m** (memory) command to modify the command input byte.

```
M>m 0f000000=41
```

To verify that 41H has been written to 0f000000H, enter the following command.

```
M>m -db 0f000000
  0f000000..0f000000  41
```

When memory was displayed in byte format earlier, the display mode was changed to "byte". The display and access modes from previous commands are saved and they become the defaults.

Specifying the Access and Display Modes

There are a couple different ways to modify the display and access modes. One is to explicitly specify the mode with the command you are entering, as with the command **m -db 0f000000**. The **mo** (display and access mode) command is another way to change the default mode. For example, to display the current modes, define the display mode as "word", and redisplay 0f000000H, enter the following commands.

```
M>mo
mo -ab -db

M>mo -dw
M>m 0f000000
  0f000000..0f000000  0041
```

To continue the rest of program.

```
M>r
U>
```

Display the **Msg_Dest** memory locations (destination of the message, 0f000002H) to verify that the program moved the correct ASCII bytes. At this time you want to see correct byte values, so "-db" option (display with byte) is used.

```
U>m -db 0f000002..0f000021
  0f000002..0f000011  54 48 49 53 20 49 53 20 4d 45 53 53 41 47 45 20
  0f000012..0f000021  41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Running the Sample Program

The emulator allows you to execute a program in memory with the **r** command. The **r** command by itself causes the emulator to begin executing at the current program counter address. The following command will begin running the sample program from 1000H.

```
M> r 1000
```

The **r rst** command specifies that the emulator begin to executing from target system reset (see the "Execution Topics" section in the "In-Circuit Emulation" chapter).

Note

Step(**s**) and run(**r**) commands from odd address are not allowed. Always you must perform step and run commands from even address.

Searching Memory for Data

The **ser** (search memory for data) command is another way to verify that the program did what it was supposed to do.

```
U>ser 0f000002..0f000021="THIS IS MESSAGE A"
pattern match at address: 0f000002
```

If any part of the data specified in the **ser** command is not found, no match is displayed (No message displayed).

Breaking into the Monitor

You can use the break command (**b**) command to generate a break to the monitor. While the break will occur as soon as possible, the actual stopping point may be many cycles after the break request (depending on the type of instruction being executed and whether the processor is in a special state).

```
U>b
M>
```

Note

If DMA transfer by internal DMAC is in progress with BURST transfer mode, **b** 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(with the **bp** command), 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.

Displaying and Modifying the Break Conditions

Before you can define software breakpoints, you must enable software breakpoints with the **bc** (break conditions) command. To view the default break conditions and change the software breakpoint condition, enter the **bc** command with no option. This command displays current configuration of break conditions.

```
M>bc
```

```
bc -d bp #disable
bc -e rom #enable
bc -d bnct #disable
bc -d cmbt #disable
bc -d trig1 #disable
bc -d trig2 #disable
```

To enable the software break point feature enter

```
M>bc -e bp
```

Defining a Software Breakpoint

Now that the software breakpoint feature is enabled, you can define software breakpoints. Enter the following command to break on the address of the **Cmd_I** (address 1024H) label.

```
M>bp 1024
M>bp
### BREAKPOINT FEATURE IS ENABLED ###
bp 00001024 #enabled
```

Run the program, and verify that execution broke at the appropriate address.

```
M>r 1000
U>m 0f000000=43
!ASYNC_STAT  615! Software breakpoint: 00001024
M>st
  # s ; reg
00001024  -                MOV #0f,R3
PC = 00001026
reg pc=00001026 sr=000000f0 r0=00000043 r1=0f000000 r2=00000000 r3=0000000f
reg r4=00001071 r5=0f000013 r6=0f000013 r7=00000000 r8=00000000 r9=00000000
reg r10=00000000 r11=00000000 r12=00000000 r13=00000000 r14=00000000
reg r15=00000000 sp=00000000 gbr=00000000 vbr=00000000 pr=00000000
reg mach=00000000 macl=00000000
```

When a breakpoint is hit, it becomes disabled. You can use the **-e** option with the **bp** command to re-enable the software breakpoint.

```
M>bp
### BREAKPOINT FEATURE IS ENABLED ###
bp 00001024 #disabled
M>bp -e 1024
M>bp
### BREAKPOINT FEATURE IS ENABLED ###
bp 00001024 #enabled
M>r
U>m 0f000000=43
!ASYNC_STAT  615! Software breakpoint: 00001024
M>bp
### BREAKPOINT FEATURE IS ENABLED ###
bp 00001024 #disabled
```

Using the Analyzer

Predefined Trace Labels

Three trace labels are predefined in the SH-7000 emulator. You can view these labels by entering the **tlb** (trace label) command with no options.

```
M>tlb
```

```
#### Emulation trace labels
tlb addr 0..27
tlb data 32..63
tlb stat 64..79
```

Predefined Status Equates

Common values for the SH-7000 status trace signals have been predefined. You can view these predefined equates by entering the **equ** command with no options.

```
M>equ
```

```
### Equates ###
equ bg=0xxxxxxxxxxxxxxx0y
equ byte=0xxxxxxxxxx00x0xxy
equ cpu=0xxxxxxxxxxxx1xxxy
equ data=0xxxxxxxxxxxxx0xxy
equ dma=0xxxxxxxxxxxx00xxy
equ fetch=0xxxxxxxxxxxx111xy
equ fg=0xxxxxxxxxxxxxxx1y
equ grd=0xxxxxxxxxxxxxxxy
equ intack=0xx0xxxxxxxxx111xy
equ long=0xxxxxxxxxx101xxxy
equ read=0xxxxxxxxxxxxxx1xy
equ refresh=0xxxxxxxxxxxx01xxy
equ word=0xxxxxxxxxx01xxxxy
equ write=0xxxxxxxxxxxxx00xy
equ wrrom=0x0xxxxxxxxxxx00xy
```

These equates may be used to specify values for the **stat** trace label when qualifying trace conditions.

Specifying a Simple Trigger

The **tg** analyzer command is a simple way to specify a condition on which to trigger the analyzer. Suppose you wish to trace the states of the program after the read of "B"(42H) command from the command input byte. Enter the following commands to set up the trace, run the program, issue the trace, and display the trace status.(Refer to the "Specifying Data for Trigger or Store Condition" section of "Using the Emulator" chapter to trigger for data)

```
M>tg addr=0f000000 and data=42xxxxxx and stat=read
M>t
emulation trace started
M>r 1000
U>ts
  --- Emulation Trace Status ---
  NEW User trace running
  Arm ignored
  Trigger not in memory
  Arm to trigger ?
  States ? (8192) ?..?
  Sequence term 1
  Occurrence left 1
```

The trace status shows that the trigger condition has not been found. You would not expect the trigger to be found because no commands have been entered. Modify the command input byte to "B"(42H) and display the trace status again.

```
U>m 0f000000=42
U>ts
  --- Emulation Trace Status ---
  NEW User trace complete
  Arm ignored
  Trigger in memory
  Arm to trigger ?
  States 8192 (8192) 0..8191
  Sequence term 2
  Occurrence left 1
```

The trace status shows that the trigger has been found. Enter the following command to display the first 15 states of the trace.

```
U>tl -t 15
     Line  addr,H  SH7032 mnemonic,H                       count,R
  -------  ------- ----------------------------------- -------------
        0  f000000   42xxxxxx  read  byte              -------------
        1  000100e   xxxxxx88  fetch                          0.26uS
        2  000100f   xxxxxx41  fetch                          0.26uS
        3  0001010   xxxxxx89  fetch                          0.24uS
          =000100e CMP/EQ #41,R0
        4  0001011   xxxxxx02  fetch                          0.26uS
        5  0001012   xxxxxx88  fetch                          0.24uS
          =0001010 BT 0001018
        6  0001013   xxxxxx42  fetch                          0.26uS
        7  0001014   xxxxxx89  fetch                          0.24uS
          =0001012 CMP/EQ #42,R0
        8  0001015   xxxxxx03  fetch                          0.26uS
        9  0001016   xxxxxx8b  fetch                          0.24uS
          =0001014 BT 000101e
       10  0001017   xxxxxx05  fetch                          0.26uS
       11  0001018   xxxxxxd4  fetch                          0.24uS
       12  0001019   xxxxxx0e  fetch                          0.26uS
```

```
   13  000101e   xxxxxxd4  fetch                              0.24uS
   14  000101f   xxxxxx0e  fetch                              0.24uS
```

Line 0 in the trace list above shows the state which triggered the analyzer. The trigger state is always on line 0.

To list the next lines of the trace, enter the following command.

```
U>tl
```

```
  Line  addr,H  SH7032 mnemonic,H                          count,R
------- ------- ---------------------------------- -------------
   15  0001020   xxxxxxa0  fetch                              0.26uS
      =000101e MOV.L @(0001058[,PC]),R4
   16  0001021   xxxxxx02  fetch                              0.24uS
   17  0001022   xxxxxxe3  fetch                              0.26uS
      =0001020 BRA 0001028
   18  0001023   xxxxxx11  fetch                              0.24uS
   19  0001058   xxxxxx00  read  long                         0.26uS
   20  0001059   xxxxxx00  read  long                         0.24uS
   21  000105a   xxxxxx10  read  long                         0.26uS
   22  000105b   xxxxxx71  read  long                         0.24uS
   23  0001028   xxxxxxd5  fetch                              0.26uS
      =0001022 MOV #11,R3
   24  0001029   xxxxxx09  fetch                              0.24uS
   25  000102a   xxxxxx66  fetch                              0.26uS
      =0001028 MOV.L @(0001050[,PC]),R5
   26  000102b   xxxxxx53  fetch                              0.24uS
   27  000102c   xxxxxx76  fetch                              0.26uS
      =000102a MOV R5,R6
   28  000102d   xxxxxx20  fetch                              0.24uS
   29  0001050   xxxxxx0f  read  long                         0.26uS
```

Trigger Position

You can specify where the trigger state will be positioned with in the emulation trace list. The following three basical trigger positions are defined.

- **s** start
- **c** center
- **e** end

When **s**(start) trigger position is selected, the trigger is positioned at the start of the trace list. You can trace the states after the trigger state.

When **c**(center) trigger position is selected, the trigger is positioned at the center of the trace list. You can trace the states around the trigger.

When **e**(end) trigger position is selected, the trigger is positioned at the end of the trace list. You can trace the state before the trigger.

In the above section, you have traced the states of the program after a certain state, because the default trigger position was **s**(start). If you want to trace the states of the program around a certain state, you need to change the trigger position.

For example, if you wish to trace the transition to the command A process, change the trigger position to "center" and specify the trigger condition.

To specify the trigger position, enter the following command.

```
U>tp c
```

Specify the trigger condition by typing

```
U>tg addr=1018
```

Enter the trace command to start the trace.

```
U>t
```

```
Emulation trace started
```

```
U>ts
```

```
--- Emulation Trace Status ---
NEW User trace complete
Arm ignored
Trigger in memory
Arm to trigger ?
States 8192 (8192) -4096..4095
Sequence term 2
Occurrence left 1
```

The trace status shows that the trigger has been found. Enter the following command to display the states about the execution state of address 1018H.

```
U>tl -10..9
```

```
   Line  addr,H  SH7032 mnemonic,H                                 count,R
-------  ------- ---------------------------------- -------------
    -10  000100d   xxxxxxfc  fetch                                 0.26uS
     -9  f000000   41xxxxxx  read  byte                            0.04uS
     -8  000100e   xxxxxx88  fetch                                 0.26uS
        =000100c BT 0001008
     -7  000100f   xxxxxx41  fetch                                 0.24uS
     -6  0001010   xxxxxx89  fetch                                 0.26uS
        =000100e CMP/EQ #41,R0
     -5  0001011   xxxxxx02  fetch                                 0.24uS
     -4  0001012   xxxxxx88  fetch                                 0.26uS
        =0001010 BT 0001018
     -3  0001013   xxxxxx42  fetch                                 0.24uS
     -2  0001014   xxxxxx89  fetch                                 0.26uS
     -1  0001015   xxxxxx03  fetch                                 0.24uS
      0  0001018   xxxxxxd4  fetch                                 0.26uS
      1  0001019   xxxxxx0e  fetch                                 0.24uS
      2  000101a   xxxxxxa0  fetch                                 0.24uS
        =0001018 MOV.L @(0001054[,PC]),R4
```

```
     3  000101b   xxxxxx05  fetch                                0.26uS
     4  000101c   xxxxxxe3  fetch                                0.24uS
       =000101a BRA 0001028
     5  000101d   xxxxxx11  fetch                                0.26uS
     6  0001054   xxxxxx00  read  long                           0.24uS
     7  0001055   xxxxxx00  read  long                           0.26uS
     8  0001056   xxxxxx10  read  long                           0.24uS
     9  0001057   xxxxxx60  read  long                           0.26uS
```

The transition states to the process for the command A are displayed.

To reduce fetch cycle from trace list, enter the following command.

U>**tl -os**

```
   Line  addr,H  SH7032 mnemonic,H                           count,R
-------  ------- ----------------------------------- -------------
     10 =000101c MOV #11,R3                                  0.24uS
     12 =0001028 MOV.L @(0001050[,PC]),R5                    0.52uS
     14 =000102a MOV R5,R6                                   0.50uS
     16  0001050   xxxxxx0f  read  long                      0.50uS
     17  0001051   xxxxxx00  read  long                      0.24uS
     18  0001052   xxxxxx00  read  long                      0.26uS
     19  0001053   xxxxxx02  read  long                      0.24uS
     20 =000102c ADD #20,R6                                  0.26uS
     22 =000102e MOV #00,R0                                  0.50uS
     24 =0001030 MOV.B R0,@R5                                0.50uS
     26 =0001032 ADD #01,R5                                  0.50uS
     28  f000002   xxxx00xx  write byte                      0.28uS
     29 =0001034 CMP/EQ R5,R6                                0.26uS
```

You can still display all states in trace list. Enter the following command.

U>**tl -od**

```
   Line  addr,H  SH7032 mnemonic,H                           count,R
-------  ------- ----------------------------------- -------------
     30  0001037   xxxxxxfb  fetch                           0.24uS
     31  0001038   xxxxxxd5  fetch                           0.26uS
        =0001036 BF 0001030
     32  0001039   xxxxxx05  fetch                           0.24uS
     33  000103a   xxxxxx66  fetch                           0.26uS
     34  000103b   xxxxxx53  fetch                           0.24uS
     35  0001030   xxxxxx25  fetch                           0.26uS
     36  0001031   xxxxxx00  fetch                           0.24uS
     37  0001032   xxxxxx75  fetch                           0.26uS
        =0001030 MOV.B R0,@R5
     38  0001033   xxxxxx01  fetch                           0.24uS
     39  0001034   xxxxxx36  fetch                           0.26uS
        =0001032 ADD #01,R5
     40  0001035   xxxxxx50  fetch                           0.26uS
     41  f000003   xxxxxx00  write byte                      0.04uS
     42  0001036   xxxxxx8b  fetch                           0.26uS
        =0001034 CMP/EQ R5,R6
     43  0001037   xxxxxxfb  fetch                           0.24uS
     44  0001038   xxxxxxd5  fetch                           0.26uS
        =0001036 BF 0001030
     45  0001039   xxxxxx05  fetch                           0.24uS
```

```
46  000103a   xxxxxx66  fetch                               0.24uS
47  000103b   xxxxxx53  fetch                               0.26uS
48  0001030   xxxxxx25  fetch                               0.24uS
49  0001031   xxxxxx00  fetch                               0.26uS
```

For a Complete Description

For a complete description of the HP 64700 Series analyzer, refer to the *HP 64700 Emulators Terminal Interface: Analyzer User's Guide.*

Resetting the Emulator

To reset the emulator, enter the following command.

```
U>rst
R>
```

The emulator is held in a reset state (suspended) until a **b** (break), **r** (run), or **s** (step) command is entered. A CMB execute signal will also cause the emulator to run if reset.

The **-m** option to the **rst** command specifies that the emulator begin executing in the monitor after reset instead of remaining in the suspended state.

```
R>rst -m
M>
```

3

Using the Emulator

Introduction

Many of the topics described in this chapter involve the commands which are unique to the SH-7000 emulator such as the **cf** command which allows you to specify emulator configuration.
A reference-type description of the SH-7000 emulator configuration items can be found in the "CONFIG_ITEMS" section in the "SH-7000 Emulator Specific Command Syntax" appendix.

This chapter will:

- Execution Topics
  - Restricting the Emulator to Real-Time Runs
  - Execution command from add address
  - Setting Up to Break on an Analyzer Trigger
  - Making Coordinated Measurements
- Memory Mapping
- Analyzer Topics
  - Analyzer Status Qualifiers
  - Specifying Data for Trigger or Store Condition
  - Analyzer Clock Speed
- Monitor Topics

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.

Execution Topics

The description in this section are of emulation tasks which involve program execution in general.

Restricting the Emulator to Real-Time Runs

By default, the emulator is not restricted to real-time runs. However, you may wish to restrict runs to real-time to prevent accidental breaks that might cause target system problems. Use the **cf** (configuration) command to enable the **rrt** configuration item.

```
R>cf rrt=en
```

When runs are restricted to real-time and the emulator is running user code, the system refuses all commands that cause a break except **rst** (reset), **r** (run), **s**(step), and **b** (break to monitor).

The following commands are not allowed when runs are restricted to real-time:

- **reg** (register display/modification).
- **m** (memory display/modification).

The following command will disable the restriction to real-time runs and allow the system to accept commands normally.

```
R>cf rrt=dis
```

Setting Up to Break on an Analyzer Trigger

The analyzer may generate a break request to the emulation processor. To set up to break on an analyzer trigger, follow the steps below.

Specify the Signal Driven when Trigger is Found

Use the **tgout** (trigger output) command to specify which signal is driven when the analyzer triggers. Either the "trig1" or the "trig2" signal can be driven on the trigger.

```
R>tgout trig1
```

Enable the Break Condition

Enable the "trig1" break condition.

```
R>bc -e trig1
```

After you specify the trigger to drive "trig1" and enable the "trig1" break condition, set up the trace, enter the **t** (trace) command, and run the program.

Making Coordinated Measurements

Coordinated measurements are measurements made between multiple HP 64700 Series emulators which communicate via the Coordinated Measurement Bus (CMB). Coordinated measurements can also include other instruments which communicate via the BNC connector. A trigger signal from the CMB or BNC can break emulator execution into the monitor, or it can arm the analyzer. An analyzer can send a signal out on the CMB or BNC when it is triggered. The emulator can send an EXECUTE signal out on the CMB when you enter the **x** (execute) command.

Coordinated measurements can be used to start or stop multiple emulators, start multiple trace measurements, or to arm multiple analyzers.

As with the analyzer generated break, breaks to the monitor on CMB or BNC trigger signals are interpreted as a "request to break". The emulator looks at the state of the CMB READY (active high) line to determine if it should break. It does not interact with the EXECUTE (active low) or TRIGGER (active low) signals.

Note

When **qbrk** (quick temporary break) is enabled in emulator configuration, you can not use CMB function.

For information on how to make coordinated measurements, refer to the *HP 64700 Emulators Terminal Interface: Coordinated Measurement Bus User's Guide* manual.

Memory Mapping

You can define up to 16 memory ranges(at 16K byte boundaries and at least 16K byte in length). You don't have to map the internal RAM/ROM area and all registers of on-chip peripheral modules, since the SH-7000 emulator has memory and map them automatically. You can characterize memory ranges as emulation RAM, emulation ROM, target RAM, target ROM, or guarded memory.

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.

Mapping as Emulation Memory

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 as emulation memory, 32K byte is used as following.

R>**map**

```
# remaining number of terms   : 16
# remaining emulation memory  : 100000h bytes
map  other tram
```

```
R>map 0..3fff eram
R>map
# remaining number of terms   : 15
# remaining emulation memory  : f8000h bytes
map  00000000..00003fff  eram    # term 1
map  other tram
```

Also, when you use 4M byte memory module and characterize memory range which does not override 128K as emulation memory, 128K byte is used by the emulation mapper.

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 spece. Address/data multiplexed I/O spece is always accessed as target RAM. However, when you map this area as guarded memory, you can not access this area by commands.

Analyzer Topics

Analyzer Status Qualifiers

The following are the analyzer status labels which may be used in the "**tg**" and "**tsto**" analyzer commands.

Qualifier	Status bits	Description
bg	0xxxxxxxxxxxxxxx0y	background cycle
byte	0xxxxxxxxxx00x0xxy	byte memory cycle
cpu	0xxxxxxxxxxxx1xxxy	CPU cycle
data	0xxxxxxxxxxxxx0xxy	data bus cycle
dma	0xxxxxxxxxxxx00xxy	DMA cycle
fetch	0xxxxxxxxxxxx111xy	program fetch
fg	0xxxxxxxxxxxxxxx1y	foreground cycle
grd	00xxxxxxxxxxxxxxxy	guarded memory access
intack	0xx0xxxxxxxxx111xy	interrupt acknowledge
long	0xxxxxxxxxx101xxxy	long word memory cycle
read	0xxxxxxxxxxxxxx1xy	read cycle
refresh	0xxxxxxxxxxxx01xxy	refresh cycle
word	0xxxxxxxxxx01xxxxy	word memory cycle
write	0xxxxxxxxxxxxx00xy	write cycle
wrrom	0x0xxxxxxxxxxx00xy	write to ROM cycle

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.

```
U>tg 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 3-1. This is because the analyzer is designed so that it can capture data on internal data bus (which has 32 bits width).

Table 3-1 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.

Analyzer Clock Speed

The emulation analyzer can capture both the execution states and bus states. The analyzer has a counter which allows to count either time or occurrence of bus states. If you use 64794A/C/D deep emulation analyzer, the trace state and time counter qualifiers can be used regardless of clock speed. If you use 64704A emulation analyzer, the trace state and time counter qualifiers are limited by clock speed as the following.

Table 3-2 Analyzer Counter

Clock Speed	Analyzer Speed Setting	Valid count qualifier options
clock =< 16.6MHz	S(slow)	counting <state> counting time
16.6MHz < clock =< 20MHz	F(fast)	counting <state>

If your target system clock is between 16.6MHz and 20MHz, you can use the analyzer state counter. In this case, the analyzer state counter counts occurrences of the states which you specify. Assume that you would like to count occurrences of the state which the processor read a data.

```
M>tcq stat=read
M>tck -s F
```

If your target system clock is equal to 16.6MHz or less than 16.6MHz, you can use analyzer time and state counter. Assume that you would like to count time.

```
M>tck -s S
M>tcq time
```

Monitor Topics

The monitor is a program which is executed by the emulation processor. It allows the emulation system controller to access target system resources. For example, when you enter a command that requires access to target system resources (display target memory, for example), the system controller writes a command code to a communications area and breaks the execution of the emulation processor into the monitor. The monitor program then reads the command from the communications area and executes the processor instructions which access the target system. After the monitor has performed its task, execution returns to the target program.

The SH-7000 emulator has memory for the monitor program. So the monitor program does not occupy processor address space and emulation memory.

4

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.

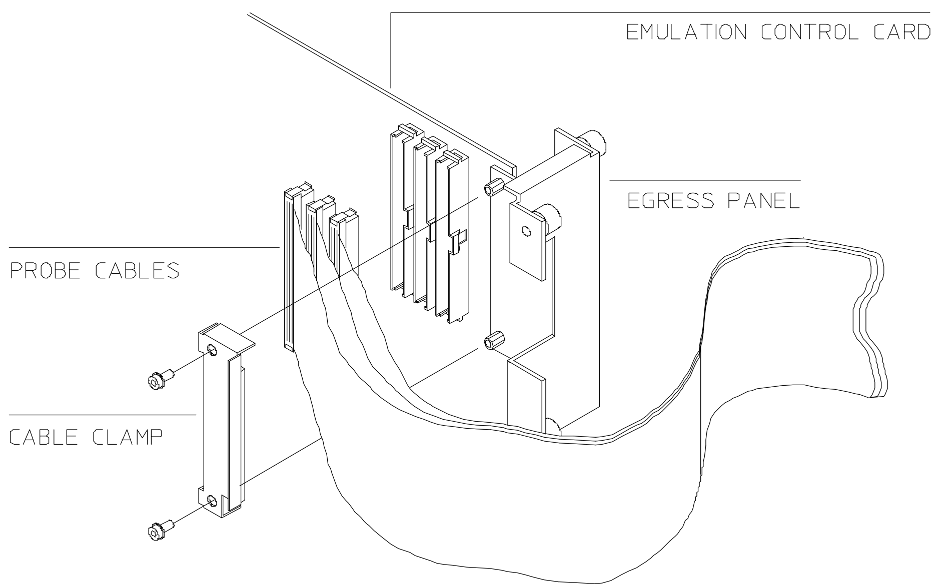


Figure 4-1 Installing cables to the control board

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

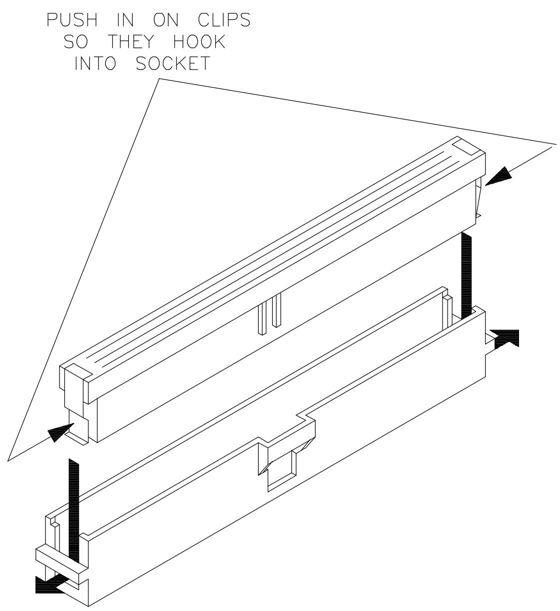


Figure 4-2 Installing cables into cable sockets

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

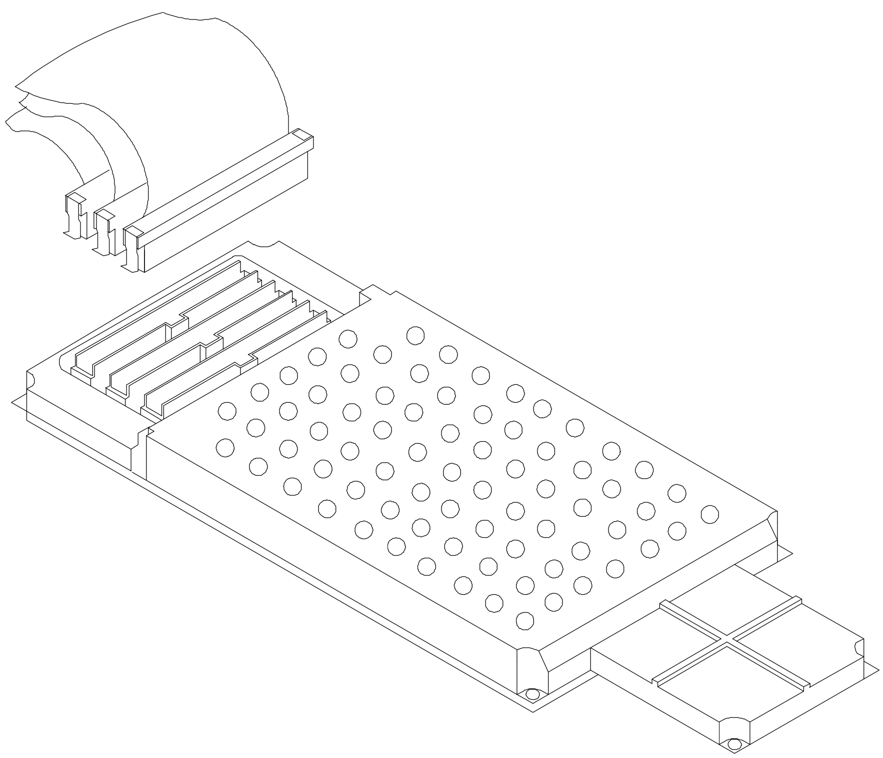


Figure 4-3 Installing cables to the emulation probe

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.

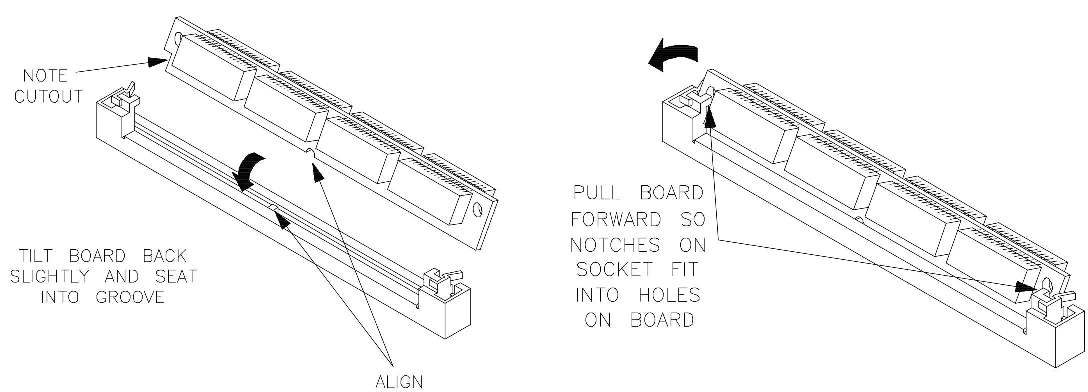


Figure 4-4 Installing the memory module

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 swicthes 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.

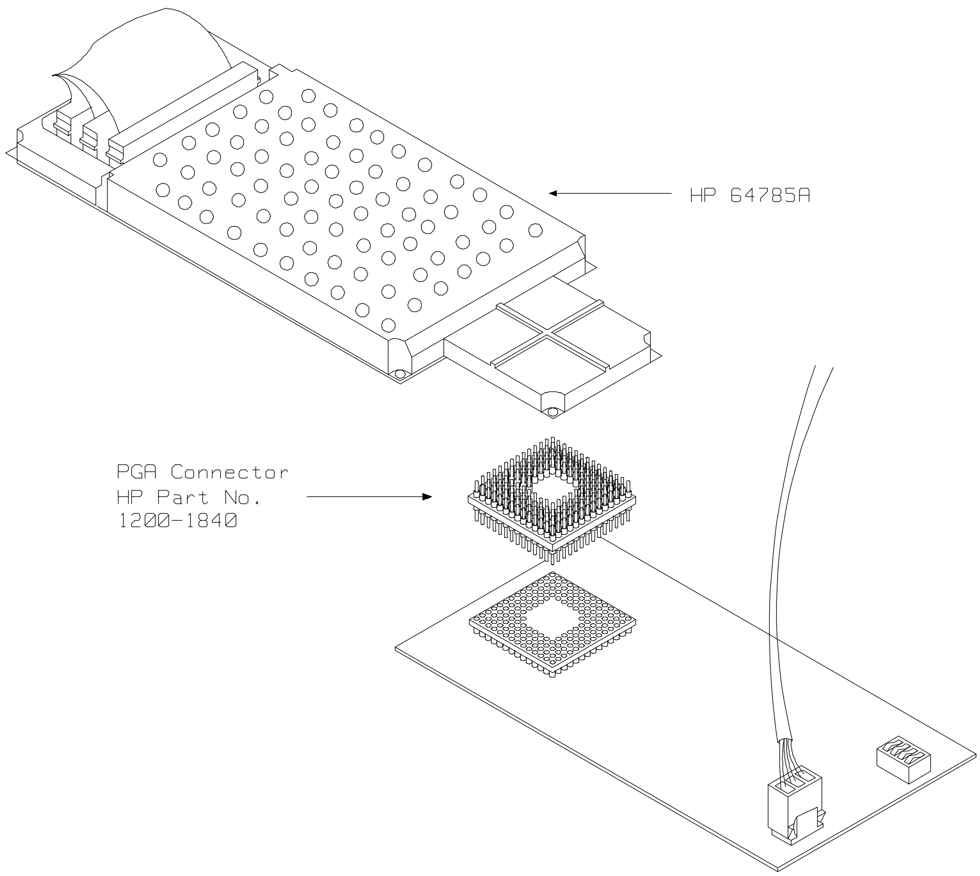


Figure 4-5 Installing the demo target board

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 electorostatic 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.

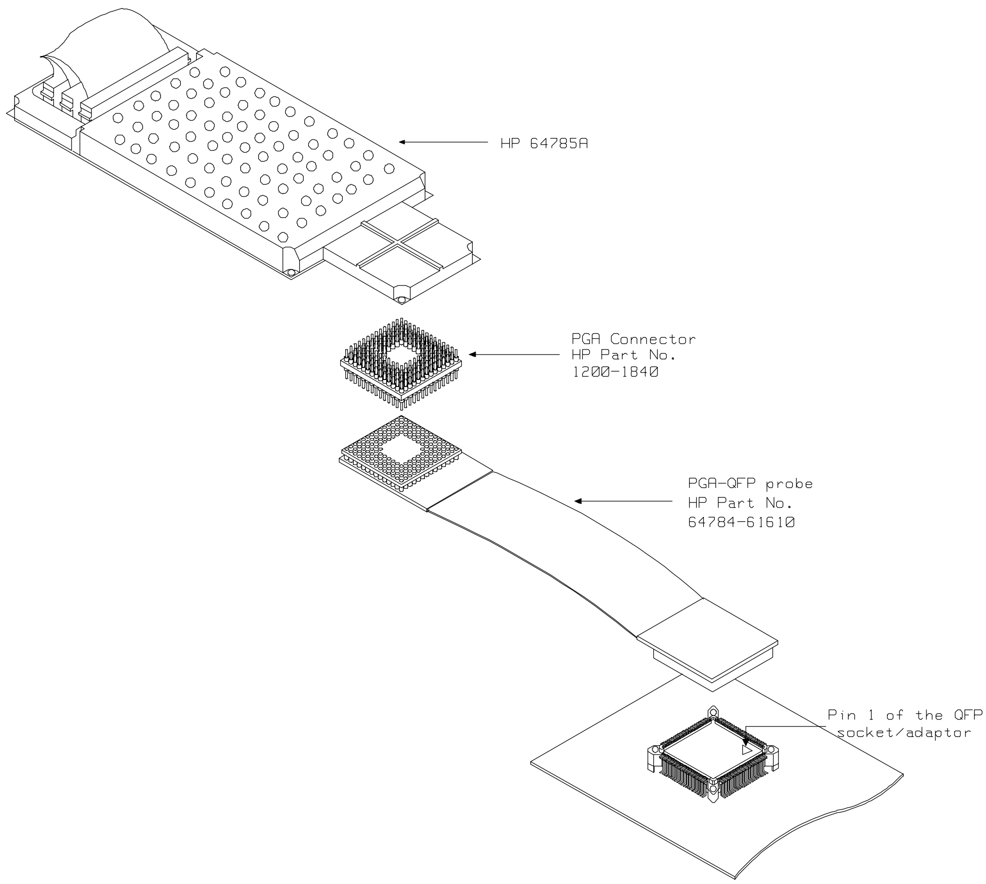


Figure 4-6 Installing into a target system board

In-Circuit configuration

The SH-7000 emulator provides configuration options for the following in-circuit emulation issues. Refer to the "CONFIG_ITEM" section in the "SH-7000 Emulator Specific Command Syntax" appendix.

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/$\overline{\text{BREQ}}$ pin. By default, this configuration is set to "PA8".

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

Execution Topics

The descriptions in this section are of emulation tasks which involve program execution in general.

Run from Target System Reset

You can use **"r rst"** command to execute program from target system reset. You will see "**T>**" system prompt when you enter "r rst". In this status, the emulator accept target system reset. Then program starts if reset signal from target system is released.

Note

In the "Awaiting target reset" status(T>), 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 'cf chip' in situations without clock source.

Memory Cycles in Background

While the SH-7000 emulator is running in the monitor program, the probe pins of the emulator are in the following state.

Address Bus	Same as runnig user's program
Data Bus	High impedance except accessing to target/emulation memory by monitor program
All Memory strobe signals	Always high except accessing to target/emulation memory by monitor program

While in the monitor program, fetch and data access cycles for address from 0 to 1000 hex occur. The SH-7000 emulator does not output these cycles to the target system, but they are effective for user break controller. Also, when you direct displaying/modifying memory or registers of on-chip peripheral modules, data access cycles for address which you specify is effective for user break controller.

Electrical Characteristics

The AC characteristics of the HP 64785A SH-7000 emulator are listed in the following table

Table 4-2 Clock Timing

Characteristic	Symbol	SH-7034 20MHz		HP 64785A Worst Case		HP 64785A Typical (*1)	Unit
		Min	Max	Min	Max		
EXTAL input high level pulse	t_{EXH}	10		-		10	ns
EXTAL input low level pulse	t_{EXL}	10		-		10	ns
EXTAL input rise time	t_{EXr}		5		-	5	ns
EXTAL input fall time	t_{EXf}		5		-	5	ns
Clock cycle time	t_{cyc}	50	500	-	-	50,500	ns
Clock high pulse width	t_{CH}	20		-		24	ns
Clock low pulse width	t_{CL}	20		-		18	ns
Clock rise time	t_{Cr}		5		-	4	ns
Clock fall time	t_{Cf}		5		-	4	ns
Reset oscillation setting time	t_{OSC1}	10		10		10	ns
Software stanby oscillation setting time	t_{OSC2}	10		10		10	ns

*1 Typical outputs measured with 50pF load

Table 4-3 Control Signal Timing

Characteristic	Symbol	SH-7034 20MHz		HP 64785A Worst Case		HP 64785A Typical (*1)	Unit
		Min	Max	Min	Max		
$\overline{\text{RESET}}$ setup time	t_{RESS}	200		250		-	ns
$\overline{\text{RESET}}$ pulse width	t_{RESW}	20		20		-	ns
NMI reset setup time	t_{NMIRS}	200		235		-	ns
NMI reset hold time	t_{NMIRH}	200		200		-	ns
NMI setup time	t_{NMIS}	100		110		-	ns
NMI hold time	t_{NMIH}	50		50		-	ns
$\overline{\text{IRQ0}}$ - $\overline{\text{IRQ7}}$ setup time (edge detection time)	t_{IRQES}	100		110		-	ns
$\overline{\text{IRQ0}}$ - $\overline{\text{IRQ7}}$ setup time (level detection time)	t_{IRQLS}	100		110		-	ns
$\overline{\text{IRQ0}}$ - $\overline{\text{IRQ7}}$ hold time	t_{IRQEH}	50		50		-	ns
$\overline{\text{IRQOUT}}$ output delay time	t_{IRQOD}		50		50	-	ns
Bus request setup time	t_{BRQS}	50		55		-	ns
Bus acknowledge delay time 1	t_{BACD1}		50		55	-	ns
Bus acknowledge delay time 2	t_{BACD2}		50		55	-	ns
Bus 3-state delay time	t_{BZD}		50		55	-	ns

*1 Typical outputs measured with 50pF load

Table 4-4 Bus Timing

Characteristic	Symbol	SH-7034 20MHz		HP 64785A Worst Case		HP 64785A Typical (*1)	Unit
		Min	Max	Min	Max		
Address delay time	t_{AD}		20		30	13	ns
$\overline{CS}$ delay time 1	t_{CSD1}		25		30	10	ns
$\overline{CS}$ delay time 2	t_{CSD2}		25		30	6	ns
$\overline{CS}$ delay time 3	t_{CSD3}		20		25	9	ns
$\overline{CS}$ delay time 4	t_{CSD4}		20		25	5	ns
Access time 1 from read strobe (35% duty)	t_{RDAC1}	12.5		2.5		12.5	ns
Access time 1 from read strobe (50% duty)	t_{RDAC1}	5		-5		5	ns
Access time 2 from read strobe (35% duty)	t_{RDAC2}	62.5		52.5		62.5	ns
Access time 2 from read strobe (50% duty)	t_{RDAC2}	55		45		55	ns
Read strobe delay time	t_{RSD}		20		25	8	ns
Read data setup time	t_{RDS}	15		25		15	ns
Read data hold time	t_{RDH}	0		0		0	ns
Write strobe delay time 1	t_{WSD1}		20		25	10	ns
Write strobe delay time 2	t_{WSD2}		20		25	6	ns
Write strobe delay time 3	t_{WSD3}		20		25	11	ns
Write strobe delay time 4	t_{WSD4}		20		25	8	ns

Table 4-4 Bus Timing (Cont’d)

Characteristic	Symbol	SH-7034 20MHz		HP 64785A			Unit
				Worst Case		Typical (*1)	
		Min	Max	Min	Max		
Write data delay time 1	t_{WDD1}		35		40	21	ns
Write data delay time 2	t_{WDD2}		20		40	23	ns
Write data hold time	t_{WDH}	0		-5		2	ns
Parity output delay time 1	t_{WPDD1}		40		45	24	ns
Parity output delay time 2	t_{WPDD2}		20		25	11	ns
Parity output hold time	t_{WPDH}	0		-5		3	ns
Wait setup time	t_{WTS}	14		24		10	ns
Wait hold time	t_{WTH}	10		10		10	ns
Read data access time 1	t_{ACC1}	20		5		20	ns
Read data access time 2	t_{ACC2}	70		55		70	ns
$\overline{RAS}$ delay time 1	t_{RASD1}		20		25	8	ns
$\overline{RAS}$ delay time 2	t_{RASD2}		30		35	14	ns
$\overline{CAS}$ delay time 1	t_{CASD1}		20		25	6	ns
$\overline{CAS}$ delay time 2	t_{CASD2}		20		25	9	ns
$\overline{CAS}$ delay time 3	t_{CASD3}		20		25	8	ns
Column address setup time	t_{CAC1}	0		-5		13	ns

Table 4-4 Bus Timing (Cont'd)

Characteristic	Symbol	SH-7034 20MHz		HP 64785A Worst Case		HP 64785A Typical (*1)	Unit
		Min	Max	Min	Max		
$\overline{CAS}$ to read data access time 1 (35% duty)	t_{CAC1}	13.5		3.5		13.5	ns
$\overline{CAS}$ to read data access time 1 (50% duty)	t_{CAC1}	6		-4		6	ns
$\overline{CAS}$ to read data access time 2	t_{CAC2}	25		15		25	ns
$\overline{RAS}$ to read data access time 1	t_{RAC1}	55		45		55	ns
$\overline{RAS}$ to read data access time 2	t_{RAC2}	105		95		105	ns
High-speed page mode $\overline{CAS}$ precharge time 1	t_{CP}	12.5		-		24	ns
$\overline{AH}$ delay time 1	t_{AHD1}		20		25	6	ns
$\overline{AH}$ delay time 2	t_{AHD2}		20		25	8	ns
Multiplexed address delay time	t_{MAD}		30		35	16	ns
Multiplexed address hold time	t_{MAH}	0		-5		6	ns
DACK0-DACK1 delay time 1	t_{DACD1}		23		28	-	ns
DACK0-DACK1 delay time 2	t_{DACD2}		23		28	-	ns
DACK0-DACK1 delay time 3	t_{DACD3}		20		25	-	ns
DACK0-DACK1 delay time 4	t_{DACD4}		20		25	-	ns
DACK0-DACK1 delay time 5	t_{DACD5}		20		25	-	ns

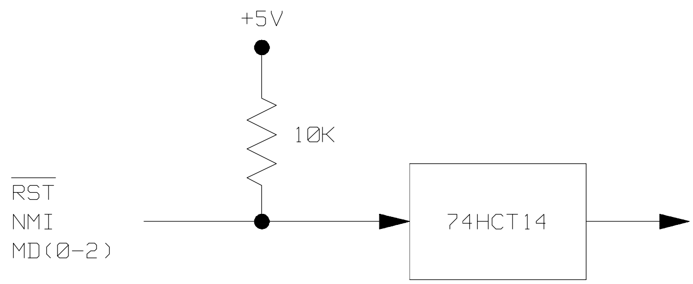
Table 4-4 Bus Timing (Cont'd)

Characteristic	Symbol	SH-7034 20MHz		HP 64785A			Unit
				Worst Case		Typical (*1)	
		Min	Max	Min	Max		
Read delay time (35% duty)	t_{RDD}		29.5		34.5	27	ns
Read delay time (50% duty)	t_{RDD}		40		45	35	ns
Data setup time for $\overline{CAS}$	t_{DS}	0		-5		6	ns
$\overline{CAS}$ setup time for $\overline{RAS}$	t_{CSR}	10		5		19	ns
Row address setup time	t_{RAH}	10		5		20	ns
Write command hold time	t_{WCH}	15		10		31	ns
Write command setup time (35% duty)	t_{WCS}	0		-5		7	ns
Write command setup time (50% duty)	t_{WCS}	0		-5		14	ns

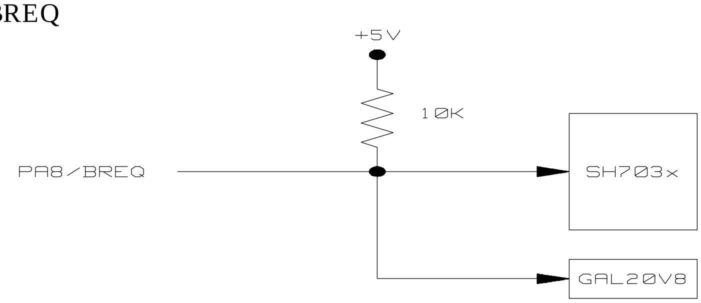
*1 Typical outputs measured with 50pF load

Target System Interface

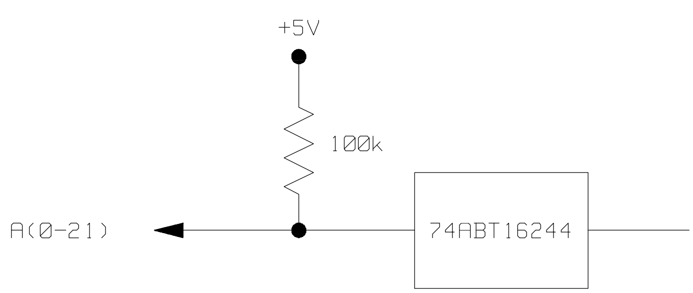
$\overline{RES}$
NMI
MD0-2



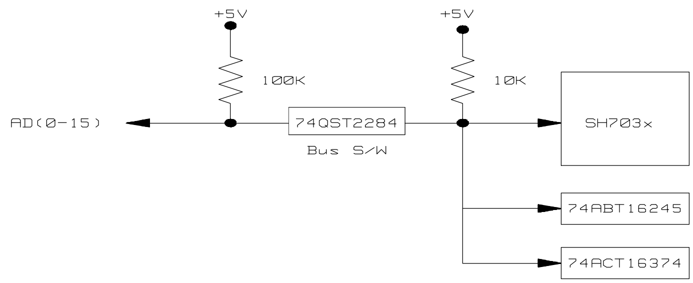
PA8/$\overline{BREQ}$



A0-A21



AD0-AD15



AVcc

Other signals

Notes

A

SH-7000 Emulator Specific Command Syntax

The following pages contain descriptions of command syntax specific to the SH-7000 emulator. The following syntax items are included (several items are part of other command syntax):

- <ACCESS_MODE>. May be specified in the **mo** (display and access mode), **m** (memory) commands. The access mode is used when the **m** commands modify target memory or I/O locations.

- <CONFIG_ITEMS>. May be specified in the **cf** (emulator configuration) and **help cf** commands.

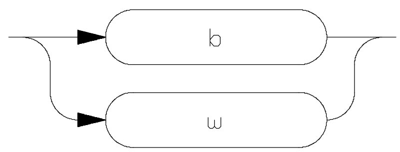
- <DISPLAY_MODE>. May be specified in the **mo** (display and access mode), **m** (memory), and **ser** (search memory for data) commands. The display mode is used when memory locations are displayed or modified.

- <REG_NAME> and <REG_CLASS>. May be specified in the **reg** (register) command.

ACCESS_MODE

Summary Specify cycles used by monitor when accessing target system memory or I/O.

Syntax



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

Parameters

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

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

Note

When the **<ACCESS_MODE>** is **w**, modifying target memory will fail if you try to modify memory from an 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 the default **b** as **<ACCESS_MODE>**.

Defaults

In the SH-7000, the **<ACCESS_MODE>** is **b** at power up initialization. Access mode specifications are saved; that is, when a command changes the access mode, the new access mode becomes the current default.

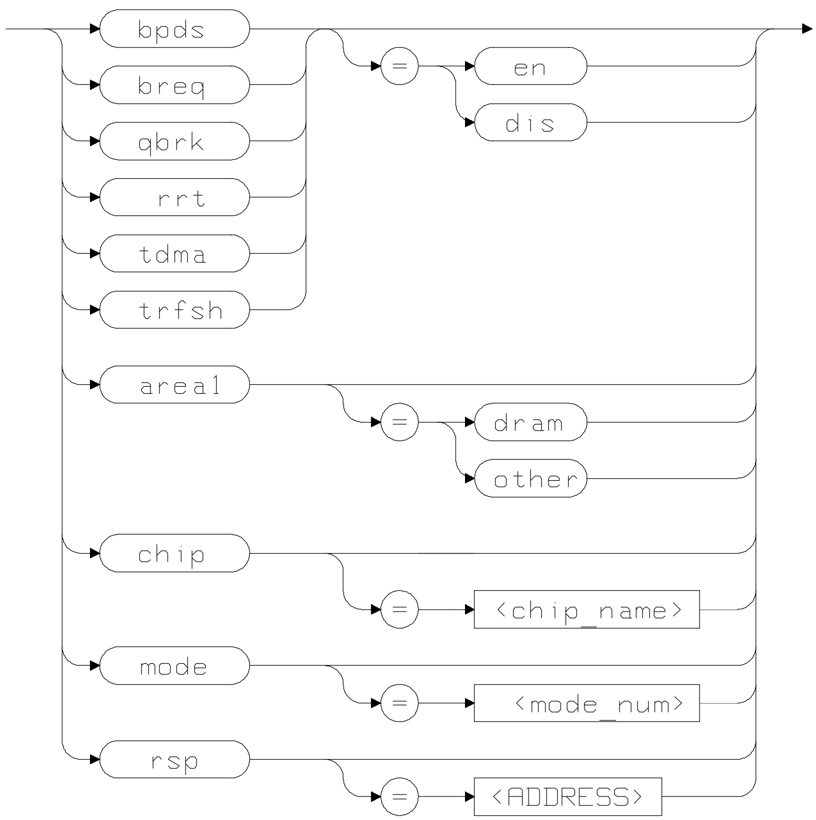
Related Commands

mo (specify display and access modes)

CONFIG_ITEMS

Summary SH-7000 emulator configuration items.

Syntax



Function The **<CONFIG_ITEMS>** are the SH-7000 specific configuration items which can be displayed/modified using the **cf** (emulator configuration) command. If the "=" portion of the syntax is not used, the current value of the configuration item is displayed.

Parameters

area1 **Memory type of area1.** This configuration item selects the memory type of the area1.

Setting **area1** equal to **dram** specifies that the memory mapper will treat the area 1 as 16M byte address space.

Setting **area1** equal to **other** specifies that the memory mapper will treat the area 1 as 4M byte address space.

Note Execution of this configuration option will drive the emulator into a reset state and all map terms will be removed.

---

bpds **Breakpoint at delay slot.** This configuration item allows you to specify a breakpoint at delay slot.

Setting **bpds** equal to **en** allows you to set the breakpoint at any address location.

Setting **bpds** equal to **dis** specified that the ’bp’ 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.

Note

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

---

breq **Function of PA8/BREQ pin.** This configuration item specfies the function of PA8/BREQ pin.

Setting **breq** equal to **en** specifies that the PA8/BREQ pin is used as BREQ input in your target system.

Setting **breq** equal to **dis** specifies that the PA8/BREQ pin is used as PA8 input/output or is not used in your target system.

chip **Emulation processor type** This configuration item allows you specify the emulation processor type.

Setting **chip** equal to **7032** specifies the SH-7000 emulator emulate SH-7032 processor.

Setting **chip** equal to **7034** specifies the SH-7000 emulator emulate SH-7034 processor.

Note

If the emulation processor without on-chip ROM is selected and the processor operation mode is configured as **mode 2** using the 'cf mode' command, the emulator will ignore the mode configuration option and the emulation processor will be operated in **mode 0**.

Note

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

Note

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

mode **Processor operation mode.** This configuration item specifies the processor operation mode.

Setting **mode** equal to **0** specifies that the emulator operates in MCU mode 0 (8bit data bus in area1).

Setting **mode** equal to **1** specifies that the emulator operates in MCU mode 1 (16bit data bus in area1).

Setting **mode** equal to **2** specifies that the emulator operates in MCU mode 2 (on-chip ROM in area1).

Note

If **mode 2** and the emulation 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 You can not select **mode_2**, when you configure the processor type as **7032**.

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

Note Execution of this configuration option will drive the emulator into a reset state and all map terms will be removed.

qbrk **Quick temporary break.** This configuration item specifies to use quick temporary break or not.

Setting **qbrk** equal to **en** specifies that a temporary break to the monitor for an operation such as display registers will spend a very small amount of time in the monitor. The CMB does not work in this setting.

Setting **qbrk** equal to **dis** specifies that a temporary break to the monitor will spend more time in the monitor.

Note Execution of this configuration option will drive the emulator into a reset state.

rrt **Restrict to Real-Time Runs.** This configuration item allows you to 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.

Setting **rrt** equal to **en** specifies that the emulator's execution is restricted to real-time. In this setting, commands which access target system resources (display/modify registers, display/modify memory or I/O) are not allowed.

setting **rrt** equal to **dis** specifies that the emulator breaks to the monitor during program execution.

rsp **Reset value for stack pointer.** This configuration item allows you to specify a value to which the stack pointer will be set upon the transition from emulation reset into the emulation monitor.

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

tdma **Trace internal DMA cycles.** This configuration item allows you to specify whether the analyzer traces in-chip DMAC cycles or not.

Setting **tdma** equal to **en** specifies that the analyzer traces on-chip DMAC cycles.

Setting **tdma** equal to **dis** 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 **tdma** equal to **dis**, the emulator will not break to monitor upon a write to ROM or guarded memory by internal DMAC

trfsh — **Trace refresh cycles.** This configuration item allows you to specify whether the analyzer traces refresh cycles.

Setting **trfsh** equal to **en** specifies that the analyzer traces refresh cycles.

Setting **trfsh** equal to **dis** specifies that the analyzer does not refresh cycles.

Defaults

The default values of SH-7000 emulator configuration items are listed below.

```
cf area1=other
cf bpds=dis
cf breq=dis
cf chip=7032
cf mode=0
cf qbrk=dis
cf rrt=dis
cf rsp=0
cf tdma=en
cf trfsh=en
```

Related Commands

help

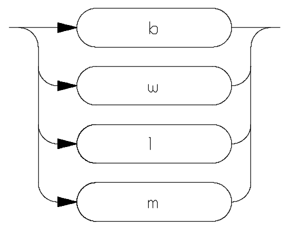
You can get an on line help information for particular configuration items by typ *g:*

R>**help cf <CONFIG_ITEM>** or **? cf <CONFIG_ITEM>**

DISPLAY_MODE

Summary Specify the memory display format or the size of memory locations to be modified.

Syntax



Function The **<DISPLAY_MODE>** specifies the format of the memory display or the size of the memory which gets changed when memory is modified.

Parameters

b — **Byte**. Memory is displayed in a byte format, and when memory locations are modified, bytes are changed.

w — **Word**. Memory is displayed in a word format, and when memory locations are modified, words are changed.

l — **Long Word**. Memory is displayed in a long word format, and when memory locations are modified, long words are changed.

m — **Mnemonic**. Memory is displayed in mnemonic format; that is, the contents of memory locations are inverse-assembled into mnemonics and operands. When memory locations are modified, the last non-mnemonic display mode specification is used. You cannot specify this display mode in the **ser** (search memory for data) command.

Defaults

At powerup or after init,in the SH-7000 emulator, the **<ACCESS_MODE>** and **<DISPLAY_MODE>** are **b**.

Display mode specifications are saved; that is, when a command changes the display mode, the new display mode becomes the current default.

Related Commands

mo (specify access and display modes)

m (memory display/modify)

ser (search memory for data)

REGISTER CLASS and NAME

Summary SH-70 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)
adcsr	A/D control/status register	
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

Function

The **<REG_CLASS>** names may be used in the **reg**(register) command to display a class of SH-7000 registers.

The **<REG_NAME>** names may be used with the **reg** command to either display or modify the contents of SH-7000 registers.

Refer to your SH-7000 user's manual for complete details on the use of the SH-7000 registers.

Related Commands

reg (register display/modify)

Index