HP 64739

H8/536 Emulator PC Interface

User's Guide

HEWLETT PACKARD

HP Part No. 64739-97004
Printed in U.S.A.
February 1994

Edition 2

Notice

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

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

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 **64739-97001, February 1990**

Edition 2 **64739-97004, February 1994**

Using This Manual

This manual introduces you to the following emulators as used with the PC Interface.

- HP 64739A H8/536 emulator
- HP 64739B H8/536S emulator

Throughout this documentation, the following names are used to denote the microprocessors listed in the following table of supported microprocessors.

Model	Supported Microprocessors	Reffered to as
HP 64739A(H8/536 emulator)	HD6475368CP HD6435368CP HD6475348CP HD6435348CP	H8/536 H8/536 H8/534 H8/534
HP 64739B(H8/536S emulator)	HD6475368CP HD6435368CP HD6475348CP HD6435348CP HD6475368SCP HD6435368SCP HD6475348SCP HD6435348SCP	H8/536 H8/536 H8/534 H8/534 H8/536S H8/536S H8/534S H8/534S

For the most part, the H8/536 and H8/536S emulators all operate the same way. Differences of between the emulators are described where they exist. Both the H8/536 and H8/536S emulators will be referred to as the "H8/536 emulator". In the specific instances where H8/536S emulator differs from H8/536 emulator, it will be described as "H8/536S emulator".

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. Topics include: restricting the emulator to real-time execution, and selecting a target system clock source.
- Show you how to use the emulator in-circuit (connected to a target system).
- Describe the command syntax which is specific to the H8/536 emulator.

This manual does not:

- Show you how to use every PC Interface command and option; the PC Interface is described in the *HP 64700 Emulators PC Interface: User's Reference.*

Organization

Chapter 1 **Introduction to the H8/536 Emulator.** This chapter lists the H8/536 emulator features and describes how they can help you in developing new hardware and software.

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 use basic emulation commands.

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

Chapter 4 **Configuring the Emulator.** You can configure the emulator to adapt it to your specific development needs. This chapter describes the options available when configuring the emulator and how to save and restore particular configurations.

Chapter 5 **Using the Emulator.** This chapter describes emulation topics which are not covered in the "Getting Started" chapter (for example, coordinated measurements and storing memory).

Appendix A **File Format Readers.** This appendix describes how to use the HP 64869 Format Reader from MS-DOS, load absolute files into the emulator, use global and local symbols with the PC Interface.

Notes

Contents

Illustrations

Tables

Notes

1

Introduction to the H8/536 Emulator

Introduction

The topics in this chapter include:

- Purpose of the H8/536 emulator.
- Features of the H8/536 emulator.

Purpose of the H8/536 Emulator

The H8/536 emulator is designed to replace the H8/536 microprocessor 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.

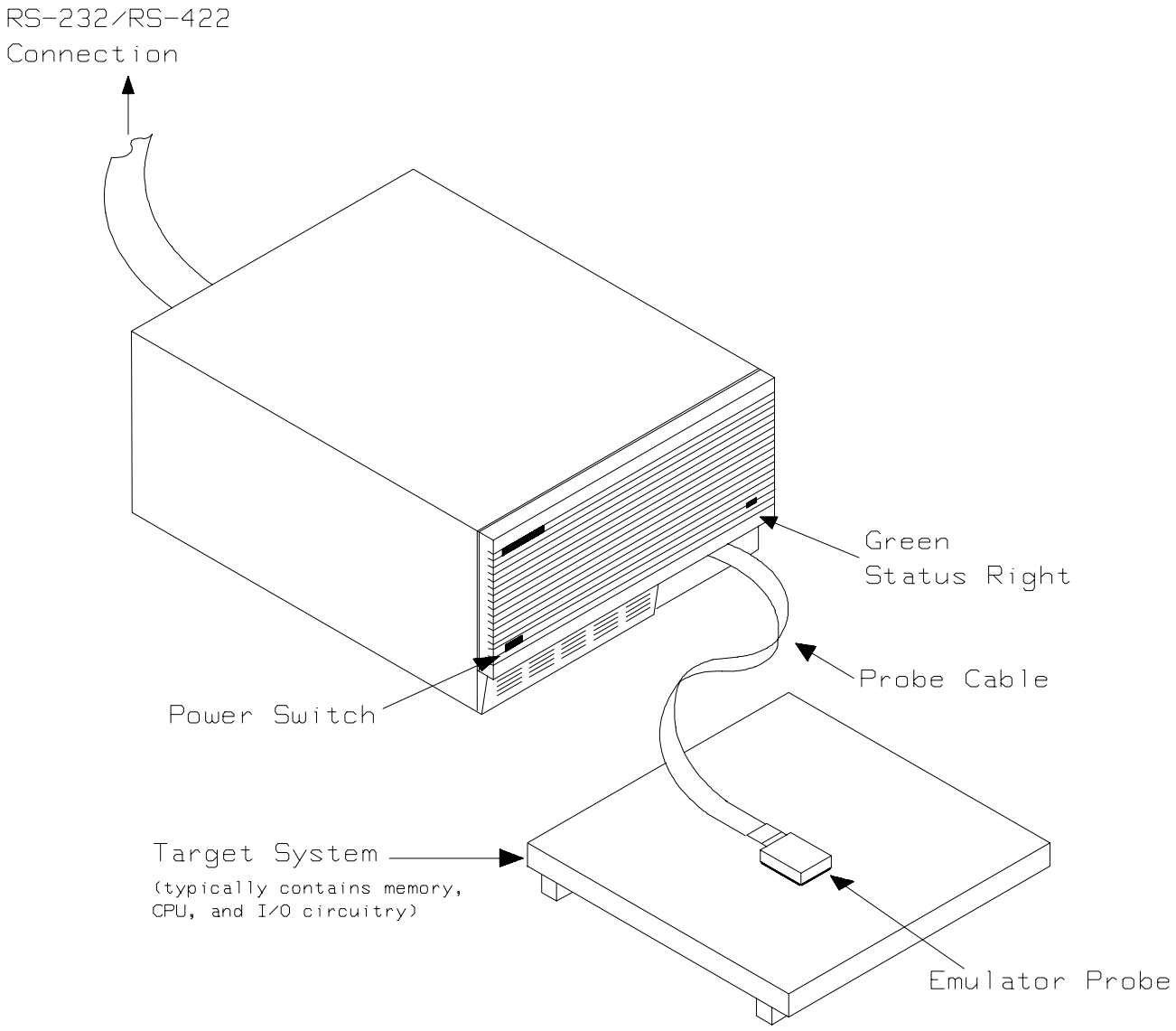


Figure 1-1. HP 64739 Emulator for the H8/536 Processor

Features of the H8/536 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 H8/536 emulator supports the microprocessors listed in Table 1-1.

Table 1-1. Supported Microprocessors

Model	Supported Microprocessors	Reffered to as
HP 64739A(H8/536 emulator)	HD6475368CP HD6435368CP HD6475348CP HD6435348CP	H8/536 H8/536 H8/534 H8/534
HP 64739B(H8/536S emulator)	HD6475368CP HD6435368CP HD6475348CP HD6435348CP HD6475368SCP HD6435368SCP HD6475348SCP HD6435348SCP	H8/536 H8/536 H8/534 H8/534 H8/536S H8/536S H8/534S H8/534S

Clock Speeds

You can select whether the emulator will be clocked by the internal clock source or by the external clock source on your target system. You must use a clock input conforming to the specification of Table 1-2.

When you use an external crystal, you need to input conforming to the specification of microprocessor.

Table 1-2. Clock Speeds

Clock source	Model	Microprocessor	Clock Speed
Internal	HP 64739A (H8/536 emulator)	H8/536 H8/534	10MHz (System clock)
	HP 64739B (H8/536S emulator)	H8/536 H8/534 H8/536S H8/534S	10MHz (System clock)
External	HP 64739A (H8/536 emulator)	H8/536 H8/534	From 0.5 up to 10MHz (System clock)
	HP 64739B (H8/536S emulator)	H8/536 H8/534	From 0.5 up to 10MHz (System clock)
		H8/536S H8/534S	From 0.5 up to 16MHz (System clock)

Emulation memory

The H8/536 emulator is used with one of the following Emulation Memory Cards.

- HP 64726A 128K byte Emulation Memory Card
- HP 64727A 512K byte Emulation Memory Card
- HP 64728A 1M byte Emulation Memory Card

You can define up to 16 memory ranges (at 256 byte boundaries and least 256 byte in length.) The emulator occupies 2K byte, which is used for monitor program, leaving 126K, 510K, 1022K byte of emulation memory which you may use. You can characterize memory range as emulation RAM (eram), emulation ROM (erom), target system RAM (tram), target system ROM (trom), or guarded memory (grd). 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.

Analysis

The H8/536 emulator is used with one of the following analyzers which allows you to trace code execution and processor activity.

- HP 64704A 80-channel Emulation Bus Analyzer
- HP 64703A 64-channel Emulation Bus Analyzer and 16-channel State/Timing Analyzer.
- HP 64794x 80-channel 8K/64K/256K Emulation Bus Analyzer.

The Emulation Bus Analyzer monitors the emulation processor using an internal analysis bus. The HP 64703A 64-channel Emulation Bus Analyzer and 16-channel State/Timing Analyzer allows you to probe up to 16 different lines in your target system.

Registers

You can display or modify the H8/536 internal register contents. This includes the ability to modify the program counter (PC) and code page register (CP) so you can control where the emulator begins executing a target system program.

Single-Step

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

Target System Interface

You can set the interface to the target system to be active or passive during background monitor operation. (See the "Configuring the Emulator" chapter for further details.)

Breakpoints

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

You can also define software breakpoints in your program. The emulator uses one of H8/536 undefined opcode (1B hex) as software breakpoint interrupt instruction. When you define a software breakpoint, the emulator places the breakpoint interrupt instruction (1B hex) at the specified address; after the breakpoint interrupt instruction causes emulator execution to break out of your program, the emulator replaces the original opcode. 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 signifies continuous execution of your program without interference from the emulator. (Such interference occurs when the emulator temporarily breaks into the monitor so that it can access register contents or target system memory.)

Emulator features performed in real time include: running and analyzer tracing.

Emulator features not performed in real time include: display or modify of target system memory; load/dump of any memory, display or modification of registers, and single step.

Limitations, Restrictions

DMA Support

Direct memory access to H8/536 emulation memory is not permitted.

Sleep and Software Stand-by Mode

When the emulator breaks into the emulation monitor, H8/536 microprocessor sleep or software stand-by mode is released and comes to normal processor mode.

Watch Dog Timer in Background

Watch dog timer suspends count up while the emulator is running in background monitor.

RAM Enable Bit

The internal RAM of H8/536 processor can be enabled/disabled by RAME (RAM enable bit). However, once you map the internal RAM area to emulation RAM, the emulator accesses emulation RAM even if the internal RAM is disabled by RAME.

2

Getting Started

Introduction

This chapter leads you through a basic, step by step tutorial that shows how to use the HP 64739 emulator with the PC Interface.

This chapter will:

- Tell you what must be done before you can use the emulator as shown in the tutorial examples.
- Describe the sample program used for this chapter's examples.
- Briefly describe how PC Interface commands are entered and how emulator status is displayed.

This chapter will show you how to:

- Start up the PC Interface from the MS-DOS prompt.
- Define (map) emulation and target system memory.
- Load programs into emulation and target system memory.
- Enter emulation commands to view execution of the sample program.

Before You Begin

Prerequisites

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

1. Connected the emulator to your computer. The *HP 64700 Emulators: Hardware Installation and Configuration* manual shows you how to do this.

2. Installed the PC Interface software on your computer. Software installation instructions are shipped with the media containing the PC Interface software. The *HP 64700 Emulators PC Interface: User's Reference* manual contains additional information on the installation and setup of the PC Interface.

3. In addition, it is recommended, although not required, that you read and understand the concepts of emulation presented in the *HP 64700 Emulators: System Overview* manual. The *System Overview* also covers HP 64700 Series system architecture. A brief understanding of these concepts may help avoid questions later.

   You should read the *HP 64700 Emulators PC Interface: User's Reference* manual to learn how to use the PC Interface in general. For the most part, this manual contains information specific to the H8/536 emulator.

A Look at the Sample Program

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

Using the various features of the emulator, we will show you how to load this program into emulation memory, execute it, monitor the program's operation with the analyzer, and simulate entry of different commands by using the "**M**emory **M**odify" emulation command.

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

                .SECTION    Table,DATA
Msgs
Msg_A           .SDATA      "Command A entered "
Msg_B           .SDATA      "Entered B command "
Msg_I           .SDATA      "Invalid Command "
End_Msgs


                .SECTION    Prog,CODE
;***********************************************
;* Sets up the stack pointer.
;***********************************************
Init            MOV:G.W     #Stack,R7
;***********************************************
;* Clear previous command.
;***********************************************
Read_Cmd        MOV:G.B     #0,@Cmd_Input
;***********************************************
;* Read command input byte.  If no command has
;* been entered, continue to scan for input.
;***********************************************
Scan            MOV:G.B     @Cmd_Input,R0
                BEQ         Scan
;***********************************************
;* A command has been entered.  Check if it is
;* command A, command B, or invalid.
;***********************************************
Exe_Cmd         CMP:E.B     #H'41,R0
                BEQ         Cmd_A
                CMP:E.B     #H'42,R0
                BEQ         Cmd_B
                BRA         Cmd_I
;***********************************************
;* Command A is entered.  R1 = the number of
;* bytes in message A.  R4 = location of the
;* message.  Jump to the routine which writes
;* the messages.
;***********************************************
Cmd_A           MOV:I.W     #Msg_B-Msg_A-1,R1
                MOV:I.W     #Msg_A,R4
                BRA         Write_Msg
;***********************************************
;* Command B is entered.
;***********************************************
Cmd_B           MOV:I.W     #Msg_I-Msg_B-1,R1
                MOV:I.W     #Msg_B,R4
                BRA         Write_Msg
```

Figure 2-2. Sample Program Listing

```
;*********************************************
;* An invalid command is entered.
;*********************************************
Cmd_I           MOV:I.W     #End_Msgs-Msg_I-1,R1
                MOV:I.W     #Msg_I,R4
;*********************************************
;* Message is written to the destination.
;*********************************************
Write_Msg       MOV:I.W     #Msg_Dest,R5
Again           MOV:G.B     @R4+,R3
                MOV:G.B     R3,@R5+
                SCB/EQ      R1,Again
;*********************************************
;* The rest of the destination area is filled
;* with zeros.
;*********************************************
Fill_Dest       MOV:G.B     #0,@R5+
                CMP:I.W     #Msg_Dest+H'20,R5
                BNE         Fill_Dest
;*********************************************
;* Go back and scan for next command.
;*********************************************
                BRA         Read_Cmd

                .SECTION    Data,COMMON
;*********************************************
;* Command input byte.
;*********************************************
Cmd_Input       .RES.B      1
                .RES.B      1
;*********************************************
;* Destination of the command messages.
;*********************************************
Msg_Dest        .RES.W      H'3E
Stack           .RES.W      1       ; Stack area.
```

Figure 2-1. Sample Program Listing (Cont'd)

Data Declarations

The "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 instruction at the **Init** label initializes the stack pointer.

Reading Input

The instruction at the **Read_Cmd** 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 0 hex).

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 41 hex), execution is transferred to the instructions at **Cmd_A**.

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

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

The instructions at **Cmd_A, Cmd_B,** and **Cmd_I** each load register R1 with the length of the message to be displayed and register R4 with the starting location of the appropriate message. Then, execution transfers to **Write_Msg** which writes the appropriate message to the destination location, **Msg_Dest**.

After the message is written, the instructions at **Fill_Dest** fill the remaining destination locations with zeros. (The entire destination area is 20 hex bytes long.) Then, the program branches back to read the next command.

Sample Program Assembly

The sample program is written for and assembled with the HP 64869 H8/500 Assembler/Linkage Editor. For example, the following command was used to assemble the sample program.

```
C>h8asm cmd_rds.src /DEBUG <RETURN>
```

In addition to the assembler listing (cmd_rds.lis), the "cmd_rds.obj" relocatable file is created.

Linking the Sample Program

The sample program can be linked with following command and generates an absolute file. The contents of "cmd_rds.k" linkage editor subcommand file is shown in Figure 2-2.

```
C>h8lnk /SUBCOMMAND=cmd_rds.k <RETURN>
```

In addition to the linker load map listing (cmd_rds.map), the "cmd_rds.abs" absolute file is created.

```
debug
input cmd_rds
start Prog(1000),Table(1100),Data(0FC00)
print cmd_rds
output cmd_rds
exit
```

Figure 2-2. Linkage Editor Subcommand File

Note

You need to specify DEBUG command line option to both assembler and linker command to generate local symbol information. The DEBUG option for the assembler and linker direct to include local symbol information to the object file.

Starting Up the PC Interface

If you have set up the emulator device table and the **HPTABLES** shell environment variable as shown in the *HP 64700 Emulators PC Interface: User's Reference*, you can start up the H8/536 PC Interface by entering the following command from the MS-DOS prompt:

```
pch8536 <emulname>
```

In the command above, **pch8536** is the command to start the PC Interface; "<emulname>" is the logical emulator name given in the emulator device table. (To start version of the PC Interface that supports external timing analysys, substitute **pth8536** for **pch8536** in this command.) If this command is successful, you will see the display shown in figure 2-3. If this command is not successful, you will be given an error message and returned to the MS-DOS prompt.

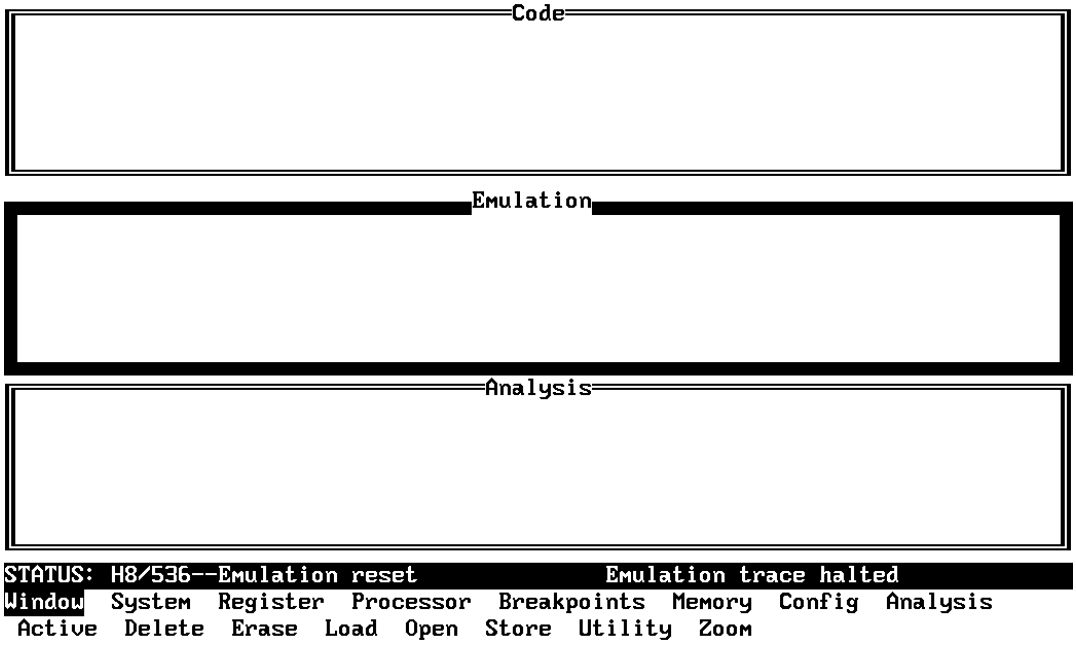


Figure 2-3. PC Interface Display

Selecting PC Interface Commands

This manual will tell you to "select" commands. You can select commands or command options by either using the left and right arrow keys to highlight the option and press the **Enter** key, or you can simply type the first letter of that option. If you select the wrong option, you can press the **ESC** key to move back up the command tree.

When a command or command option is highlighted, a short message describing that option is shown on the bottom line of the display.

Emulator Status

The status of the emulator is shown on the line above the command options. The PC Interface periodically checks the status of the emulator and updates the status line.

Modifying Configuration

You need to set up the emulation configuration before using the sample program. To access the emulation configuration display, enter:

```
Config, General
```

Defining the Reset Value for the Stack Pointer

Even though the H8/536 emulator has a background monitor, it requires you to define a stack pointer.

Use the arrow keys to move the cursorto the "Reset value for Stack Pointer" field, type **0fc80** and press **Enter**.

The stack pointer value will be set to the stack pointer (SP) on entrance to the emulation monitor initiated RESET state (the "Emulation reset" status).

Selecting your Processor

You need to select the processor you are going to emulate. Use the arrow keys to move the cursor to the "Processor type?" field. Use the **TAB** key to select the processor you are going to emulate.

Saving the Configuration

To save the configuration, use the **Enter** key to exit the field in the last field. (The **End** key on Vectra keyboards moves the cursor directly to the last field.)

Mapping Memory

The H8/536 emulator contains 126 kilobytes of high-speed emulation memory (no wait states required) that can be mapped at a resolution of 256 bytes.

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

Blocks of memory can also be characterized as guarded memory. Guarded memory accesses will generate "break to monitor" requests. Writes to ROM will generate "break to monitor" requests if the "Enable breaks on writes to ROM?" configuration item is enabled (see the "Configuring the Emulator" chapter).

The memory mapper allows you to define up to 16 different map terms.

Note

When you use the H8/536 internal ROM and RAM, you must map memory space where internal ROM and RAM are located as each emulation ROM and RAM.

Which Memory Locations Should Be Mapped?

Typically, assemblers generate relocatable files and linkers combine relocatable files to form the absolute file. The linker load map listing will show what locations your program will occupy in memory. A part of linker load map listing for the sample program (cmd_rds.map) is shown in Figure 2-4.

```
                    ***  LINKAGE   EDITOR LINK MAP  LIST   ***

SECTION     NAME                  START      -      END         LENGTH
                                        UNIT NAME                    MODULE   NAME


ATTRIBUTE  :   CODE  NOSHR
Prog                          H'0000:1000   -   H'0000:1042   H'00000043
                                    cmd_rds                          cmd_rds
* TOTAL   ADDRESS   *         H'0000:1000   -   H'0000:1042   H'00000043


ATTRIBUTE  :   DATA  NOSHR
Table                         H'0000:1100   -   H'0000:1133   H'00000034
                                    cmd_rds                          cmd_rds
* TOTAL   ADDRESS   *         H'0000:1100   -   H'0000:1133   H'00000034


ATTRIBUTE  :   DATA  SHR
Data                          H'0000:FC00   -   H'0000:FC7F   H'00000080
                                    cmd_rds                          cmd_rds
* TOTAL   ADDRESS   *         H'0000:FC00   -   H'0000:FC7F   H'00000080
```

Figure 2-4. Sample Program Load Map Listing

From the load map listing, you can see that the sample program occupies locations in three address ranges. The code area, which contains the opcodes and operands which make up the sample program, occupies locations 1000 hex through 1042 hex. The data area, which contains the ASCII values of the messages the program displays, is occupies locations 1100 hex through 1133 hex. The destination area, which contains the command input byte and the locations of the message destination and the stack, occupies locations FC00 hex through FC7F hex.

Two mapper terms will be specified for the example program. Since the program writes to the destination locations, the mapper block containing the destination locations should not be characterized as ROM memory.

To map memory for the sample program, select:

Config, **M**ap, **M**odify

Using the arrow keys, move the cursor to the "address range" field of term 1. Enter:

```
1000..1fff
```

Move the cursor to the "memory type" field of term 1, and press the TAB key to select the **erom** (emulation ROM) type. Move the cursor to the "address range" field of term 2 and enter:

```
0fc00..0fcff
```

Move the cursor to the "memory type" field of term 2, and press the TAB key to select the **eram** (emulation RAM) type. To save your memory map, use the **Enter** key to exit the field in the lower right corner. (The **End** key on Vectra keyboards moves the cursor directly to the last field.) The memory configuration display is shown in Figure 2-5.

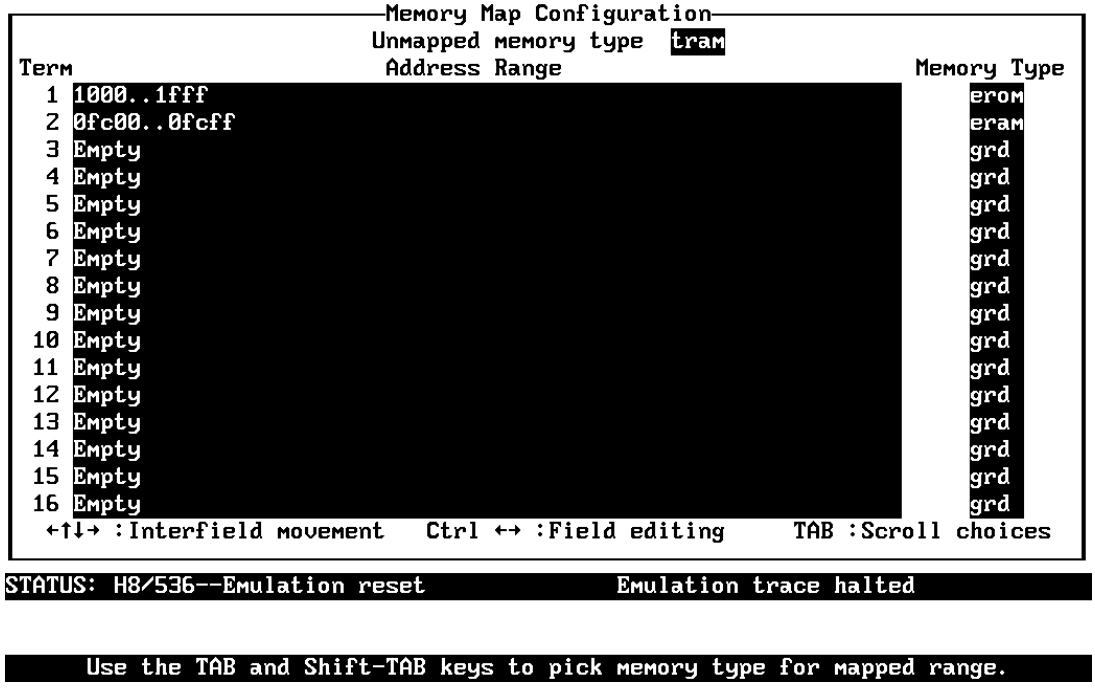


Figure 2-5. Memory Configuration Display

When mapping memory for your target system programs, you may wish to characterize emulation memory locations containing programs and constants (locations which should not be written to) as ROM. This will prevent programs and constants from being written over accidentally, and will cause breaks when instructions attempt to do so.

Note

The memory mapper re-assigns blocks of emulation memory after the insertion or deletion of mapper terms. For example, if you modified the contents of FC00 hex through FCFF hex above, deleted term 1, and displayed locations FC00 hex through FCFF hex, you would notice the contents of those locations are not the same as they were before deleting the mapper term.

Loading Programs into Memory

If you have already assembled and linked the sample program, you can load the absolute file by selecting:

```
Memory, Load
```

File Format

Enter the format of your absolute file. The emulator will accept absolute files in the following formats:

- HP 64869 absolute.
- HP absolute.
- Raw HP64000 absolute.
- Intel hexadecimal.
- Tektronix hexadecimal.
- Motorola S-records.

The HP 64869 absolute file is generated with HP 64869 H8/300 Assembler/Linkage Editor. For this tutorial, choose the HP 64869 format.

HP 64869 Format: When you load HP 64869 format files, the PC Interface creates files (whose base names are the same as the absolute file) with the extensions ".HPA" and ".HPS". The ".HPA" file is in a binary format that is compatible with the HP 64736 firmware. The ".HPS" file is an ASCII source file which contains the symbols to address mappings used by the PC Interface. Refer to "Using HP 64869 Format Reader" section in Appendix A for more information.

HP64000 Format: Your language tool may generate Raw HP64000 format absolute files (with extension .X, .L, .A). You can load these files by selecting "HP64000" or "Raw HP64000" as file format. When you select "HP64000", the PC Interface creates .HPA absolute file and .HPS symbol database. When you select "Raw HP64000", the PC Interface doesn't create these files.

Memory Type

The second field allows you to selectively load the portions of the absolute file which reside in emulation memory, target system memory, or both.

Since emulation memory is mapped for sample program locations, you can enter either "emulation" or "both".

Force Absolute File Read

This option is only available for HP 64869 and HP64000 formats. It forces the file format readers to regenarate the emulator absolute file (.hpa) and symbol data base (.hps) before loading the code. Normally, these files are only regenarated whenever the file you specify (the output of your language tools) is newer than the emulator absolute file and symbol data base.

For more information, refer to the "Using the HP 64869 Format Reader" section in Appendix A.

Absolute File Name

For most formats, you enter the name of your absolute file in the last field. Type **cmd_rds.abs**, and press **Enter** to start the memory load.

Using Symbols

The following pages show you how to display global and local symbols for the sample program. For more information on symbol display, refer to the *PC Interface Refernce*.

Displaying Global Symbols

When you load HP 64869 or HP64000 format absolute files into the emulator, the corresponding symbol database is also loaded.

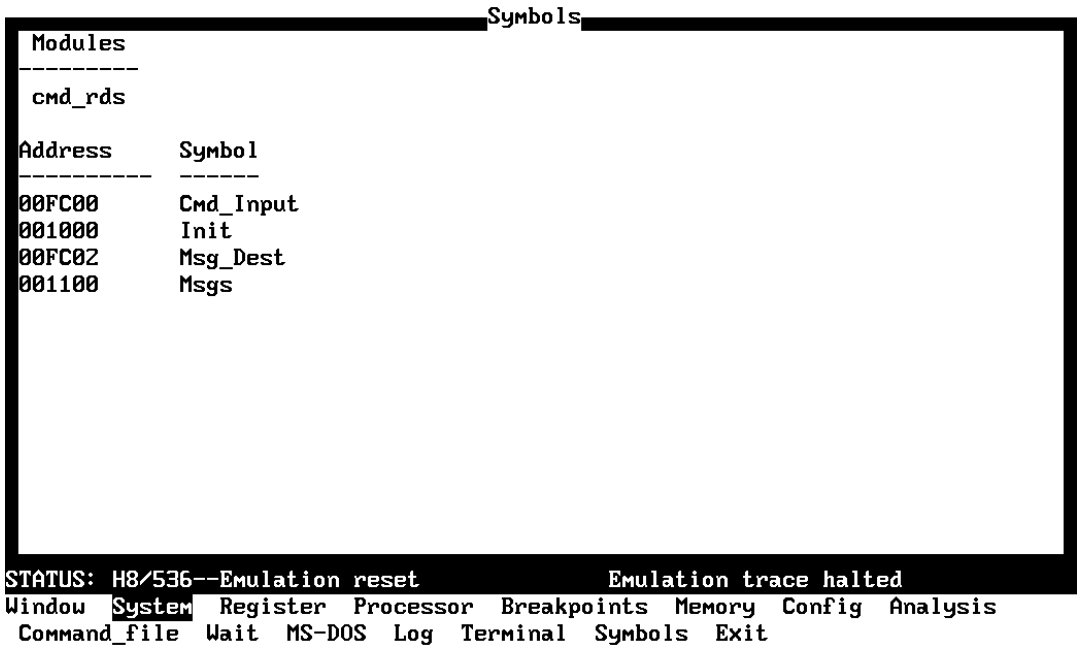
The symbols database can also be loaded with the "**S**ystem **S**ymbols **G**lobal **L**oad" command. This command is provided for situations where multiple absolute files are loaded into the emulator; it allows you to load the various sets of global symbols corresponding to the various absolute files. When global symbols are loaded into the emulator, information about previous global symbols is lost (that is, only one set of global symbols can be loaded at a time).

After global symbols are loaded, both global and local symbols can be used when entering expressions. Global symbols are entered as they appear in the source file or in the global symbols display.

To display global symbols, select:

```
System, Symbols, Global, Display
```

The symbols window automatically becomes the active window as a result of this command. You can press <CTRL>**z** to zoom the window. The resulting display follows.



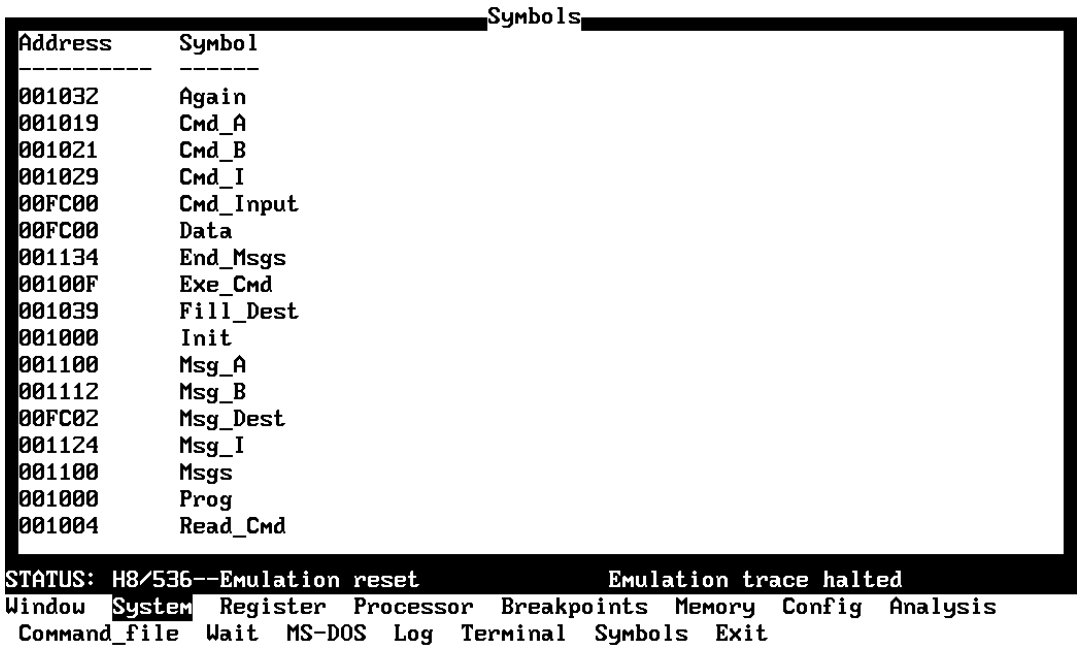
The global symbols display has two parts. The first parts lists all the modules that were linked to produce this object file. These module names are used by you when you want to refer to a local symbol, and are case-sensitive. The second part of the display lists all global symbols in this module. These names can be used in measurement specifications, and are case-snesitive. For example, if you wish to make a measurement using the symbol **Cmd_Input**, you must specify **Cmd_Input**. The strings **cmd_input** or **CMD_INPUT** are not valid symbol names here.

Displaying Local Symbols

To display local symbols, select:

```
System, Symbols, Local, Display
```

Enter the name of the module you want to specify (from the first part of the global symbols display; in this case, **cmd_rds**) and press **Enter**. The resulting display follows.



After you display local symbols with the "**S**ystem **S**ymbols **L**ocal **D**isplay" command, you can enter local symbols as they appear in the source file or local symbol display. When you display local symbols for a given module, that module becomes the default local symbol module.

If you have not displayed local symbols, you can still enter a local symbol by including the name of the module:

```
module_name:symbol
```

Remember that the only valid module names are those listed in the first part of the global symbols display, and are case-sensitive for compatibility with other systems (such as HP-UX).

When you include the name of an source file with a local symbol, that module becomes the default local symbol module, as with the "**S**ystem **S**ymbols **L**ocal **D**isplay" command.

Local symbols must be from assembly modules that form the absolute whose symbol database is currently loaded. Otherwise, no symbols will be found (even if the named assembler symbol file exists and contains information).

One thing to note: It is possible for a symbol to be local in one module and global in another, which may result in some confusion. For example, suppose symbol “XYZ” is a global in module A and a local in module B and that these modules link to form the absolute file. After you load the absolute file (and the corresponding symbol database), entering “XYZ” in an expression refers to the symbol from module A. Then, if you display local symbols from module B, entering “XYZ” in an expression refers to the symbol from module B, **not the global symbol**. Now, if you again want to enter “XYZ” to refer to the global symbol from module A, you must display the local symbols from module A (since the global symbol is also local to that module). Loading local symbols from a third module, if it was linked with modules A and B and did not contain an“XYZ” local symbol, would also cause “XYZ” to refer to the global symbol from module A.

Transfer Symbols to the Emulator

You can use the emulator’s symbol-handling capability to improve measurement displays. You do this by transferring the symbol database information to the emulator. To transfer the global symbol information to the emulator, use the command:

```
System, Symbols, Global, Transfer
```

Transfer the local symbol information for all modules by entering:

```
System, Symbols, Local, Transfer, All
```

You can find more information on emulator symbol handling commands in the *Emulator PC Interface Reference*.

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. To do this, select:

Memory, **D**isplay, **M**nemonic

Enter the address range "1000..1029". (You could also specify this address range using symbols, for example, **"Init..Cmd_I"** or **"Init..Init+29"**.) The emulation window automatically becomes the active window as a result of this command. You can press <CTRL>**z** to zoom the emulation window. The resulting display follows.

```
                                Emulation
Address     Symbol              Mnemonic
----------  ----------------    ---------------------------------------------
001000      Init                MOV:G.W #fc7e,R7
001004      md_rds:Read_Cmd     MOV:G.B #00,@fc00
001009      cmd_rds:Scan        MOV:G.B @fc00,R0
00100d      -                   BEQ cmd_rds:Scan
00100f      cmd_rds:Exe_Cmd     CMP:E.B #41,R0
001011      -                   BEQ cmd_rds:Cmd_A
001013      -                   CMP:E.B #42,R0
001015      -                   BEQ cmd_rds:Cmd_B
001017      -                   BRA cmd_rds:Cmd_I
001019      cmd_rds:Cmd_A       MOV:I.W #0011,R1
00101c      -                   MOV:I.W #1100,R4
00101f      -                   BRA cmd_rds:Write_Msg
001021      cmd_rds:Cmd_B       MOV:I.W #0011,R1
001024      -                   MOV:I.W #1112,R4
001027      -                   BRA cmd_rds:Write_Msg
001029      cmd_rds:Cmd_I       MOV:I.W #000f,R1

STATUS: H8/536--Emulation reset              Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Display  Modify  Load  Store  Copy  Find  Report
```

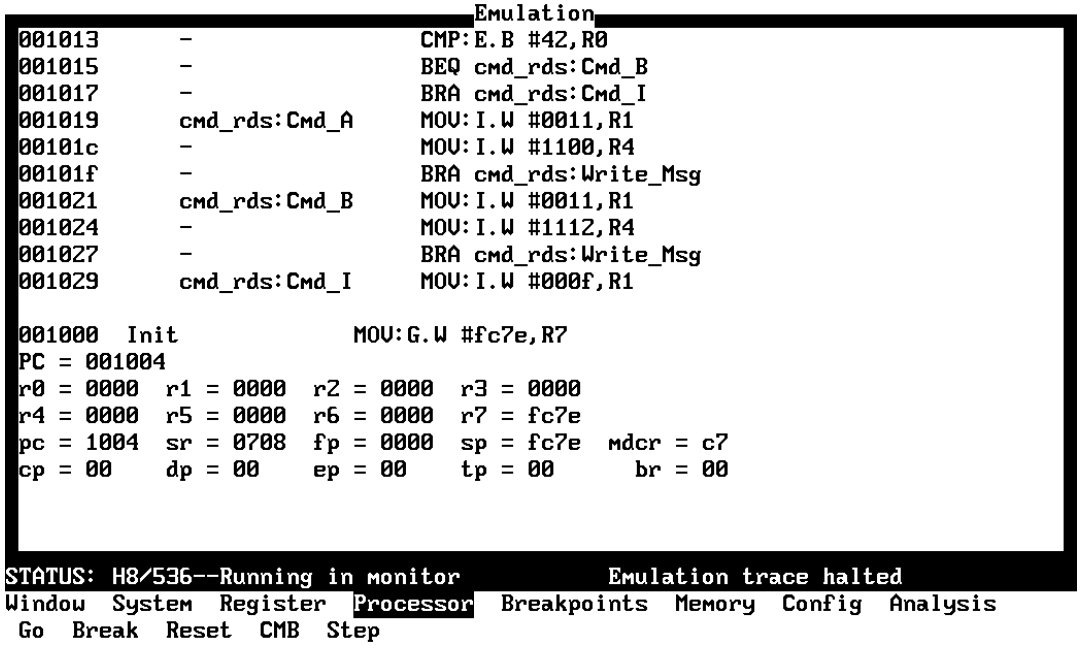
If you wish to view the rest of the sample program memory locations,you can select "**M**emory **D**isplay **M**nemonic" command again and enter the range from "102a..1042".

Stepping Through the Program

The emulator allows you to execute one instruction or a number of instructions with step command. To begin stepping through the sample program, select:

Processor, **S**tep, **A**ddress

Enter a step count of 1, enter the symbol **Init** (defined as a global in the source file), and press **Enter** to step from program's first address, 1000 hex. The executed instruction, the program counter address, and the resulting register contents are displayed as shown in the following listing.



Note

You cannot display registers if the processor is reset. Use the "**P**rocessor **B**reak" command to cause the emulator to start executing in the monitor.

You can display registers while the emulator is executing a user program (if execution is not restricted to real-time); emulator execution will temporarily break to the monitor.

To continue stepping through the program, you can select:

```
Processor, Step, Pc
```

After selecting the command above, you have an opportunity to change the previous step count. If you wish to step the same number of times, you can press **Enter** to start the step.

To save time when single-stepping, you can use the function key macro <F1>, which executes the command,

```
Processor, Step, Pc, 1
```

For more information, see the *Emulator PC Interface Reference* manual.

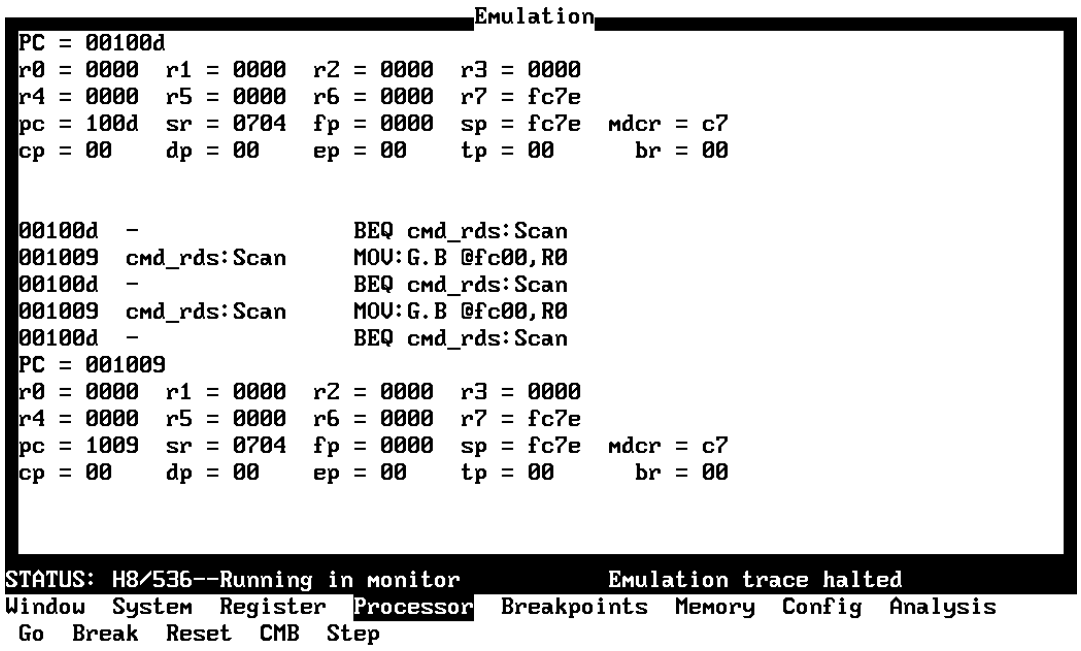
To repeat the previous command, you can press <CTRL>**r**.

Specifying a Step Count

If you wish to continue to step a number of times from the current program counter, select:

```
Processor, Step, Pc
```

The previous step count is displayed in the "number of instructions" field. You can enter a number from 1 through 99 to specify the number of times to step. Type 5 into the field, and press **Enter**. The resulting display follows.



When you specify step counts greater than 1, only the last register contents are displayed.

Modifying Memory

The preceding step commands show the sample program is executing in the **Scan** loop, where it continually reads the command input byte to check if a command has been entered. To simulate the entry of a sample program command, you can modify the command input byte by selecting:

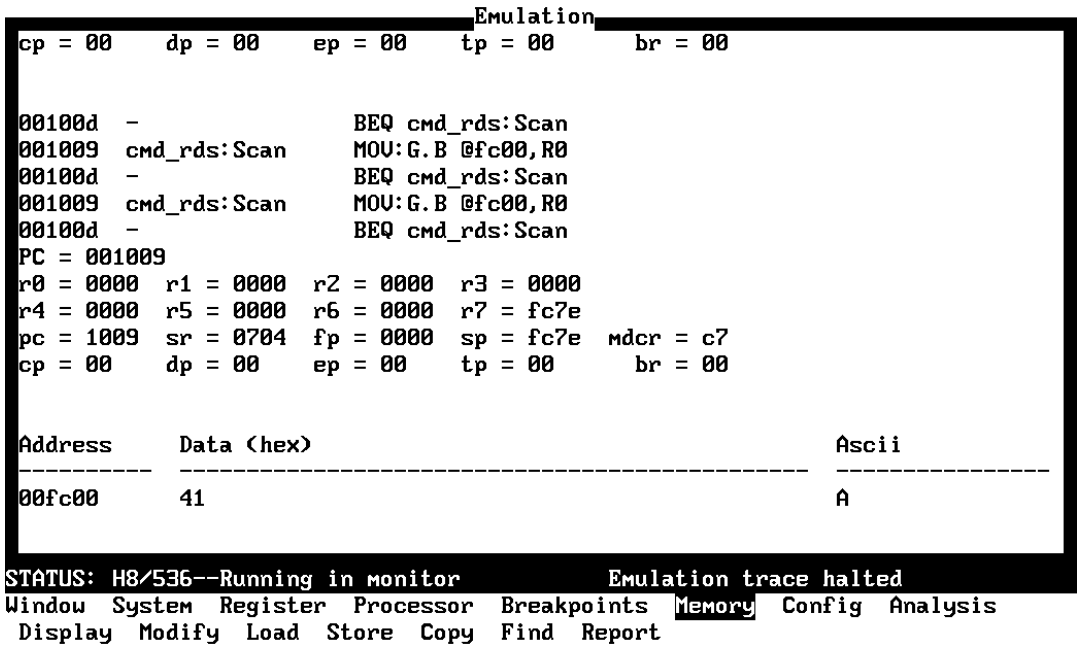
Memory, **M**odify, **B**yte

Now enter the address of the memory location to be modified, an equal sign, and new value of that location, for example, "Cmd_Input=41". (The **Cmd_Input** label was defined as a global symbol in the source file.)

To verify that 41 hex was indeed written to **Cmd_Input** (FC00 hex), select:

Memory, **D**isplay, **B**yte

Type the symbol **Cmd_Input**, and press **Enter**. The resulting display is shown below.



You can continue to step through the program as shown earlier in this chapter to view the instructions which are executed when an "A" (41 hex) command is entered.

Running the Program

To start the emulator executing the sample program, select:

```
Processor, Go, Pc
```

The status line will show that the emulator is "Running user program".

Searching Memory for Data

You can search the message destination locations to verify that the sample program writes the appropriate messages for the allowed commands. The command "A" (41 hex) was entered above, so the "Command A entered" message should have been written to the **Msg_Dest** locations. Because you must search for hexadecimal values, you will want to search for a sequence of characters which uniquely identify the message, for example, " A " or 20 hex, 41 hex, and 20 hex. To search the destination memory location for this sequence of characters, select:

```
Memory, Find
```

Enter the range of the memory locations to be searched, FC02 hex through FC21 hex, and enter the data 20 hex, 41 hex, and 20 hex. The resulting information in the memory window shows you that the message was indeed written as it was supposed to have been.

To verify that the sample program works for the other allowed commands, you can modify the command input byte to "B" and search for " B " (20 hex, 42 hex, and 20 hex), or you can modify the command input byte to "C" and search for "d C" (64 hex, 20 hex, and 43 hex).

Breaking into the Monitor

To break emulator execution from the sample program to the monitor program, select:

```
Processor, Break
```

The status line shows that the emulator is "Running in monitor".

While the break will occur as soon as possible, the actual stopping point may be many cycles after the break request (dependent on the type of instruction being executed and whether the processor is in a hold state).

Using Software Breakpoints

Software breakpoints are provided with one of H8/536 undefined opcode (1B hex) as breakpoint interrupt instruction.
When you define or enable a software breakpoint, the emulator will replace the opcode at the software breakpoint address with the breakpoint interrupt instruction.

When software breakpoints are enabled and emulator detects the breakpoint interrupt instruction (1B hex), it generates a break to background request which as with the "processor break" command. Since the system controller knows the locations of defined software breakpoints, it can determine whether the breakpoint interrupt instruction (1B hex) is a software breakpoint or opcode in your target program.

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

If it is an opcode of your target program, execution still breaks to the monitor, and an "Undefined software breakpoint" status message is displayed.

When software breakpoints are disabled, the emulator replaces the breakpoint interrupt instruction with the original opcode. Up to 32 software breakpoints may be defined.

Note

You must set software breakpoints only at memory locations which contain instruction opcodes (not operands or data). If a software breakpoint is set at a memory location which is not an instruction opcode, the software breakpoint instruction will never be executed and the break will never occur.

Note

Because software breakpoints are implemented by replacing opcodes with the undefined opcode (1B hex), you cannot define software breakpoints in target ROM. You can, however, use the Terminal Interface **cim** command to copy target ROM into emulation memory (see the *Terminal Interface: User's Reference* manual for information on the **cim** command).

Note

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

Defining a Software Breakpoint

To define a breakpoint at the address of the **Cmd_I** label of the sample program (1029 hex), select:

```
Breakpoints, Add
```

Enter the local symbol "Cmd_I". After the breakpoint is added, the breakpoint window becomes active and shows that the breakpoint is set.

You can add multiple breakpoints in a single command by separating each one with a semicolon. For example, you could type "1019;1021;1029" to set three breakpoints.

Run the program by selecting:

```
Processor, Go, Pc
```

The status line shows that the emulator is running the user program. Modify the command input byte to an invalid command by selecting:

```
Memory, Modify, Bytes
```

Enter an invalid command, such as "Cmd_Input=75". The following messages result:

```
ALERT:  Software breakpoint: 001029
STATUS: H8/536--Running in monitor
```

To continue program execution, select:

```
Processor, Go, Pc
```

Displaying Software Breakpoints

To view the status of the breakpoint, select:

```
Breakpoints, Display
```

The resulting display will show that the breakpoint has been cleared.

Setting a Software Breakpoint

When a breakpoint is hit, it becomes disabled. To re-enable the software breakpoint, you can select:

```
Breakpoints, Set, Single
```

The address of the breakpoint you just added is still in the address field; to set this breakpoint again, press **Enter**. As with the "**B**reakpoints **A**dd"command, the breakpoint window becomes active and shows that the breakpoint is set.

Clearing a Software Breakpoint

If you wish to clear a software breakpoint that does not get hit during program execution, you can select:

```
Breakpoints, Clear, Single
```

The address of the breakpoint set in the previous section is still in the address field; to clear this breakpoint again, press **Enter**.

Using the Analyzer

The H8/536 emulation analyzer has 48 trace signals which monitor internal emulation lines (address, data, and status lines). Optionally, you may have an additional 16 trace signals which monitor external input lines. The analyzer collects data at each pulse of a clock signal, and saves the data (a trace state) if it meets a "storage qualification" condition.

Note

Emulators which have the optional external analyzer will display the Interna/External options after the commands in the following examples. Enter **Internal** to execute the examples.

Resetting the Analysis Specification

To be sure that the analyzer is in its default or power-up state, select:

```
Analysis, Trace, Reset
```

Specifying a Simple Trigger

Suppose you wish to trace the states of the sample program which follow the read of a "B" (42 hex) command from the command input byte. To do this, you must modify the default analysis specification by selecting:

```
Analysis, Trace, Modify
```

The emulation analysis specification is shown. Use the right arrow key to move the cursor to the "Trigger on" field. Type "a" and press **Enter**.

You'll enter the pattern expression menu. Press the up arrow key until the **addr** field directly opposite the pattern **a=** is highlighted. Type the address of the command input byte, using either the global symbol **Cmd_Input** or address 0fc00, and press **Enter**.

The "Data" field is now highlighted. Type 42xx and press **Enter**. 42 is the value of the "B" command and the "x"s specify "don't care" values.

Triggering the Analyzer by Data

You may want to trigger the emulation analyzer when specific data appears on the data bus. You can accomplish this by specifying "Data" in the "Find State" field of analysis specification display.

There are some points to be noticed when you trigger the analyzer in this way. You always need to specify the "Data" with 16 bits value even when access to the data is performed by byte access. This is because the analyzer is designed so that it can capture data on internal data bus (which has 16 bits width). The following table shows the way to specify the trigger condition by data.

(DATA READ/WRITE)

Location of data	Accesses	Available <DATA> Specification
Internal ROM,RAM	Word	HHLL *1
	Byte	DDxx *2
Others		DDxx

(INSTRUCTION FETCH)

Location of data	Address	Available <DATA> Specification
Internal ROM,RAM	EVEN	HHLL *1
	ODD	xxDD *2
Others		DDxx *2

*1 HHLL means 16 bits data
*2 DD means 8 bits data

For example, to trigger the analyzer when the processor performs word access to data 1234 hex in internal ROM, you can use 1234h as the "Data" specification.

To trigger the analyzer when the processor accesses data 12 hex in external ROM, you can use 12xxh as "Data" specification.

H8/536 Analysis Status Qualifiers

Now the "Status" field is highlighted. Use the **Tab** key to view the status qualifiers which may be entered. The status qualifiers are defined as follows.

Qualifier	Status Bits (36..47)	Description
backgrnd	0xxx xxxx xxxxB	Background cycle
brelease	x111 xxxx xxxxB	Bus release cycle
byte	x110 xxxx xx1xB	Byte access
cpu	x110 xx1x xxxxB	CPU cycle
data	x110 xxxx x1xxB	Data access
dtc	x110 xx0x xxxxB	Data transfer controller cycle
exec	x101 xxxx xxxxB	Instruction execution cycle
fetch	x110 xx1x x001B	Program fetch cycle
foregrnd	1xxx xxxx xxxxB	Foreground cycle
grd	x110 0xx1 xxxxB	Guarded memory access
intack	x011 xxxx xxxxB	Interrupt acknowledge cycle
io	x110 xxx0 xxxxB	Internal I/O access
memory	x110 xxx1 xxxxB	Memory access
read	x110 xxxx xxx1B	Read cycle
word	x110 xxxx xx0xB	Word access
write	x110 xxxx xxx0B	Write cycle
wrrom	x110 x0x1 xxx0B	Write to ROM cycle

Select the **read** status and press **Enter**. Figure 2-6 and 2-7 show the resulting analysis specification. To save the new specification, use **End Enter** to exit the field in the lower right corner. You'll return to the trace specification. Press **End** to move to the trriger spec field. Press **Enter** to exit the trace specification.

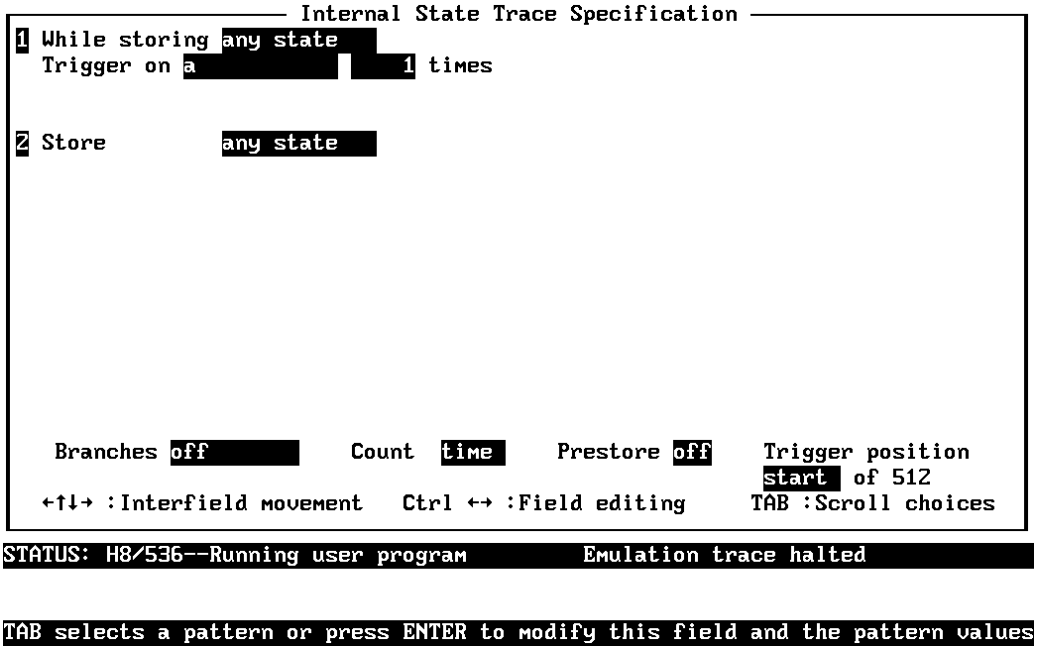


Figure 2-6. Modifying the Trace Specification

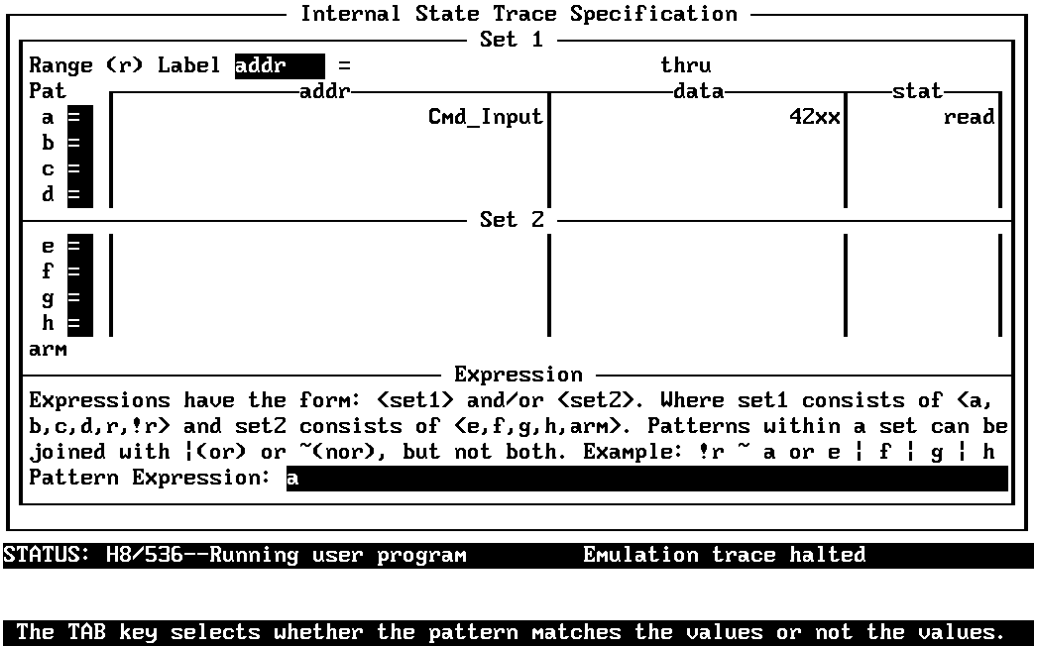


Figure 2-7. Modifying the Pattern Specification

Starting the Trace

To start the trace, select:

```
Analysis, Begin
```

A message on the status line will show you that the trace is running. You do not expect the trigger to be found because no commands have been entered. Modify the command input byte to "B" by selecting:

```
Memory, Modify, Byte
```

Enter "Cmd_Input=42". The status line now shows that the trace is complete.

Displaying the Trace

To display the trace, select:

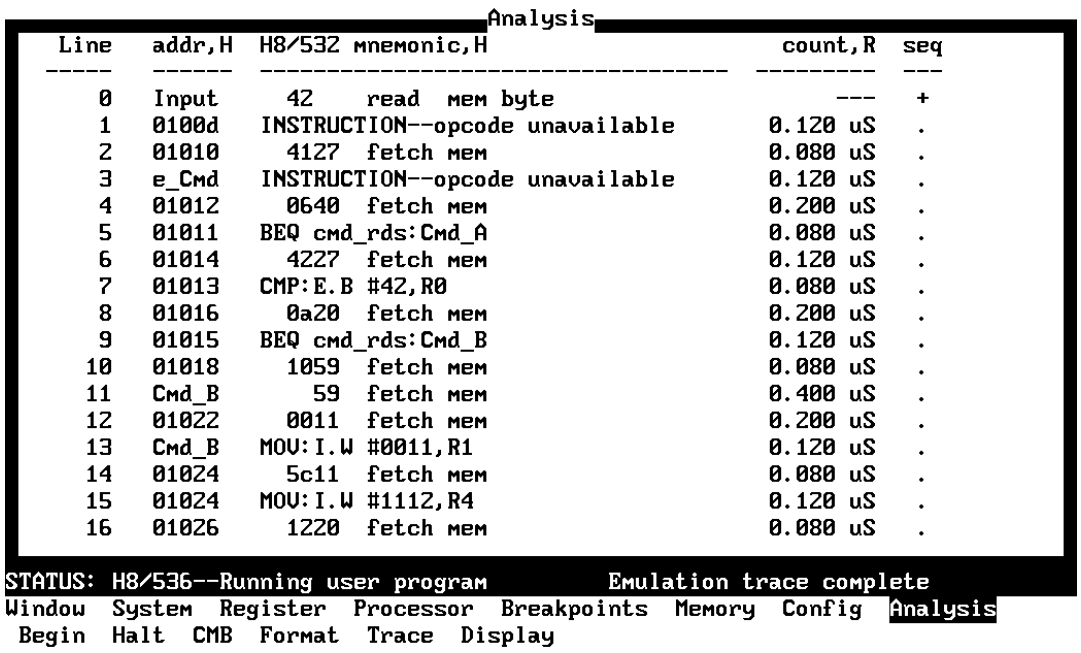
```
Analysis, Display
```

You are now given two fields in which to specify the states to display. Use the right arrow key to move the cursor to the "Ending state to display" field. Type "60" into the ending state field, press **Enter**, and use <CTRL>**z** to zoom the trace window.

Note

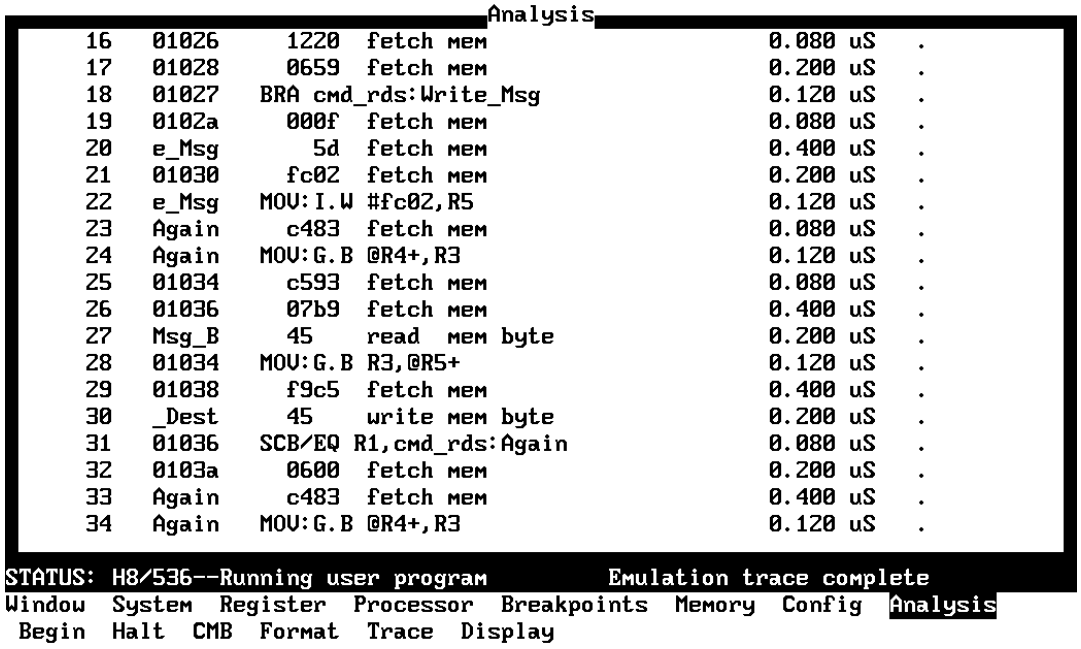
If you choose to dump a complete trace into the trace buffer, it will take a few minutes to display the trace.

Use the **Home** key to get the top of the trace. The resulting trace is similar to the trace shown in the following display.



Line 0 in the trace list above shows the state which triggered the analyzer. The trigger state is always on line 0. The other states show the exit from the **Scan** loop and the **Exe_Cmd** instructions.

To list the next lines of the trace, press the **PgDn** or **Next** key.



The resulting display shows the **Cmd_B** instructions and the branch to **Write_Msg** and the beginning of the instructions which move the "THIS IS MESSAGE B" message to the destination locations.

For a Complete Description

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

Using a Command File

You can use a command file to perform many functions for you, without having to manually type each function. For example, you might want to create a command file that modifies configuration, maps memory and loads program into memory for the sample program. To create such a command file:

```
System, Log, Input, Enable
```

Enter command file name "cmd_rds.cmd", and press **Enter**. This sets up a file to record all commands you execute. The commands will be logged to the file cmd_rds.cmd in the current directory. You can then use this file as a command file to execute these commands automatically.

First, to set up the reset value for the stack pointer:

```
Config, General
```

Use the arrow keys to move the cursor to the "Reset value for Stack Pointer" field, type 0fc80, and press **End** and **Enter**.

To map the memory:

```
Config, Map, Memory
```

Map 1000 hex through 1fff hex to **erom** and fc00 hex through fcff hex to **eram**. (As shown in Figure 2-5.)

To load the program into memory:

```
Memory, Load
```

Enter file format, memory type, and absolute file name, and press **Enter**.

Now we're finished logging commands to the file. To disable logging:

```
System, Log, Input, Disable
```
The command file cmd_rds.cmd will no longer accept command input.

Let’s execute the command file "cmd_rds.cmd".

```
System, Command_file
```
Enter "cmd_rds.cmd", press **Enter**. As you can see, the sequence of commands you entered is automatically executed.

Resetting the Emulator

To reset the emulator, select:

```
Processor, Reset, Hold
```
The emulator is held in a reset state (suspended) until a "**P**rocessor **B**reak", "**P**rocessor **G**o", or "**P**rocessor **S**tep" command is entered. A CMB execute signal will also cause the emulator to run if reset.

You can also specify that the emulator begin executing in the monitor after reset instead of remaining in the suspended state. To do this, select:

```
Processor, Reset, Monitor
```

Exiting the PC Interface

There are two different ways to exit the PC Interface. You can exit the PC Interface using the "locked" option which specifies that the current configuration will be present next time you start up the PC Interface. You can select this option as follows.

```
System, Exit, Locked
```

Symbols are lost when you use the "System Exit Locked" command; however, you can reload them (after you reenter the PC Interface) with the "System Symbols Global Load" command.

The other way to execute the PC Interface is with the "unlocked" option which specifies that the default configuration will be present the next time you start up the PC Interface. You can select this option with the following command.

```
System, Exit, Unlocked
```

Notes

3

In-Circuit Emulation

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

This chapter will:

- Describe the issues concerning the installation of the emulator probe into target systems.
- Show you how to install the emulator probe.

We will cover the first topic in this chapter. For complete details on in-circuit emulation configuration, refer to the "Configuring the Emulator" chapter.

Prerequisites

Before performing the tasks described in this chapter, you should be familiar with how the emulator operates in general. Refer to the *HP 64700 Emulators: System Overview* manual and the "Getting Started" chapter of this manual.

Installing the Target System Probe

The emulator probe has a PLCC connector. The emulator is shipped with a pin guard over the target system probe. This guard is designed to prevent impact damage to the pins and should be left in place while you are not using the emulator.

Caution

DAMAGE TO THE EMULATOR CIRCUITRY MAY RESULT IF THESE PRECAUTIONS ARE NOT OBSERVED. The following precautions should be taken while using the H8/536 emulator.

Power Down Target System. Turn off power to the user target system and to the H8/536 emulator before inserting the user plug to avoid circuit damage resulting from voltage transients or mis-insertion of the user plug.

Verify User Plug Orientation. Make certain that Pin 1 of the target system microprocessor socket and Pin 1 of the user plug are properly aligned before inserting the user plug in the socket. Failure to do so may result in damage to the emulator circuitry.

Protect Against Static Discharge. The H8/536 emulator contains devices which are susceptible to damage by static discharge. Therefore, operators should take precautionary measures before handling the user plug to avoid emulator damage.

Protect Target System CMOS Components. If your target system includes any CMOS components, turn on the target system first, then turn on the H8/536 emulator; when powering down, turn off the emulator first, then turn off power to the target system.

Installing into a PLCC Type Socket

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

1. Remove the H8/536 microprocessor from the target system socket (PLCC socket). Note the location of pin 1 on the processor and on the target system socket.

2. Store the microprocessor in a protected environment (such as antistatic foam).

3. Install the target system probe into the target system microprocessor socket.

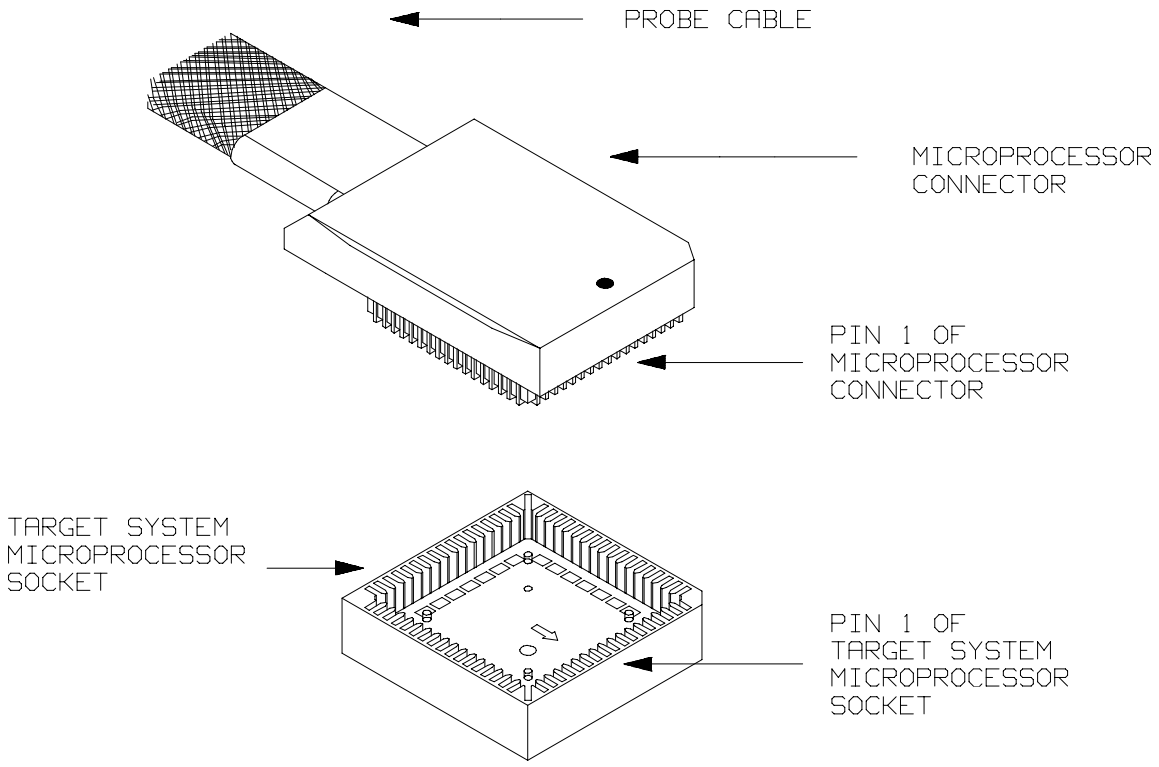


Figure 3-8. Installing into a PLCC type socket

Note

To make sure the contact between emulator probe and target system microprocessor socket, we recommend that you use

ITT CANNON "**LCS-84**" series 84 pin PLCC socket.

Running the Emulator from Target Reset

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

At First, you must specify the emulator responds to /RES signal by the target system (see the "Enable /RES input from Target" configuration in Chapter 4 of this manual).

To specify a run from reset state, select:

`Processor, Go, Reset`

The status now shows that the emulator is "Awaiting target reset".

After the target system is reset, the status line message will change to show the appropriate emulator status.

4

Configuring the Emulator

Introduction

The H8/536 emulator can be used in all stages of target system development. For instance, you can run the emulator out-of-circuit when developing target system software, or you can use the emulator in-circuit when integrating software with target system hardware. Emulation memory can be used in place of, or along with, target system memory. You can use the emulator's internal clock or the target system clock. You can execute target programs in real-time or allow emulator execution to be diverted into the monitor when commands request access of target system resources (target system memory, register contents, etc.)

The emulator is a flexible instrument and it may be configured to suit your needs at any stage of the development process. This chapter describes the options available when configuring the H8/536 emulator.

This chapter will:

- Show you how to access the emulator configuration options.
- Describe the emulator configuration options.
- Show you how to save a particular emulator configuration, and load it again at a later time.

Accessing the Emulator Configuration Options

To enter the general configuration menu, Select:

```
Config, General
```

The general configuration menu appers as follows:

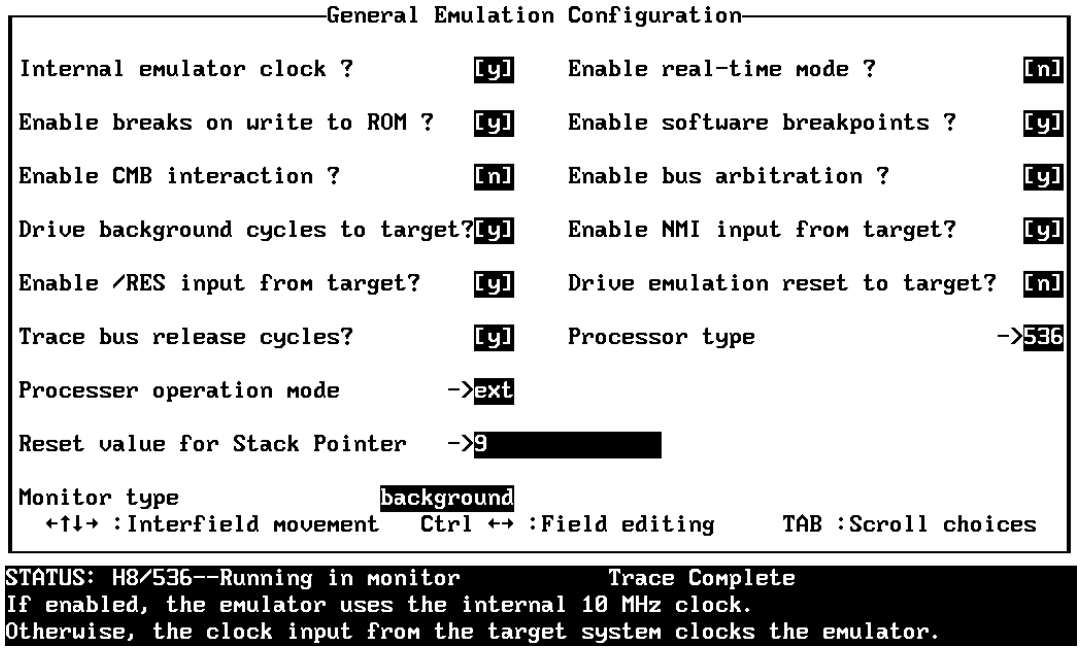


Figure 4-9. H8/536 General Emulator Configuration

When you position the cursor to a configuration item, a brief description of the item appears at the bottom of the display.

Note

It is possible to use the System Terminal window to modify the emulator configuration. However, if you do this, some PC Interface features may no longer work properly. We recommend that you only modify the emulator configuration by using the options presented in the PC Interface.

Internal Emulator Clock?

This configuration question allows you to select the emulator's clock source; you can choose either the internal clock source or the target system clock source. The default emulator configuration selects the internal clock.

yes Selects the internal clock oscillator as the emulator clock source. The emulators' internal clock speed is 10 MHz (system clock).

no Selects the clock input to the emulator probe from the target system. You must use a clock input conforming to the specifications for the H8/536 microprocessor.

You should always select the external clock option when using the emulator in-circuit to ensure that the emulator is properly synchronized with your target system.

Note 

Changing the clock source drives the emulator into the reset state.

Enable Real-Time Mode?

If it is important that the emulator execute target system programs in real-time, you can enable the real-time emulator mode. In other words, when you execute target programs (with the "**P**rocessor, **G**o" command), the emulator will execute in real-time.

No The default emulator configuration disables the real-time mode. When the emulator is executing the target program, you are allowed to enter emulation commands that require access to target system resources (display/modify: registers or target system memory). If one of these commands is entered, the system controller will temporarily break emulator execution into the monitor.

Yes If your target system program requires real-time execution, you should enable the real-time mode in order to prevent temporary breaks that might cause target system problems.

Commands Not Allowed when Real-Time Mode is Enabled

When emulator execution is restricted to real-time and the emulator is running user code, the system refuses all commands that require access to processor registers or target system memory. The following commands are not allowed when runs are restricted to real-time:

- Register display/modification.
- Target system memory display/modification.
- Internal I/O registers display/modification.

If the real-time mode is enabled, these resources can only be displayed or modified while running in the monitor.

Breaking out of Real-Time Execution

The only commands which are allowed to break real-time execution are:

```
Processor, Reset
Processor, Go
Processor, Break
Processor, Step
```

Enable Breaks on Writes to ROM?

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

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

no The emulator will not break to the monitor upon a write to ROM. The emulator will not modify the memory location if it is in emulation ROM.

Note

The **wrrom** analysis specification status option allows you to use "write to ROM" cycles as trigger and storage qualifiers.

Enable Software Breakpoints?

When you define or enable a software breakpoint to a specified address, the emulator will replace the opcode with one of H8/536 undefined opcode (1B hex) as breakpoint interrupt instruction. When the emulator detects the breakpoint interrupt instruction (1B hex), user program breaks to the monitor, and the original opcode will be replaced at the software breakpoint address. A subsequent run or step command will execute from this address.

Refer to the "Getting Started" for information on using software breakpoints.

No The software breakpoints feature is disabled. This is specified by the default emulator configuration, so you must change this configuration item before you can use software breakpoints.

Yes The software breakpoints feature is enabled. The emulator detects the breakpoint interrupt instruction (1B hex), it generates a break to background request which as with the "processor break" command. Since the system controller knows the locations of defined software breakpoints, it can determine whether the breakpoint interrupt instruction (1B hex) is a software breakpoint or opcode in your target program.

When you define (add) a breakpoint, software breakpoints are automatically enabled.

Enable CMB Interaction?

Coordinated measurements are measurements synchronously made in multiple emulators or analyzers. Coordinated measurements can be made between HP 64700 Series emulators which communicate over the Coordinated Measurement Bus (CMB).

Multiple emulator start/stop is one type of coordinated measurement. The CMB signals READY and /EXECUTE are used to perform multiple emulator start/stop.

This configuration item allows you to enable/disable interaction over the READY and /EXECUTE signals. (The third CMB signal, TRIGGER, is unaffected by this configuration item.)

No The emulator ignores the /EXECUTE and READY lines, and the READY line is not driven.

Yes Multiple emulator start/stop is enabled. If the

```
Processor, CMB, Go, ...
```

command is entered, the emulator will start executing code when a pulse on the /EXECUTE line is received. The READY line is driven false while the emulator is running in the monitor; it goes true whenever execution switches to the user program.

Note

CMB interaction will also be enabled when the

```
Processor, CMB, Execute
```

command is entered.

Enable Bus Arbitration?

The bus arbitration configuration question defines how your emulator responds to bus request signals from the target system during foreground operation. The /BREQ signal from the target system is always ignored when the emulator is running the background monitor.

yes When bus arbitration is enabled, the /BREQ (bus request) signal from the target system is responded to exactly as it would be if only the emulation processor was present without an emulator. In other words, if the emulation processor receives a /BREQ from the target system, it will respond by asserting /BACK and will set the various processor lines to tri-state. /BREQ is then released by the target; /BACK is negated by the processor, and the emulation processor restarts execution.

Note

DMA (direct memory access) devices is prohibited from accessing to emulation memory.

no When you disable bus arbitration, the emulator ignores the /BREQ signal from the target system. The emulation processor will never drive the /BACK line true; nor will it place the address, data and control signals into the tri-state mode.

Enabling and disabling bus master arbitration can be useful to you in isolating target system problems. For example, you may have a situation where the processor never seems to execute any code. You can disable bus arbitration to check and see if faulty arbitration circuitry in your target system is contributing to the problem.

Drive Background Cycles to Target?

This question allows you specify whether or not the emulator will drive the target system bus on background cycles.
If you have selected to use a foreground monitor, emulator foreground monitor cycles will appear at the target interface exactly as if they were bus cycles caused by any target system program.

yes Specifies that background cycles are driven to the target system. Emulation processor's address and control strobes (except /WR) are driven during background cycles. Background write cycles won't appear to the target system.

no Background monitor cycles are not driven to the target system. When you select this option, the emulator will appear to the target system as if it is between bus cycles while it is operating in the background monitor.

Note

Changing this configuration drives the emulator into the reset state.

Enable NMI Input from Target?

This configuration allows you to specify whether or not the emulator responds to NMI(non-maskable interrupt request) signal from the target system during foreground operation.

yes	The emulator will respond to the NMI request from the target system.
no	The emulator will not respond to the NMI request from the target system.

If you are using the background monitor, the emulator does not accept any interrupt during background execution. All edge-sensed interrupts (include NMI) are latched last one during in background, and such interrupts will occur when context is changed to foreground. All level-sensed interrupts and internal interrupts are ignored during in background operation.

Note 

Changing this configuration drives the emulator into the reset state.

Enable /RES Input from Target?

This configuration allows you to specify whether or not the emulator responds to /RES and /STBY signals by the target system during foreground operation.

While running the background monitor, the emulator ignores /RES and /STBY signals except that the emulator's status is "Awaiting target reset".

yes	The emulator will respond to /RES and /STBY input during foreground operation.
no	The emulator will not respond to /RES and /STBY input from the target system.

Note

Changing this configuration drives the emulator into the reset state.

Note

If you specify that the emulator will drive the /RES signal to the target system during emulation reset or by the overflow of Watch Dog Timer, the emulator should be configured to respond to the /RES input to the target system.

Drive Emulation Reset to Target?

This configuration allows you to select whether or not the emulator will drive the /RES signal to the target system during emulation reset.

no Specifies that the emulator will not drive the /RES signal during emulation reset.

yes The emulator will drive an active level on the /RES signal to the target system during emulation reset.

This configuration option is meaningful only when the emulator is configured to respond to the /RES input to the target system
Refer to the "Enable /RES Input from Target?" configuration in this chapter.

Trace Bus Release Cycles?

You can direct the emulator to send bus release cycle data to emulation analyzer or not to send it.

yes When you enable tracing bus release cycles, bus release cycles will appear as one analysis trace line.

no Bus release cycles will not appear on analysis trace list (display).

Processor type

The configuration allows you to select processor to be emulated.

536	When you are going to emulate H8/536 microprocessor, select this item.
534	When you are going to emulate H8/534 microprocessor, select this item.

Note

Changing this configuration drives the emulator into the reset state.

Processor Operation Mode

This configuration defines operation mode in which the emulator works.

ext	The emulator will work using the mode setting by the target system. The target system must supply appropriate input to MD0, MD1 and MD2. If you are using the emulator out of circuit when "external" is selected, the emulator will operate in mode 7.
<mode_num>	When <mode_num> is selected, the emulator will operate in selected mode regardless of the mode setting by the target system.

Valid <mode_num> are following:

<mode_num>	**Description**
1	The emulator will operate in mode 1. (expanded minimum mode)

2	The emulator will operate in mode 2. (expanded minimum mode with internal ROM)
3	The emulator will operate in mode 3. (expanded maximum mode)
4	The emulator will operate in mode 4. (expanded maximum mode with internal ROM)
7	The emulator will operate in mode 7. (single chip mode)

Note  Changing this configuration drives the emulator into the reset state.

Monitor Type

bg	Specify monitor type as **background monitor**. When you select background monitor, configuration question of foreground monitor address have no effect to emulator's operation.
fg	Specify monitor type as **foreground monitor**. When you select foreground monitor, you must specify correct foreground monitor start address with next configuration question (foreground monitor address). After you completed the configuration setting, you need to load foreground monitor program to the emulator with "Memory Load" feature. The foreground monitor program must already assembled and linked with appropriate monitor start address specification.

To use the foreground monitor, follow below steps.

1. Decide which location the foreground monitor should be loaded.

2. Assemble and link the foreground monitor program along with the location you decided.

3. Configure the emulator as described in this chapter. (monitor type selection and monitor location).

4. Load the foreground monitor program into memory by "**M**emory **L**oad" feature.

Note 

Changing this configuration drives the emulator into the reset state.

Foreground Monitor Address

You must specify foreground monitor start address when you select **"fg"** by above configuration question "Monitor type". This address specification must be same with the address specification when you assemble the foreground monitor program.

The address must be specified in a 20-bit hexadecimal address, and must be located on a 2K byte boundary other than 0 hex.

Reset Value for Stack Pointer?

This question allows you to specify the value to which the stack pointer (SP) and the stack page register (TP) will be set on entrance to the emulation monitor initiated RESET state (the "Emulation reset" status).

The address specified in response to this question must be a 20-bit hexadecimal even address.

You **cannot** set this address at the following location.

- Odd address
- Internal I/O register address

When you are using the foreground monitor, this address should be defined in an emulation or target system RAM area which is not used by user program.

Note

We recommend that you use this method of configuring the stack pointer and the stack page register. Without a stack pointer and a stack page register, the emulator is unable to make the transition to the run state, step, or perform many other emulation functions. However, using this option **does not** preclude you from changing the stack pointer value or location within your program; it just sets the initial conditions to allow a run to begin.

Storing an Emulator Configuration

The PC Interface lets you store a particular emulator configuration so that it may be re-loaded later. The following information is saved in the emulator configuration.

- Emulator configuration items.
- Memory map.
- Break conditions.
- Trigger configuration.
- Window specifications.

To store the current emulator configuration, select:

`Config, Store`

Enter the name of file to which the emulator configuration will be saved.

Loading an Emulator Configuration

If you have previously stored an emulator configuration and wish to re-load it into the emulator, select:

`Config, Load`

Enter the configuration file name and press **Enter**. The emulator will be re-configured with the values specified in the configuration file.

Notes

5

Using the Emulator

Introduction

In the "Getting Started" chapter, you learned how to load code into the emulator, how to modify memory and view a register, and how to perform a simple analyzer measurement. In this chapter, we will discuss in more detail other features of the emulator.

This chapter shows you how to:

- Making Coordinated Measurements.
- Store the contents of memory into absolute files.

This chapter also discusses:

- Display or Modify registers.

Making Coordinated Measurements

Coordinated measurements are measurements synchronously made in multiple emulators or analyzers. Coordinated measurements can be made between HP 64700 Series emulators which communicate over the Coordinated Measurement Bus (CMB). Coordinated measurements can also be made between an emulator and some other instrument connected to the BNC connector.

This section will describe coordinated measurements made from the PC Interface which involve the emulator. These types of coordinated measurements are:

- Running the emulator on reception of the CMB /EXECUTE signal.
- Using the analyzer trigger to break emulator execution into the monitor.

Note

You must use the background emulation monitor to perform coordinated measurements.

Three signal lines on the CMB are active and serve the following functions when enabled:

/TRIGGER Active low. The analyzer trigger line on the CMB and on the BNC serve the same logical purpose. They provide a means for the analyzer to drive its trigger signal out of the system or for external trigger signals to arm the analyzer or break the emulator into its monitor.

READY Active high. This line is for synchronized, multi-emulator start and stop. When CMB run control interaction is enabled, all emulators are required to break to background upon reception of a false READY signal and will not return to foreground until this line is known to be in a true state.

/EXECUTE Active low. This line serves as a global interrupt signal. Upon reception of an enabled /EXECUTE signal, each emulator is to interrupt whatever it is doing and execute a previously defined process, typically, run the emulator or start a trace measurement.

Running the Emulator at /EXECUTE

Before you can specify that the emulator run upon receipt of the /EXECUTE signal, you must enable CMB interaction. To do this, select:

```
Config, General
```

Use the arrow keys to move the cursor to the "Enable CMB Interaction? [n]" question, and type "y". Use the **Enter** key to exit out of the lower right-hand field in the configuration display.

To specify that the emulator begin executing a program upon reception of the /EXECUTE signal, select:

```
Processor, CMB, Go
```

At this point you may either select the current program counter, or you may select a specific address.

The command you enter is saved and is executed when the /EXECUTE signal becomes active. Also, you will see the message "ALERT: CMB execute; run started".

Breaking on the Analyzer Trigger

To cause emulator execution to break into the monitor when the analyzer trigger condition is found, you must modify the trigger configuration. To access the trigger configuration, select:

```
Config, Trigger
```

The trigger configuration display contains two diagrams, one for each of the internal TRIG1 and TRIG2 signals.

To use the internal TRIG1 signal to connect the analyzer trigger to the emulator break line, move the cursor to the highlighted "Analyzer" field in the TRIG1 portion of the display, and use the **Tab** key to select the "----->>" arrow which shows that the analyzer is driving TRIG1. Next, move the cursor to the highlighted "Emulator" field and use the **Tab** key to select the arrow pointing towards the emulator (<<-----); this specifies that emulator execution will break into the monitor when the TRIG1 signal is driven. The trigger configuration display is shown in figure 5-1.

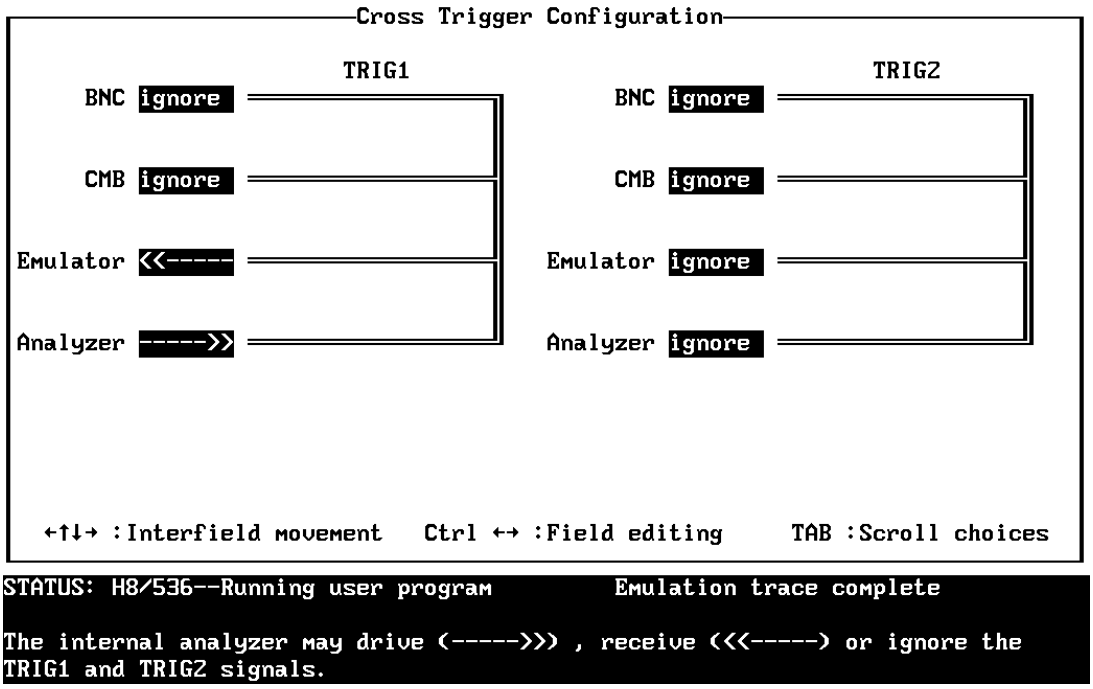


Figure 5-1. Cross Trigger Configuration

Note

If your emulator is configured with external analyzer, "Timing" cross trigger options are displayed.

Storing Memory Contents to an Absolute File

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

```
Memory, Store
```

Note

The first character of the absolute file name must be a letter. You can name the absolute file with a total of 8 alphanumeric characters, and optionally, you can include an extension of up to 3 alphanumeric characters.

Caution

The "Memory Store" command writes over an existing file if it has the same name that is specified with the command. You may wish to verify beforehand that the specified filename does not already exist.

Accessing Target System with E clock synchronous instruction

You can access target system devices in synchronization with the E clock. To do this, use the following commands:

```
Processor, IO, Display
Processor, IO, Modify
```

The emulator will access the device using the MOVFPE/MOVTPE instruction.

Register Names and Classes

The following register names and classes may be used with "**R**egister **D**isplay/**M**odify" commands.

Summary

H8/536 register designators. All available register class names and register names are listed below.

* (Basic) Class

Register name	Description
pc	Program counter
cp	Code page register
sr	Status register
dp	Data page register
ep	Extended page register
tp	Stack page register
br	Base register
r0	Register R0
r1	Register R1
r2	Register R2
r3	Register R3
r4	Register R4
r5	Register R5
r6	Register R6
r7	Register R6
r7	Register R7
fp	Frame pointer
sp	Stack pointer
mdcr	Mode control register

sys Class

System control registers

Register name	Description
wcr	Wait control register
ramcr	RAM control register
mdcr	Mode control register
sbycr	Software stand-by control register

intc Class

Interrupt control registers

ipra	Interrupt priority register A
iprab	Interrupt priority register B
iprc	Interrupt priority register C
iprd	Interrupt priority register D
ipre	Interrupt priority register E
iprf	Interrupt priority register F

dtc Class

Data transfer controller registers

dtea	DT enable register A
dteb	DT enable register B
dtec	DT enable register C
dted	DT enable register D
dtee	DT enable register E
dtef	DT enable register F

port Class

I/O port registers

Register name	Description
p1ddr	Port 1 data direction register
p2ddr	Port 2 data direction register
p3ddr	Port 3 data direction register
p4ddr	Port 4 data direction register
p5ddr	Port 5 data direction register
p6ddr	Port 6 data direction register
p7ddr	Port 7 data direction register
p9ddr	Port 9 data direction register
p1dr	Port 1 data register
p2dr	Port 2 data register
p3dr	Port 3 data register
p4dr	Port 4 data register
p5dr	Port 5 data register
p6dr	Port 6 data register
p7dr	Port 7 data register
p8dr	Port 8 data register
p9dr	Port 9 data register
p1cr	Port 1 control register
p69cr	Port 69 control register

frt1 Class

Free running timer 1 registers

frtcr1	Timer control register
frtcsr1	Timer control/status register
frc1	Free running counter
ocra1	Output compare register A
ocrb1	Output compare register B
icr1	Input capture register

frt2 Class

Free running timer 2 registers

Register name	Description
frtcr2	Timer control register
frtcsr2	Timer control/status register
frc2	Free running counter
ocra2	Output compare register A
ocrb2	Output compare register B
icr2	Input capture register

frt3 Class

Free running timer 3 registers

frtcr3	Timer control register
frtcsr3	Timer control/status register
frc3	Free running counter
ocra3	Output compare register A
ocrb3	Output compare register B
icr3	Input capture register

tmr Class

Timer registers

tcr	Timer control register
tcsr	Timer control/status register
tcora	Timer constant register A
tcorb	Timer constant register B
tcnt	Timer counter

pwm1 Class

PWM timer1 registers

pwmtcr1	Timer control register
dtr1	Duty register
pwmtcnt1	Timer counter

pwm2 Class

PWM timer2 registers

Register name	Description
pwmtcr2	Timer control register
dtr2	Duty register
pwmtcnt2	Timer counter

pwm3 Class

PWM timer3 registers

pwmtcr3	Timer control register
dtr3	Duty register
pwmtcnt3	Timer counter

wdt Class

Watchdog timer registers

wdtcsr	Timer control/status register
wdtcnt	Timer counter
rstcsr	Reset control/status register

sci1 Class

Serial communication interface 1 registers.

rdr1	Receive data register
tdr1	Transmit data register
smr1	Serial mode register
scr1	Serial control register
ssr1	Serial status register
brr1	Bit rate register

sci2 Class

Serial communication interface 2 registers.

Register name	Description
rdr2	Receive data register
tdr2	Transmit data register
smr2	Serial mode register
scr2	Serial control register
ssr2	Serial status register
brr2	Bit rate register

adc Class

A/D converter registers

Register name	Description
addra	A/D data register A
addrb	A/D data register B
addrc	A/D data register D
addrd	A/D data register D
adcsr	A/D control/status register
adcr	A/D control register

Notes

A

File Format Readers

Using the HP 64000 Reader

An HP 64000 "reader" is provided with the PC Interface. The HP 64000 Reader converts the files into two files that are usable with your emulator. This means that you can use available language tools to create HP 64000 absolute files, then load those files into the emulator using the PC Interface.

The HP 64000 Reader can operate from within the PC Interface or as a separate process. When operating the HP 64000 Reader, it may be necessary to execute it as a separate process if there is not enough memory on your personal computer to operate the PC Interface and HP 64000 Reader simultaneously. You can also operate the reader as part of a "make file."

What the Reader Accomplishes

Using the HP 64000 files (<file.X>, <file.L>, <scr1.A>, <scr2.A>, ...) the HP 64000 Reader will produce two new files, an "absolute" file and an ASCII symbol file, that will be used by the PC Interface. These new files are named: "<file>.hpa" and "<file>.hps."

The Absolute File

During execution of the HP 64000 Reader, an absolute file (<file>.hpa) is created. This absolute file is a binary memory image which is optimized for efficient downloading into the emulator.

The ASCII Symbol File

The ASCII symbol file (<file>.hps) produced by the HP 64000 Reader contains global symbols, module names, local symbols, and, when using applicable development tools such as a "C" compiler, program

line numbers. Local symbols evaluate to a fixed (static, not stack relative) address.

Note 

You must use the required options for your specific language tools to include symbolic ("debug") information in the HP 64000 symbol files. The HP 64000 Reader will only convert symbol information present in the HP 64000 symbol files (<file.L>, <src1.A>, <src2.A>, ...).

The symbol file contains symbol and address information in the following form:

```
 module_name1
 module_name2
 ...
 module_nameN
global_symbol1  address
global_symbol2  address
...
global_symbolN  address
|module_name1|# 1234          address
|module_name1|local_symbol1   address
|module_name1|local_symbol2   address
...
|module_name1|local_symbolN   address
```

Each of the symbols is sorted alphabetically in the order: module names, global symbols, and local symbols.

Line numbers will appear similar to a local symbol except that "local_symbolX" will be replaced by "#NNNNN" where NNNNN is a five digit decimal line number. The addresses associated with global and local symbols are specific to the processor for which the HP 64000 files were generated.

Note 

When the line number symbol is displayed in the emulator, it appears in brackets. Therefore, the symbol "MODNAME: line 345" will be displayed as "MODNAME:[345]" in mnemonic memory and trace list displays.

The space preceding module names is required. Although formatted for readability here, a single tab separates symbol and address.

The local symbols are scoped. This means that to access a variable named “count” in a source file module named “main.c,” you would enter “main.c:count” as shown below.

Module Name	Variable Name	You Enter:
main.c	count	main.c:count
main.c	line number 23	main.c: line 23

You access line number symbols by entering the following on one line in the order shown:

module name
colon (:)
space
the word “line”
space
the decimal line number

For example:

```
main.c: line 23
```

Location of the HP 64000 Reader Program

The HP 64000 Reader is located in the directory named \hp64700\bin by default, along with the PC Interface. This directory must be in the environment variable PATH for the HP 64000 Reader and PC Interface to operate properly. The PATH is usually defined in the “\autoexec.bat” file.

The following examples assume that you have “\hp64000\bin” included in your PATH variable. If not, you must supply the directory name when executing the Reader program.

Using the Reader from MS-DOS

The command name for the HP 64000 Reader is **RHP64000.EXE**. To execute the Reader from the command line, for example, enter:

```
RHP64000 [-q] <filename>
```

-q — This option specifies the "quiet" mode, and suppresses the display of messages.

<filename> — This represents the name of the HP 64000 linker symbol file (file.L) for the absolute file to be loaded.

The following command will create the files "TESTPROG.HPA" and "TESTPROG.HPS"

```
RHP64000 TESTPROG.L
```

Using the Reader from the PC Interface

The PC Interface has a file format option under the "**M**emory **L**oad" command. After you select HP64000 as the file format, the HP 64000 Reader will operate on the file you specify. After this completes successfully, the PC Interface will accept the absolute and symbol files produced by the Reader.

To use the Reader from the PC Interface:

1. Start up the PC Interface.
2. Select "**M**emory **L**oad." The memory load menu will appear.
3. Specify the file format as "HP64000." This will appear as the default file format.
4. Specify the name of an HP 64000 linker symbol file (TESTFILE.L for example).

Using the HP 64000 file that you specify (TESTFILE.L, for example), the PC Interface performs the following:

- It checks to see if two files with the same base name and extensions .HPS and .HPA already exist (for example, TESTFILE.HPS and TESTFILE.HPA).

- If TESTFILE.HPS and TESTFILE.HPA don't exist, the HP 64000 Reader produces them. The new absolute file, TESTFILE.HPA, is then loaded into the emulator.
- If TESTFILE.HPS and TESTFILE.HPA already exist but the create dates and times are earlier than the HP 64000 linker symbol file creation date/time, the HP 64000 Reader recreates them. The new absolute file, TESTFILE.HPA, is then loaded into the emulator.
- If TESTFILE.HPS and TESTFILE.HPA already exist but the dates and times are later than the creation date and time for the HP 64000 linker symbol file, the HP 64000 Reader will not recreate TESTFILE.HPA. The current absolute file, TESTFILE.HPA, is then loaded into the emulator.

Note

Date/time checking is only done within the PC Interface.

When running the HP 64000 Reader at the MS-DOS command line prompt, the HP 64000 Reader will always update the absolute and symbol files.

When the HP 64000 Reader operates on a file, a status message will be displayed indicating that it is reading an HP 64000 file. When the HP 64000 Reader completes its processing, another message will be displayed indicating the absolute file is being loaded.

If the Reader Won't Run

If your program is very large, the PC Interface may run out of memory while attempting to create the database file. If this occurs, you will need to exit the PC Interface and execute the program at the MS-DOS command prompt to create the files that are downloaded to the emulator.

Including RHP64000 in a Make File

You may wish to incorporate the “RHP64000” process as the last step in your “make file,” as a step in your construction process, to eliminate the possibility of having to exit the PC Interface due to space limitations describe above. If the files with “.HPA” and “.HPS” extensions are not current, loading an HP 64000 file will automatically create them.

Using the HP 64869 Reader

A HP 64869 format "reader" is provided with the PC Interface. The HP 64869 Reader converts a HP 64869 format file into two files that are usable with the HP 64739 emulator. This means you can use available language tools to create HP 64869 format absolute files, then load those files into the emulator using the H8/536 PC Interface.

The HP 64869 Reader can operate from within the PC Interface or as a separate process. Operation from within the PC Interface is available if there is enough memory on your personal computer to run the PC Interface and HP 64869 Reader simultaneously.

You can also run the reader as part of a "make file."

What the Reader Accomplishes

Using any HP 64869 format absolute file in the form "<file>.<ext>", the HP 64869 Reader will produce two new files, an "absolute" file and an ASCII symbol file, that will be used by the H8/536 PC Interface.

The Absolute File

During execution of the HP 64869 Reader, an absolute file (<file>.HPA) is created. This absolute file is a binary memory image which is optimized for efficient downloading into the emulator.

The ASCII Symbol File

The ASCII symbol file (<file>.HPS) produced by the HP 64869 Reader contains global symbols, module names, local symbols, and, when using applicable development tools like a "C" compiler, program line

numbers. Local symbols evaluate to a fixed (static, not stack relative) address.

Note 

You must use the required options for you specific language tools to include symbolic ("debug") information in the HP 64869 format absolute file.

The symbol file contains symbol and address information in the following form:

```
 module_name1
 module_name2
 ...
 module_nameN
global_symbol1                    address
global_symbol2                    address
...
global_symbolN                    address
|module_name|local_symbol1        address
|module_name|local_symbol2        address
...
|module_name|local_symbolN        address
|module_name|# 1234               address
```

Each of the symbols is sorted alphabetically in the order: module names, global symbols, and local symbols.

Line numbers will appear similar to a local symbol except that "local_symbolX" will be replaced by "#NNNNN" where NNNNN is a five digit decimal line number. Line numbers should appear in ascending order in both the line number itself and its associated address.

Note 

When the line number symbol is displayed in the emulator, it appears as a bracketed number. Therefore, the symbol "modname:# 345" will be displayed as "modname:[345]" in mnemonic memory and trace list displays

---

The space preceding module names is required. Although formatted for readability here, a single tab separates symbol and address.

The local symbols are scoped. When accessing the variable named "count" in the source file module named "main.c", you would enter "main:count". Notice that the module name of the source file "main.c" is "main". see the following table.

Module Name	Variable Name	You Enter:
main	count	main:count
main	line number 23	main: line 23

Location of the HP 64869 Reader Program

The HP 64869 Reader is located in the directory named \hp64700\bin by default, along with the PC Interface. This directory must be in the environment variable PATH for the HP 64869 Reader and PC Interface to operate properly. This is usually defined in "\autoexec.bat" file.

Using the HP 64869 Reader from MS-DOS

The command name for the HP 64869 Reader is **RD64869.EXE**. You can execute the HP 64869 Reader from the command line with the command:

```
C:\HP64700\BIN\RD64869 [-q] <filename>
<RETURN>
```

where:

[-q] specifies the "quiet" mode. This option suppresses the display of messages.

<filename> is the name of the file containing the HP 64869 format absolute program.

The command

```
C:\HP64700\BIN\RD64869 TESTPROG.ABS
```

will therefore create the files "TESTPROG.HPA" and "TESTPROG.HPS".

Using the HP 64869 Reader from the PC Interface

The H8/536 PC Interface has a file format option under the "**M**emory, **L**oad" command. After you select this option, the HP 64869 Reader will operate on the file you specify. After this completes successfully, the H8/536 PC Interface will accept the absolute and symbol files produced by the Reader.

To use the Reader from the PC Interface, follow these steps:

1. Start up the H8/536 PC Interface.
2. Select "**M**emory, **L**oad". The memory load menu will appear.
3. Specify the file format as "HP64869". This will appear as the default file format.
4. Specify a file in HP 64869 format ("TESTFILE.ABS", for example,). The file extension can be something other than ".ABS", but cannot be ".HPA", ".HPT", or ".HPS".

Note

The "<filename>.HPT" file is a temporary file used by the HP 64869 Reader to process the symbols.

Using the HP 64869 format file that you specify (TESTFILE.ABS, for example), the PC Interface performs the following:

- Checks to see if two files with the same base name and extensions .HPS and .HPA already exist (for example, TESTFILE.HPS and TESTFILE.HPA).
- If TESTFILE.HPS and TESTFILE.HPA don't exist, the HP 64869 Reader produces them. The new absolute file, TESTFILE.HPA, is then loaded into the emulator.
- If TESTFILE.HPS and TESTFILE.HPA already exist but the create dates and times are earlier than the HP 64869 format file creation date/time, the HP 64869 Reader recreates them. The new absolute file, TESTFILE.HPA, is then loaded into the emulator.

- If TESTFILE.HPS and TESTFILE.HPA already exist but the dates and times are later than the creation date/time for the HP 64869 format file, the current absolute file, TESTFILE.HPA, is then loaded into the emulator.

Note 

Date/time checking is only done within the PC Interface. When running the HP 64869 Reader at the MS-DOS command line prompt, the HP 64869 Reader will always update the absolute and symbol files.

When the HP 64869 Reader operates on a file, a status message will be displayed indicating that it is reading a HP 64869 format file. When the HP 64869 Reader completes its processing, another message will be displayed indicating the absolute file is being loaded.

If the Reader Won't Run

If your program is very large, then the PC Interface may run out of memory while attempting to create the database file used. If this condition occurs, you will need to exit the PC Interface and execute the program at the command prompt to create the files that are downloaded to the emulator.

Including RD64869 in a Make File

You may wish to incorporate the "RD64869" process as the last step in your "make" file, or as a step in your construction process, so as to eliminate the possibility of having to exit the PC Interface due to space limitations describe above. If the "-.HPA" and "-.HPS" files are not current, the process of loading an HP 64869 format file will automatically create them.

Index