HP 64730

H8/570 Emulator
PC Interface

User's Guide

HEWLETT PACKARD

Printing History

Using This Manual

This manual will show you how to use the HP 64730 H8/570 Emulator with the PC Interface.

This manual will:

- Show you how to use emulation commands by executing them on a sample program and describing their results.
- Show you how to configure the emulator for your development needs. 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 H8/570 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 H8/570 Emulator.** This chapter lists the H8/570 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 **Debugging ISP Functions.** This chapter shows you how to use the emulator to debug your ISP functions. This chapter describes how to: load ISP functions into the emulator, display ISP memory, display ISP registers, step through ISP functions, run ISP functions, set ISP breakpoints, and use the analyzer.

Chapter 4 **'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 5 **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 6 **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.

Contents

2-Contents

Illustrations

Notes

1

Introduction to the H8/570 Emulator

Introduction

The topics in this chapter include:

- Purpose of the H8/570 emulator.

- Features of the H8/570 emulator.

Purpose of the H8/570 Emulator

The H8/570 emulator is designed to replace the H8/570 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.

RS—232/RS—422
Connection

Green
Status Right

Probe Cable

Power Switch

Target System
(typically contains memory,
CPU, and I/O circuitry)

Emulator Probe
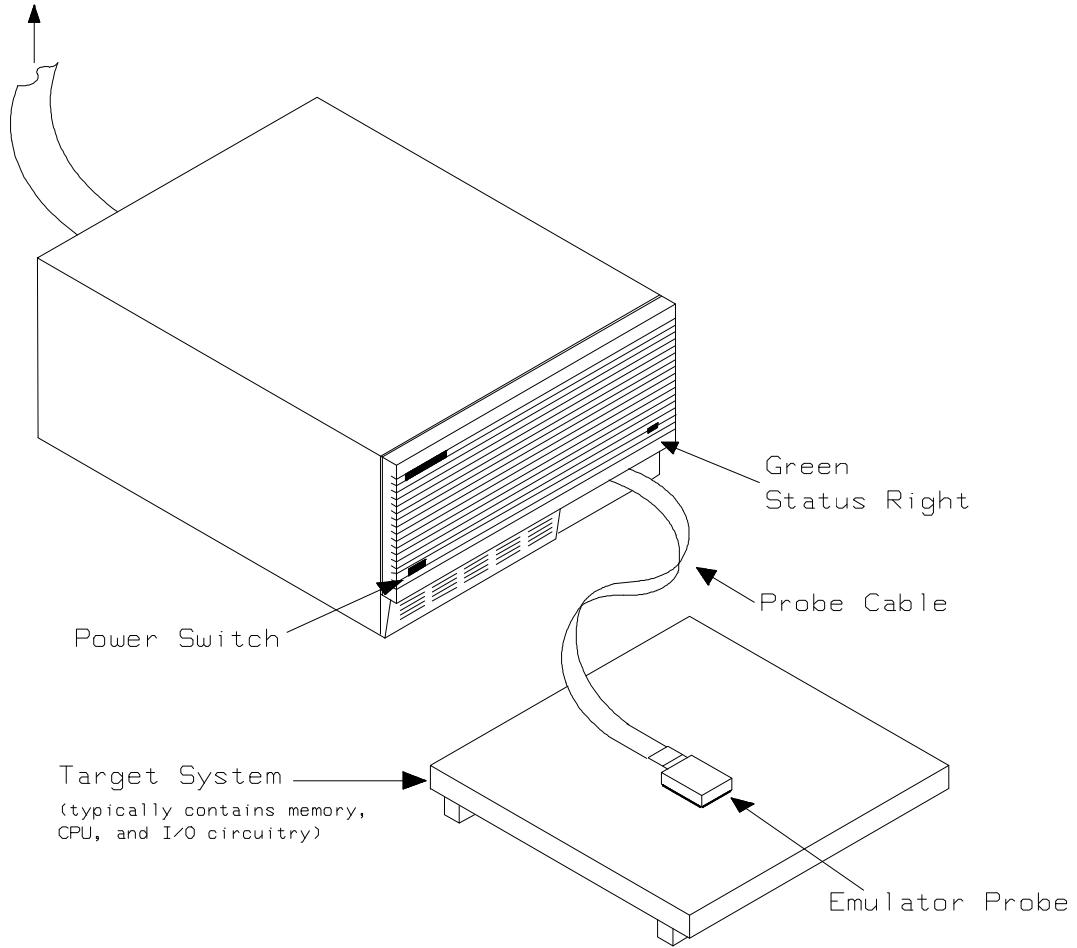
Figure 1-1. HP 64730 Emulator for the H8/570 Processor

1-2 Introduction

Features of the
H8/570 Emulator

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

Supported Microprocessors

HITACHI HD6475708F (H8/570) microprocessor is supported.

Clock Speeds

Maximum external clock speed is 12 MHz (system clock). Internal clock of the emulator is 10 MHz.

Emulation memory

The HP 64730 H8/570 emulator is used with one of the following Emulation Memory Cards.

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

The emulator uses 4K bytes of emulation memory, and the rest of emulation memory is available for user program. You can define up to 15 memory ranges (at 128 byte boundaries and at least 128 byte in length). You can characterize memory ranges as emulation RAM, emulation ROM, target system RAM, target system ROM, or as guarded memory. The emulator generates an error message when accesses are made to guarded memory locations. You can also configure the emulator so that writes to memory defined as ROM cause emulator execution to break out of target program execution.

Analysis

The HP 64730 H8/570 emulator is used with one of the following analyzers which allows you to trace code execution and processor activity.

- HP 64703 64-channel Emulation Bus Analyzer and 16-channel State/Timing Analyzer
- HP 64704 80-channel Emulation Bus Analyzer

The Emulation Bus Analyzer monitors the emulation processor using an internal analysis bus. The HP 64703 64-channel

Introduction 1-3

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 H8/570 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 "Emulator Pod Configuration" section of the "Configuring the Emulator" chapter for further details.)

Breakpoints

You can set the emulator/analyzer interaction so that when the analyzer finds a specific state, emulator execution will break out of the user program into the monitor.

You can also define software breakpoints in your program. The emulator uses one of H8/570 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 Execution

Real-time execution 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.

Easy Products Upgrades

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

Features for ISP debug

The ISP (Intelligent Subprocessor) is a programmable internal peripheral device of the H8/570 processor. The HP 64730A emulator provides useful features to debug ISP functions.

ISP Function Load

You can load your ISP functions into the microprogram memory and SCM (Sequence Control Matrix) of the emulator.

Execution Control

You can direct the ISP to run, halt, or execute a specified number of instructions.

Memory Display

You can display the contents of ISP microprogram memory in mnemonic format.

Register Display

You can display/modify the contents of H8/570 ISP registers.

Analysis

You can direct the emulator to monitor the execution of CPU program or ISP functions, or both of them.

Limitations,
Restrictions

DMA Support   Direct memory access to H8/570 emulation memory is not permitted.

Sleep and Software   When the emulator breaks into the emulation monitor, H8/570
Stand-by Mode   microprocessor sleep or software stand-by mode is released and comes to normal processor mode.

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

ISP Microprogram   The contents of ISP microprogram memory cannot be modified by
Modify   emulation commands. To modify your ISP program, you need to re-assemble/link your program, and load it into the emulator.

Symbolic Information   The H8/570 PC Interface does not support symbolic information
for ISP Functions   for ISP functions. No symbolic information for ISP functions is displayed in ISP memory display and trace listing.

RAM Enable Bit   The internal RAM of H8/510 processor can be enabled/disabled by RAME (RAM enable bit). However, the H8/570 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 64730 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 Series Installation/Service* 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 *Concepts of Emulation and Analysis* manual. The *Installtion/Service* 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 H8/570 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

WCR                 .EQU         H'FF48

                    .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 and the Wait-state
;* controller.
;*****************************************************
Init                MOV.W        #Stack,R7
                    MOV.B        #H'f0,@WCR
;*****************************************************
;* Clear previous command.
;*****************************************************
Read_Cmd            MOV.B        #0,@Cmd_Input
;*****************************************************
;* Read command input byte.  If no command has
;* been entered, continue to scan for input.
;*****************************************************
Scan                MOV.B        @Cmd_Input,R0
                    BEQ          Scan
;*****************************************************
;* A command has been entered.  Check if it is
;* command A, command B, or invalid.
;*****************************************************
Exe_Cmd             CMP.B        #H'41,R0
                    BEQ          Cmd_A
                    CMP.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.W        #Msg_B-Msg_A-1,R1
                    MOV.W        #Msg_A,R4
                    BRA          Write_Msg
;*****************************************************
;* Command B is entered.
;*****************************************************
Cmd_B               MOV.W        #Msg_I-Msg_B-1,R1
                    MOV.W        #Msg_B,R4
```

Figure 2-1. Sample Program Listing

```
                        BRA         Write_Msg
;*****************************************************
;* An invalid command is entered.
;*****************************************************
Cmd_I           MOV.W       #End_Msgs-Msg_I-1,R1
                MOV.W       #Msg_I,R4
;*****************************************************
;* Message is written to the destination.
;*****************************************************
Write_Msg       MOV.W       #Msg_Dest,R5
Again           MOV.B       @R4+,R3
                MOV.B       R3,@R5+
                SCB/EQ      R1,Again
;*****************************************************
;* The rest of the destination area is filled
;* with zeros.
;*****************************************************
Fill_Dest       MOV.B       #0,@R5+
                CMP.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      H'1
                .RES.B      H'1
;*****************************************************
;* Destination of the command messages.
;*****************************************************
Msg_Dest        .RES.B      H'3E
                .RES.W      H'80        ; Stack area.
Stack

                .END        Init
```

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(2000), 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 H8/570 PC Interface by entering the following command from the MS-DOS prompt:

        C>**pch8570** <emulname>

In the command above, **pch8570** 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 **pth8570** for **pch8570** 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.

```
┌──────────────────────────Code──────────────────────────┐
│                                                         │
│                                                         │
│                                                         │
│                                                         │
└─────────────────────────────────────────────────────────┘
┏━━━━━━━━━━━━━━━━━━━━━━━━━Emulation━━━━━━━━━━━━━━━━━━━━━━━━┓
┃                                                         ┃
┃                                                         ┃
┃                                                         ┃
┃                                                         ┃
┗━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┛
┌────────────────────────Analysis────────────────────────┐
│                                                         │
│                                                         │
│                                                         │
│                                                         │
└─────────────────────────────────────────────────────────┘
STATUS: H8/570--Emulation reset            Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Isp  Config  Analysis
 Active  Delete  Erase  Load  Open  Store  Utility  Zoom
```
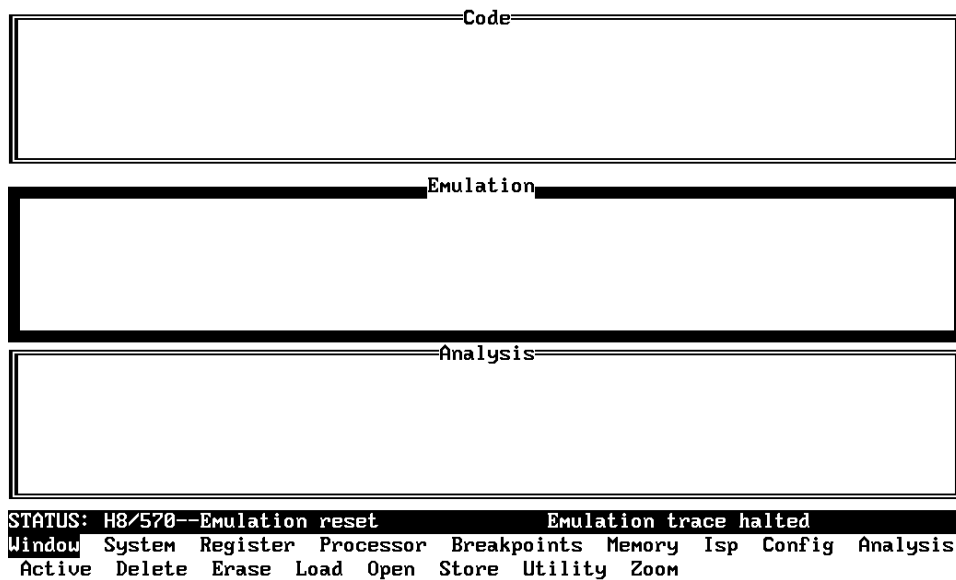
Figure 2-3. PC Interface Display

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, **G**eneral

Defining the Reset Value for the Stack Pointer

Even though the H8/570 emulator has a background monitor, it requires you to define a stack pointer.

Use the arrow keys to move the cursor to the "Reset value for Stack Pointer" field, type **0fd40** 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).

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 H8/570 emulator contains high-speed emulation memory (no wait states required) that can be mapped at a resolution of 128 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 15 different map terms.

Note

F680 hex through FE7F hex is automatically mapped as emulation RAM. This term is used for the internal RAM area of H8/570 microprocessor. You cannot delete this term.

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'0000:1046  H'00000047
                                        cmd_rds                          cmd_rds
* TOTAL   ADDRESS   *            H'0000:1000   -   H'0000:1046  H'00000047


ATTRIBUTE  :  DATA  NOSHR
Table                            H'0000:2000   -   H'0000:2030  H'00000031
                                        cmd_rds                          cmd_rds
* TOTAL   ADDRESS   *            H'0000:2000   -   H'0000:2030  H'00000031


ATTRIBUTE  :  DATA  SHR
Data                             H'0000:fc00   -   H'0000:fd3f  H'00000140
                                        cmd_rds                          cmd_rds
* TOTAL   ADDRESS   *            H'0000:fc00   -   H'0000:fd3f  H'00000140
```

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 1046 hex. The data area, which contains the ASCII values of the messages the program displays, is occupies locations 2000 hex through 2030 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 FD3F 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:

```
0..2fff
```

Move the cursor to the "memory type" field of term 1, and press the TAB key to select the **erom** (emulation ROM) 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.
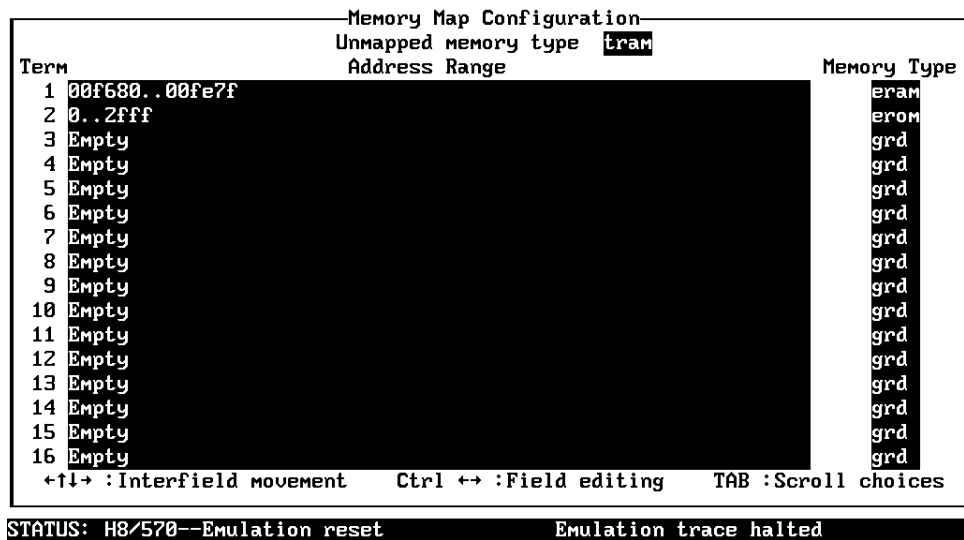
```
                        ┌Memory Map Configuration┐
                        Unmapped memory type  ▐tram▌
Term                          Address Range                    Memory Type
  1 00f680..00fe7f                                                eram
  2 0..2fff                                                       erom
  3 Empty                                                         grd
  4 Empty                                                         grd
  5 Empty                                                         grd
  6 Empty                                                         grd
  7 Empty                                                         grd
  8 Empty                                                         grd
  9 Empty                                                         grd
 10 Empty                                                         grd
 11 Empty                                                         grd
 12 Empty                                                         grd
 13 Empty                                                         grd
 14 Empty                                                         grd
 15 Empty                                                         grd
 16 Empty                                                         grd
  ←↑↓→ :Interfield movement    Ctrl ←→ :Field editing    TAB :Scroll choices
STATUS: H8/570--Emulation reset              Emulation trace halted

       Use the TAB and Shift-TAB keys to pick memory type for mapped range.
```

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. You should map all memory ranges used by your programs before loading programs into memory.

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.
- HP-MRI IEEE 695 absolute
- Raw HP64000 absolute.
- Intel hexadecimal.
- Tektronix hexadecimal.
- Motorola S-records.

The HP 64869 absolute file is generated with HP 64869 H8/500 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 64730 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" as file format. When you select "HP64000", the PC Interface creates .HPA absolute file and .HPS symbol database.

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, HP-MRI IEEE 695 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, **S**ymbols, **G**lobal, **D**isplay

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.

```
                              Symbols
 Modules
 ---------
 cmd_rds

 Address      Symbol
 ----------   ------
 000FC00      Cmd_Input
 0001000      Init
 000FC02      Msg_Dest
 0002000      Msgs










STATUS: H8/570--Emulation reset            Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Isp  Config  Analysis
 Command_file  Wait  MS-DOS  Log  Terminal  Symbols  Exit
```

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, **S**ymbols, **L**ocal, **D**isplay

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.

Getting Started 2-15

```
                             Symbols
Address       Symbol
----------    ------
0001036       Again
000101D       Cmd_A
0001025       Cmd_B
000102D       Cmd_I
000FC00       Cmd_Input
000FC00       Data
0002031       End_Msgs
0001013       Exe_Cmd
000103D       Fill_Dest
0001000       Init
0002000       Msg_A
0002011       Msg_B
000FC02       Msg_Dest
0002022       Msg_I
0002000       Msgs
0001000       Prog
0001008       Read_Cmd

STATUS: H8/570--Emulation reset           Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Isp  Config  Analysis
 Command_file  Wait  MS-DOS  Log  Terminal  Symbols  Exit
```

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, **S**ymbols, **G**lobal, **T**ransfer

Transfer the local symbol information for all modules by entering:

System, **S**ymbols, **L**ocal, **T**ransfer, **A**ll

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..102b". (You could also specify this address range using symbols, for example, **'Init..Cmd_I'** or **'Init..Init + 2b'**.) 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:I.W #fd40,R7
001003        -                MOV:G.B #f0,@ff48
001008        md_rds:Read_Cmd  MOV:G.B #00,@fc00
00100d        cmd_rds:Scan     MOV:G.B @fc00,R0
001011        -                BEQ cmd_rds:Scan
001013        cmd_rds:Exe_Cmd  CMP:E.B #41,R0
001015        -                BEQ cmd_rds:Cmd_A
001017        -                CMP:E.B #42,R0
001019        -                BEQ cmd_rds:Cmd_B
00101b        -                BRA cmd_rds:Cmd_I
00101d        cmd_rds:Cmd_A    MOV:I.W #0010,R1
001020        -                MOV:I.W #2000,R4
001023        -                BRA cmd_rds:Write_Msg
001025        cmd_rds:Cmd_B    MOV:I.W #0010,R1
001028        -                MOV:I.W #2011,R4
00102b        -                BRA cmd_rds:Write_Msg


STATUS: H8/570--Emulation reset              Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Isp  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 "102d..1045".

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.

```
                              ┌Emulation┐
001015      -                 BEQ cmd_rds:Cmd_A
001017      -                 CMP:E.B #4Z,R0
001019      -                 BEQ cmd_rds:Cmd_B
00101b      -                 BRA cmd_rds:Cmd_I
00101d      cmd_rds:Cmd_A     MOV:I.W #0010,R1
001020      -                 MOV:I.W #2000,R4
001023      -                 BRA cmd_rds:Write_Msg
001025      cmd_rds:Cmd_B     MOV:I.W #0010,R1
001028      -                 MOV:I.W #2011,R4
00102b      -                 BRA cmd_rds:Write_Msg

001000   Init              MOV:I.W #fd40,R7
PC = 001003
r0 = 0000   r1 = 0000   r2 = 0000   r3 = 0000
r4 = 0000   r5 = 0000   r6 = 0000   r7 = fd40
pc = 1003   sr = 0708   fp = 0000   sp = fd40   mdcr = c1
cp = 00     dp = 00     ep = 00     tp = 00       br = 00



STATUS: H8/570--In monitor ISP halted        Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Isp  Config  Analysis
 Go  Break  Reset  I/O  CMB  Step
```

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, **S**tep, **P**c

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, **S**tep, **P**c, **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, **S**tep, **P**c

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.

```
                              Emulation
PC = 00100d
r0 = 0000  r1 = 0000  r2 = 0000  r3 = 0000
r4 = 0000  r5 = 0000  r6 = 0000  r7 = fd40
pc = 100d  sr = 0704  fp = 0000  sp = fd40  mdcr = c1
cp = 00    dp = 00    ep = 00    tp = 00      br = 00


00100d  cmd_rds:Scan     MOV:G.B @fc00,R0
001011  -               BEQ cmd_rds:Scan
00100d  cmd_rds:Scan     MOV:G.B @fc00,R0
001011  -               BEQ cmd_rds:Scan
00100d  cmd_rds:Scan     MOV:G.B @fc00,R0
PC = 001011
r0 = 0000  r1 = 0000  r2 = 0000  r3 = 0000
r4 = 0000  r5 = 0000  r6 = 0000  r7 = fd40
pc = 1011  sr = 0704  fp = 0000  sp = fd40  mdcr = c1
cp = 00    dp = 00    ep = 00    tp = 00      br = 00



STATUS: H8/570--In monitor ISP halted       Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Isp  Config  Analysis
  Go  Break  Reset  I/O  CMB  Step
```

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.

```
                              Emulation
cp = 00     dp = 00     ep = 00     tp = 00       br = 00


00100d  cmd_rds:Scan      MOV:G.B @fc00,R0
001011  -                 BEQ cmd_rds:Scan
00100d  cmd_rds:Scan      MOV:G.B @fc00,R0
001011  -                 BEQ cmd_rds:Scan
00100d  cmd_rds:Scan      MOV:G.B @fc00,R0
PC = 001011
r0 = 0000  r1 = 0000  r2 = 0000  r3 = 0000
r4 = 0000  r5 = 0000  r6 = 0000  r7 = fd40
pc = 1011  sr = 0704  fp = 0000  sp = fd40  mdcr = c1
cp = 00     dp = 00     ep = 00     tp = 00       br = 00


Address     Data (hex)                                          Ascii
----------  --------------------------------------------------  -----------------

00fc00      41                                                  A


STATUS: H8/570--In monitor ISP halted          Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Isp  Config  Analysis
 Display  Modify  Load  Store  Copy  Find  Report
```

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, **G**o, **P**c

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

2-22 Getting Started

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, **F**ind

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, **B**reak

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 H8/570 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 (102d hex), select:

    Breakpoints, **A**dd

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 "101d;1025;102d" to set three breakpoints.

Run the program by selecting:

Processor, **G**o, **P**c

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

Memory, **M**odify, **B**ytes

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

```
ALERT:  Software breakpoint: 0102d
STATUS: H8/570--In monitor ISP halted
```

To continue program execution, select:

Processor, **G**o, **P**c

Displaying Software Breakpoints

To view the status of the breakpoint, select:

Breakpoints, **D**isplay

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, **S**et, **S**ingle

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, **C**lear, **S**ingle

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 H8/570 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 When you are using the emulator with HP 64703 analyzer, Interna/External options are displayed 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, **T**race, **R**eset

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, **T**race, **M**odify

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.

Notice that you need to specify the don't care bits. When a byte access is performed, the data appears on the upper 8 bits of analyzer data bus.

H8/570 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   Description                       Status Bits  (36..63)
--------    ------------------------------    ---------------------------------
backgrnd    Background cycle                  xxxx xxxx xxxx xxx0 0xxx xxxx xxxxB
brelease    Bus release cycle                 xxxx xxxx xxxx xxx0 x11x xxxx xxxxB
byte        Byte Access                       xxxx xxxx xxxx xxx0 x10x xxxx xx1xB
cpu         CPU cycle                         xxxx xxxx xxxx xxx0 x101 1xxx xxxxB
data        Data access                       xxxx xxxx xxxx xxx0 x10x xxxx x1xxB
dtc         DTC cycle                         xxxx xxxx xxxx xxx0 x101 0xxx xxxxB
exec        Instruction execution cycle       xxxx xxxx xxxx xxx0 x01x xxxx xxxxB
fetch       Program fetch cycle               xxxx xxxx xxxx xxx0 x101 1xxx x001B
foregrnd    Foreground cycle                  xxxx xxxx xxxx xxx0 1xxx xxxx xxxxB
grd         Guarded memory access             xxxx xxxx xxxx xxx0 x10x x011 xxxxB
io          Internal I/O access               xxxx xxxx xxxx xxx0 x10x xxx0 xxxxB
isp         Memory cycle by ISP               xxxx xxxx xxxx xxx0 xx00 1xxx xxxxB
ispexec     ISP instruction execution cycle   xxxx xxxx xxxx x0xx xxxx xxxx xxxxB
memory      Memory access                     xxxx xxxx xxxx xxx0 x10x xxx1 xxxxB
read        Read cycle                        xxxx xxxx xxxx xxx0 x10x xxxx xxx1B
refresh     Refresh cycle                     xxxx xxxx xxxx xxx0 x000 1xxx xxxxB
word        Word Access                       xxxx xxxx xxxx xxx0 x10x xxxx xx0xB
write       Write cycle                       xxxx xxxx xxxx xxx0 x10x xxxx xxx0B
wrrom       Write to ROM cycle                xxxx xxxx xxxx xxx0 x10x x101 xxx0B
```

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.

Note ☞	You need to specify the "**exec**" status qualifier to trigger the anaylzer by an execution cycle.

```
┌──────────── Internal State Trace Specification ────────────┐
│ [1] While storing [any state  ]                            │
│     Trigger on [a         ] [1] times                      │
│                                                            │
│                                                            │
│ [2] Store        [any state  ]                             │
│                                                            │
│                                                            │
│                                                            │
│                                                            │
│                                                            │
│                                                            │
│     Branches [off    ]     Count [time]    Prestore [off]  Trigger position │
│                                                            [start] of 512    │
│     ←↑↓→ :Interfield movement    Ctrl ←→ :Field editing    TAB :Scroll choices │
└────────────────────────────────────────────────────────────┘
STATUS: H8/570--Running user program        Emulation trace halted

TAB selects a pattern or press ENTER to modify this field and the pattern values
```

Figure 2-6. Modifying the Trace Specification

```
┌──────────── Internal State Trace Specification ────────────┐
│ ──────────────────────── Set 1 ──────────────────────────  │
│ Range (r) Label [addr  ] =            thru                  │
│ Pat ┌──────addr──────┬──data──┬──────stat──────┬ispaddr┬ispfunc┐ │
│  a ▣│    Cmd_Input   │   42xx │           read │       │       │ │
│  b ▣│                │        │                │       │       │ │
│  c ▣│                │        │                │       │       │ │
│  d ▣│                │        │                │       │       │ │
│     ├──────────────────── Set 2 ────────────────────────┤ │
│  e ▣│                │        │                │       │       │ │
│  f ▣│                │        │                │       │       │ │
│  g ▣│                │        │                │       │       │ │
│  h ▣│                │        │                │       │       │ │
│ arm └────────────────┴────────┴────────────────┴───────┴───────┘ │
│ ──────────────────────── Expression ─────────────────────  │
│ Expressions have the form: <set1> and/or <set2>. Where set1 consists of <a, │
│ b,c,d,r,!r> and set2 consists of <e,f,g,h,arm>. Patterns within a set can be │
│ joined with ¦(or) or ~(nor), but not both. Example: !r ~ a or e ¦ f ¦ g ¦ h │
│ Pattern Expression: [a                                    ] │
└────────────────────────────────────────────────────────────┘
STATUS: H8/570--Running user program        Emulation trace halted

TAB selects a simple pattern or enter an expression or move up to edit patterns.
```

Figure 2-7. Modifying the Pattern Specification

Starting the Trace

To start the trace, select:

    Analysis, **B**egin

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

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

Displaying the Trace

To display the trace, select:

    Analysis, **D**isplay

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.

```
                            ┌Analysis───────────────────────┐
  Line    addr,H   H8/570-ISP Mnemonic                  count,R  seq
 ─────    ──────   ──────────────────────────────────   ─────────  ───
     0    Input       42xx   read   mem byte                ---    +
     1    01011    INSTRUCTION--opcode unavailable       0.080 uS   .
     2    01014       4127   fetch mem                    0.120 uS   .
     3    e_Cmd    INSTRUCTION--opcode unavailable       0.080 uS   .
     4    01016       0640   fetch mem                    0.200 uS   .
     5    01015    BEQ cmd_rds:Cmd_A                      0.120 uS   .
     6    01018       4227   fetch mem                    0.080 uS   .
     7    01017    CMP:E.B #42,R0                         0.120 uS   .
     8    0101a       0a20   fetch mem                    0.200 uS   .
     9    01019    BEQ cmd_rds:Cmd_B                      0.080 uS   .
    10    0101c       1059   fetch mem                    0.120 uS   .
    11    Cmd_B       xx59   fetch mem                    0.400 uS   .
    12    01026       0010   fetch mem                    0.200 uS   .
    13    Cmd_B    MOV:I.W #0010,R1                       0.080 uS   .
    14    01028       5c20   fetch mem                    0.120 uS   .
    15    01028    MOV:I.W #2011,R4                       0.080 uS   .
    16    0102a       1120   fetch mem                    0.120 uS   .
```
STATUS: H8/570--Running user program   Emulation trace complete
Window System Register Processor Breakpoints Memory Isp Config Analysis
 Begin Halt CMB Format Trace 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.

```
                            ┌Analysis───────────────────────┐
    16    0102a       1120   fetch mem                    0.120 uS   .
    17    0102c       0659   fetch mem                    0.200 uS   .
    18    0102b    BRA cmd_rds:Write_Msg                  0.080 uS   .
    19    0102e       000e   fetch mem                    0.120 uS   .
    20    e_Msg       xx5d   fetch mem                    0.400 uS   .
    21    01034       fc02   fetch mem                    0.200 uS   .
    22    e_Msg    MOV:I.W #fc02,R5                       0.080 uS   .
    23    Again       c483   fetch mem                    0.120 uS   .
    24    Again    MOV:G.B @R4+,R3                        0.080 uS   .
    25    01038       c593   fetch mem                    0.120 uS   .
    26    0103a       07b9   fetch mem                    0.400 uS   .
    27    Msg_B       xx45   read   mem byte              0.200 uS   .
    28    01038    MOV:G.B R3,@R5+                        0.080 uS   .
    29    0103c       f9c5   fetch mem                    0.400 uS   .
    30    _Dest       45xx   write mem byte               0.200 uS   .
    31    0103a    SCB/EQ R1,cmd_rds:Again                0.120 uS   .
    32    0103e       0600   fetch mem                    0.200 uS   .
    33    Again       c483   fetch mem                    0.400 uS   .
    34    Again    MOV:G.B @R4+,R3                        0.080 uS   .
```
STATUS: H8/570--Running user program   Emulation trace complete
Window System Register Processor Breakpoints Memory Isp Config Analysis
 Begin Halt CMB Format Trace Display

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, **L**og, **I**nput, **E**nable

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, **G**eneral

Use the arrow keys to move the cursor to the "Reset value for Stack Pointer" field, type 0fd40, and press **End** and **Enter**.

To map the memory:

Config, **M**ap, **M**emory

Map 0 hex through 2fff hex to **erom**. (As shown in Figure 2-5.)

To load the program into memory:

Memory, **L**oad

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, **L**og, **I**nput, **D**isable

The command file cmd_rds.cmd will no longer accept command input.

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

> **S**ystem, **C**ommand_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:

> **P**rocessor, **R**eset, **H**old

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:

> **P**rocessor, **R**eset, **M**onitor

Exiting the PC Interface

There are 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, **E**xit, **L**ocked

Another way to exit the PC Interface is with the "unlocked" option which presents the default configuration next time you start the PC Interface. You can select this option with the following command.

System, **E**xit, **U**nlocked

Or, You can exit the PC Interface without saving the current configuration using the command:

System, **E**xit, **N**o_save

3

Debugging ISP Functions

The HP 64730 H8/570 emulator is equipped with commands for debugging ISP functions. You can direct the ISP to run, halt, or execute a specified number of instructions. The analyzer allows you to monitor the execution of your program, or ISP functions, or both of them.

In this chapter, we use a sample program and learn how to use the emulator to debug the ISP functions. When you have completed this chapter, you will be able to perform these tasks:

- Load ISP functions into the emulator

- Use run/stop controls to control operation of your ISP functions

- Use register display command to view the contents of ISP registers

- Use analyzer commands to view the real time execution of your ISP functions

Sample Program with Small ISP Functions

In the "Getting Started" chapter, we looked at a sample program which functioned as a primitive command interpreter. It wrote various messages to an output buffer, depending on the character you inserted in the input buffer.

In this chapter, we use a modified version of the "Getting Started" program. It still performs the same function, but works with a small ISP function. The ISP function takes charge of the transfer of the messages. Once a command is written to the input buffer, the sample program determines the message to be written and pass the source address to an ISP register. The ISP function starts to transfer the message when an ISP flag is cleared by the program. When the transfer is finished, the program goes back to read the next command. Figure 3-1 lists the sample program and Figure 3-2 lists the sample ISP functions.

Processing Commands

The instructions at **Cmd_A**, **Cmd_B**, and **Cmd_I** each load ISP data register 2 with the length of the message to be written and ISP data register 0 with the starting location of the message. Then, execution transfers to **Write_Msg** which loads the destination address into the ISP data register 1.

The ISP starts transferring a message by clearing an ISP flag. The program will wait the completion of the transfer.

ISP Function 0

ISP function 0 performs data transfer from a specified address to a destination address. ISP data register 0 is used to contain the source address. ISP data register 1 is used to contain the destination address. When the ISFL (Interrupt Status Flag) 0 is cleared, the function starts transferring data.

ISP Function 1 and 2

Function 1 and 2 are dummy functions.

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

WCR              .EQU        H'FF48
ISP_DR0          .EQU        H'FEC0
ISP_DR1          .EQU        H'FEC2
ISP_DR2          .EQU        H'FEC4
ISP_ISFL         .EQU        H'FEB1
ISP_ICSR         .EQU        H'FF19

                       .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 and the Wait-state
;* controller.  Enables the ISP.
;******************************************************
Init             MOV.W       #Stack,R7
                 MOV.W       #H'f0,@WCR
                 BCLR.B      #5,@ISP_ICSR
;******************************************************
;* Clear previous command.
;******************************************************
Read_Cmd         MOV.B       #0,@Cmd_Input
;******************************************************
;* Read command input byte.  If no command has
;* been entered, continue to scan for input.
;******************************************************
Scan             MOV.B       @Cmd_Input,R0
                 BEQ         Scan
;******************************************************
;* A command has been entered.  Check if it is
;* command A, command B, or invalid.
;******************************************************
Exe_Cmd          CMP.B       #H'41,R0
                 BEQ         Cmd_A
                 CMP.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.W       #Msg_B-Msg_A,@ISP_DR2
                 MOV.W       #Msg_A,@ISP_DR0
                 BRA         Write_Msg
;******************************************************
;* Command B is entered.
;******************************************************
```

Figure 3-1. Sample Program with ISP

```
Cmd_B            MOV.W      #Msg_I-Msg_B,@ISP_DR2
                 MOV.W      #Msg_B,@ISP_DR0
                 BRA        Write_Msg
;******************************************************
;* An invalid command is entered.
;******************************************************
Cmd_I            MOV.W      #End_Msgs-Msg_I,@ISP_DR2
                 MOV.W      #Msg_I,@ISP_DR0
;******************************************************
;* Message is written to the destination.
;******************************************************
Write_Msg        MOV.W      #Msg_Dest,@ISP_DR1
;******************************************************
;* Clear ISFL0 to start the DMA.
;******************************************************
                 BCLR.B     #0,@ISP_ISFL
Wait_ISP         BTST.B     #0,@ISP_ISFL
                 BEQ        Wait_ISP
;******************************************************
;* The rest of the destination area is filled
;* with zeros.
;******************************************************
Fill_Dest        MOV.W      @ISP_DR1,R5
Fill_Loop        MOV.B      #0,@R5+
                 CMP.W      #Msg_Dest+H'20,R5
                 BNE        Fill_Loop
;******************************************************
;* Go back and scan for next command.
;******************************************************
                 BRA        Read_Cmd

                 .SECTION   Data,COMMON
;******************************************************
;* Command input byte.
;******************************************************
Cmd_Input        .RES.B     H'1
                 .RES.B     H'1
;******************************************************
;* Destination of the command messages.
;******************************************************
Msg_Dest         .RES.B     H'3E
                 .RES.W     H'80        ; Stack area.
Stack

                 .END       Init
```

Figure 3-1. Sample Program with ISP (Cont'd)

3-4 Debugging ISP Functions

```
        .program sample;

        .SCM;
                func0/R, func1/R, func0/R, func2/R;
        .end;

        /* Function 0
         *      dr0: source address
         *      dr1: destination address
         *      dr2: loop counter
         *      isfl0: DMA starts when CPU sets this flag to 0 */
        .function func0, ar0;
        init:   out() 1, isfl0;
                next (isfl0) $, label;
        label:  next() loop;
        loop:   read.b dr0, mab          next(!c) $, labelS;
        labelS: add.w 0, #1, dr0;
                write.b dr1, mab         next(!c) $, labelD;
        labelD: add.w 0, #1, dr1;
                sub.w 0, #1, dr2         next(!z) loop2, exit;
        loop2:  next() loop;
        exit:   next() init;
        .end;

        .function func1, ar1;
        loop1:  mov.w #3, dr3;
                mov.w #0, dr3;
                next() loop1;
        .end;

        .function func2, ar2;
        loop2:  mov.w #4, dr4;
                mov.w #0, dr4;
                next() loop2;
        .end;
        .end;
```
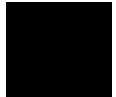
Figure 3-2. Sample ISP Functions

Assembling the Sample Program

You can assemble and link the sample program with the following commands:

```
C>h8asm cmd_rds2.src /DEBUG <RETURN>
C>h8lnk /SUBCOMMAND=cmd_rds2.k <RETURN>
```

In the above command, **cmd_rds2.k** is a linkage editor command file, and its contents is as follows:

```
debug
input cmd_rds2
start Prog(1000), Table(2000), Data(0FC00)
output cmd_rds2
print cmd_rds2
exit
```

Assembling the Sample ISP Functions

You can assemble the sample ISP functions by HITACHI ISP Assembler. Refer to the manual provided with the tool for information on the usage of the ISP assembler.

The HITACHI ISP Assembler generates absolute file in Motorola-S records. The PC Interface can load the Motorola format.

Entering the PC Interface

Start the PC Interface with the following command:

```
C>pch8570 <emul_name>
```
If you have been working with the emulator and the PC Interface is already running, please **S**ystem, **E**xit, **U**nlocked the interface and restart it. You should follow the steps to ensure that the emulator will work as described in the examples below.

Modifying Configuration

Defining the Stack Pointer

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

    Config, **G**eneral

Use the arrow keys to move the cursor to the "Reset value for Stack Pointer" field, type **0fd40**. 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

Enter the following command to enter the memory mapping display:

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

Map the following address as **erom**.

    0..2fff

Loading Absolute Files

Load the sample program with the following command:

    Memory, **L**oad

Type **cmd_rds2.abs** in the absolute file name field, and press **Enter** to start the memory load.

To load ISP functions, the ISP must be in the halt state. Halt the ISP with the following command:

    Isp, **H**alt

Load the sample ISP function:

    Isp, **L**oad

Type **ispsamp.mot** in the absolute file name field, and press **Enter** to start the memory load.

Note ☞ The only way to modify ISP microprogram memory is loading ISP functions with the **I**sp, **L**oad command. You cannot modify the memory with any emulation commands.

Displaying and Transferring Symbols

To display global and local symbols, select:

System, **S**ymbols, **G**lobal, **D**isplay

System, **S**ymbols, **L**ocal, **D**isplay

Type **cmd_rds2** and press **Enter**.

To include symbolic information in displays, transfer symbols into the emulator. Select:

System, **S**ymbols, **G**lobal, **T**ransfer

System, **S**ymbols, **L**ocal, **T**ransfer

Looking at Your ISP Code

Now that you have loaded the sample ISP function into the emulator, you can display it in mnemonic format. To display the ISP microprogram memory from address 0 through 8, select:

    Isp, **D**isplay, **A**ddress

Type **0..8** in the address filed, and press **Enter**. Use < CTRL> **z** to zoom the emulation window. You will see:

```
                              ┌Emulation┐
Adr Fn  Mnemonic
--- --  ----------------------------------------------------------------------
000 00  OUT () 1,ISFL0
        NEXT () 004
001 01  MOV.W #0003,DR3
        NEXT () 00e
002 02  MOV.W #0004,DR4
        NEXT () 010
003 ??  NEXT () 000
004 00  NEXT (ISFL0) 004,005
005 00  NEXT () 006
006 00  READ.B DR0,MAB
        NEXT (!C) 006,007
007 00  ADD.W 0,#0001,DR0
        NEXT () 008
008 00  WRITE.B DR1,MAB
        NEXT (!C) 008,009



STATUS: H8/570--In monitor ISP halted          Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Isp  Config  Analysis
 Load  Display  Go  Halt  Step  Breakpoints
```

The contents of ISP microprogram memory is displayed in mnemonic format. The first column shows the address in the microprogram memory. The second column is the number of the function to which each instruction belongs. If this field shows "??", the address is not used by any functions defined in the SCM. The third column is the instruction at the address.

You can also display instructions which belong to a specified function. For example, to see only instructions of function 0, enter:

    Isp, **D**isplay, **F**unction

Type **0** in the function number filed, and press **Enter**.

```
┌────────────────────────────Emulation────────────────────────────────────────┐
│Adr Fn  Mnemonic                                                              │
│─── ──  ────────────────────────────────────────────────────────────────────  │
│000 00  OUT () 1,ISFL0                                                         │
│        NEXT () 004                                                            │
│004 00  NEXT (ISFL0) 004,005                                                   │
│005 00  NEXT () 006                                                            │
│006 00  READ.B DR0,MAB                                                         │
│        NEXT (!C) 006,007                                                      │
│007 00  ADD.W 0,#0001,DR0                                                      │
│        NEXT () 008                                                            │
│008 00  WRITE.B DR1,MAB                                                        │
│        NEXT (!C) 008,009                                                      │
│009 00  ADD.W 0,#0001,DR1                                                      │
│        NEXT () 00a                                                            │
│00a 00  SUB.W 0,#0001,DR2                                                      │
│        NEXT (!Z) 00c,00d                                                      │
│00c 00  NEXT () 006                                                            │
│00d 00  NEXT () 000                                                            │
│                                                                              │
│                                                                              │
│STATUS: H8/570--In monitor ISP halted        Emulation trace halted           │
│Window  System  Register  Processor  Breakpoints  Memory  Isp  Config  Analysis│
│ Load  Display  Go  Halt  Step  Breakpoints                                    │
└──────────────────────────────────────────────────────────────────────────────┘
```

Note   ☞   The H8/570 PC Interface does **not** support symbolic information
for ISP functions. Symbolic information for ISP functions is not
displayed in memory display and trace listing.

Controlling ISP Execution

Reset the emulator with the following command:

> **P**rocessor, **R**eset, **H**old

Run the ISP with the following command:

> **I**sp, **G**o

The status message will be displayed as follows:

```
STATUS:  H8/570--Running in monitor
```

The ISP started execution from the current ISP address by the **run** command. The emulator breaks into the monitor when the command is used while the emulator is in the reset state.

Halt the ISP with the following command:

> **I**sp, **H**alt

```
STATUS:  H8/570--In monitor ISP halted
```

The **I**sp, **H**alt command breaks the emulator into the monitor, and halts the ISP.

Run the sample program from the **Init** label:

> **P**rocessor, **G**o, **A**ddress

```
STATUS:  H8/570--Running user program
```

Type **Init** in the address filed, and press **Enter**.

The ISP is enabled by the sample program, and starts execution. Now break the execution into the monitor:

> **P**rocessor, **B**reak

```
STATUS:  H8/570--In monitor ISP halted
```

By default, the ISP is halted when the emulator breaks into the monitor. You can configure the emulator not to halt the ISP on emulation break. Refer to Chapter 5 of this manual.

Using ISP Breakpoints

You can stop ISP execution at specific address by using the ISP breakpoints feature. When you define an ISP breakpoint to a certain address, the emulator halts the ISP and breaks into the monitor after the instruction at the address is executed.

Note   👆 Notice that the ISP is halted **after** the instruction at a breakpoint address is executed.

Note   👆 You can set/remove ISP breakpoints when the ISP is halted. To halt the ISP, use the **I**sp, **H**alt command.

Defining an ISP Breakpoints

Enter the following command to halt the ISP when the instruction at address 4 hex is executed.

   Isp, **B**reakpoints, **S**et, **S**ingle

Type 4 into the "ISP address" field. **R**un the sample program with the following command:

   Processor, **G**o, **A**ddress

Type **Init** in the address field, and press **Enter**.

ALERT: ISP breakpoint: 004

Removing ISP Breakpoints

Remove the breakpoint by the following command:

   Isp, **B**reakpoints, **R**emove

Stepping ISP Function

You can direct the emulator to execute one or specified number of ISP instructions. To step the sample ISP function from the current ISP address, select:

> Isp, Step

Press **Enter** to execute the command. You will see a similar display to the following:

```
┌─────────────────────────────Emulation──────────────────────────────┐
│007 00  ADD.W 0,#0001,DR0                                            │
│        NEXT () 008                                                  │
│008 00  WRITE.B DR1,MAB                                              │
│        NEXT (!C) 008,009                                            │
│009 00  ADD.W 0,#0001,DR1                                           │
│        NEXT () 00a                                                  │
│00a 00  SUB.W 0,#0001,DR2                                            │
│        NEXT (!Z) 00c,00d                                            │
│00c 00  NEXT () 006                                                  │
│00d 00  NEXT () 000                                                  │
│                                                                     │
│ISP breakpoint is set at address 004.                               │
│                                                                     │
│No ISP breakpoint is currently set.                                 │
│                                                                     │
│00e 01  MOV.W #0000,DR3                                             │
│        NEXT () 00f                                                  │
│AR = 004 00                                                          │
│                                                                     │
└─────────────────────────────────────────────────────────────────────┘
STATUS: H8/570--In monitor ISP halted        Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Isp  Config  Analysis
 Load  Display  Go  Halt  Step  Breakpoints
```

The first line shows the instruction executed by the step command. The last line shows the value of the ISP address register and the function number of the instruction at the address.

You can also specify the number of instructions to be executed:

> Isp, Step

Type 5 into the "number of instructions" field, and press **Enter**.

```
                              Emulation
00c 00   NEXT () 006
00d 00   NEXT () 000

ISP breakpoint is set at address 004.

No ISP breakpoint is currently set.

00e 01   MOV.W #0000,DR3
         NEXT () 00f
AR = 004 00

004 00   NEXT (ISFL0) 004,005
010 02   MOV.W #0000,DR4
         NEXT () 011
004 00   NEXT (ISFL0) 004,005
00f 01   NEXT () 001
004 00   NEXT (ISFL0) 004,005
AR = 011 02



STATUS: H8/570--In monitor ISP halted        Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Isp  Config  Analysis
 Load  Display  Go  Halt  Step  Breakpoints
```

Displaying/ Modifying ISP Registers

You can display/modify ISP registers. Registers are grouped in several "register classes." For example, to display ISP data registers, use the **ispdr** register class as follows:

Register, **D**isplay, **C**lass

Press the < TAB> key to select **ispdr** class, and press **Enter**.

```
                          ┌Emulation┐
AR = 004 00

004 00  NEXT (ISFL0) 004,005
010 02  MOV.W #0000,DR4
        NEXT () 011
004 00  NEXT (ISFL0) 004,005
00f 01  NEXT () 001
004 00  NEXT (ISFL0) 004,005
AR = 011 02

dr0  = ffff  dr1  = ffff  dr2  = ffff  dr3  = 0000
dr4  = 0000  dr5  = ffff  dr6  = ffff  dr7  = ffff
dr8  = ffff  dr9  = ffff  dr10 = ffff  dr11 = ffff
dr12 = ffff  dr13 = ffff  dr14 = ffff  dr15 = ffff
dr16 = ffff  dr17 = ffff  dr18 = ffff  dr19 = ffff
dr20 = ffff  dr21 = ffff  dr22 = ffff  dr23 = ffff
dr24 = ffff  dr25 = ffff  dr26 = ffff  dr27 = ffff
dr28 = ffff  dr29 = ffff  dr30 = ffff  dr31 = ffff


STATUS: H8/570--In monitor ISP halted      Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Isp  Config  Analysis
 Display  Modify
```

You can use the "register name" to display/modify registers. For example, to modify ISP data register 31, use the **dr31** register name as follows:

> **R**egister, **M**odify

Type **dr31** in the "Modify Register" field. Type **0** in the "To Value" filed, and press **Enter** to execute the command.

Note

Modifying registers in the **ispscm** register class is not allowed while the ISP is running. Displaying and modifying registers in the **ispdr** register class is not allowed while the ISP is running.

Refer to the Chapter 6 of this manual for the list of register classes and names.

Using the Analyzer to Debug ISP Functions

Tracing ISP Execution

You can configure the emulator to trace execution of the CPU, or ISP, or both of them. To configure the emulator to trace only execution or ISP, type:

    Config, General

Select **isp** in the "Trace CPU of ISP cycles" field, and press **End Enter** to save the configuration.

Using ISP Address for Trace Specification

Now, configure the analyzer to start the trace when the instruction at ISP address 6 hex. Select:

    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 **ispaddr** field directly opposite the pattern **a=** is highlighted. Type 6 into the field, and press **Enter**.

To save the specification, use **End Enter**. You'll return to the trace specification. Press **End Enter** to exit the trace specification display.

```
┌──────────────────── Internal State Trace Specification ─────────────────────┐
│ ▌ While storing ▐any state ▌                                                │
│   Trigger on ▐a          ▌     ▌1▌ times                                    │
│                                                                             │
│                                                                             │
│ ▌ Store           ▐any state  ▌                                            │
│                                                                             │
│                                                                             │
│                                                                             │
│                                                                             │
│                                                                             │
│                                                                             │
│                                                                             │
│     Branches ▐off      ▌      Count ▐time ▌     Prestore ▐off ▌   Trigger position│
│                                                                   ▐start ▌ of 512 │
│     ←↑↓→ :Interfield movement     Ctrl ←→ :Field editing     TAB :Scroll choices │
└─────────────────────────────────────────────────────────────────────────────┘
▐STATUS: H8/570--In monitor ISP halted        Emulation trace halted        ▌
```

▐TAB selects a pattern or press ENTER to modify this field and the pattern values▌

```
┌──────────────────── Internal State Trace Specification ─────────────────────┐
│ ┌──────────────────────────────── Set 1 ─────────────────────────────────┐ │
│ Range (r) Label ▐addr  ▌ =                    thru                         │
│ Pat ┌──────addr──────┬─────data─────┬──────stat──────┬ispaddr┬ispfunc┐    │
│  a ▐┊                │              │                │      6│       │    │
│  b ▐┊                │              │                │       │       │    │
│  c ▐┊                │              │                │       │       │    │
│  d ▐┊                │              │                │       │       │    │
│ └─────────────────────────────────┴ Set 2 ┴────────────────┴───────┴───┘ │
│  e ▐┊                │              │                │       │       │    │
│  f ▐┊                │              │                │       │       │    │
│  g ▐┊                │              │                │       │       │    │
│  h ▐┊                │              │                │       │       │    │
│ arm                                                                        │
│ ┌──────────────────────────────── Expression ───────────────────────────┐ │
│ Expressions have the form: <set1> and/or <set2>. Where set1 consists of <a,│
│ b,c,d,r,!r> and set2 consists of <e,f,g,h,arm>. Patterns within a set can be│
│ joined with |(or) or ~(nor), but not both. Example: !r ~ a or e | f | g | h │
│ Pattern Expression: ▐a                              ▌                       │
└─────────────────────────────────────────────────────────────────────────────┘
▐STATUS: H8/570--In monitor ISP halted        Emulation trace halted        ▌
```

▐TAB selects a simple pattern or enter an expression or move up to edit patterns.▌

Debugging ISP Functions 3-17

Start the trace and run the sample program:

Analysis, **B**egin

Processor, **G**o, **A**ddress

Press **Enter** to execute the command. Modify memory to let the ISP function jump to the address specified by the trace specification.

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

Enter "Cmd_Input= 41", and press **Enter**. Now display the trace list:

Analysis, **D**isplay

Type **6** in the "Ending state to display" field and press **Enter**. Use <CTRL> **z** to zoom the analysis window. You will see a display similar to the following:

```
                              Analysis
  Line    addr,H  H8/570-ISP Mnemonic                count,R  seq
 ------   ------  ------------------------------------ --------- ---
    0             006 00  READ.B DR0,MAB               0.080 uS  +
                          NEXT (!C) 006,007
    1             001 01  MOV.W #0003,DR3              0.120 uS  .
                          NEXT () 00e
    2             007 00  ADD.W 0,#0001,DR0            0.080 uS  .
                          NEXT () 008
    3             002 02  MOV.W #0004,DR4              0.120 uS  .
                          NEXT () 010
    4             008 00  WRITE.B DR1,MAB              0.080 uS  .
                          NEXT (!C) 008,009
    5             00e 01  MOV.W #0000,DR3              0.120 uS  .
                          NEXT () 00f
    6             008 00  WRITE.B DR1,MAB              0.080 uS  .
                          NEXT (!C) 008,009




 STATUS: H8/570--Running user program      Emulation trace complete
 Begin  Halt  CMB  Format  Trace  Display
  Display internal analysis trace.
```

The first column in the mnemonic field shows address of ISP microprogram memory. The second column is function number of the instruction. The third column is the mnemonic of the instruction executed.

As you can see in the above trace listing. the analyzer was triggered by an instruction at address 6.

3-18 Debugging ISP Functions

Using Function Number for Trace Specification

You also can use ISP function number for trace specification. Suppose that you want to see only instructions of ISP function 0.

Analysis, **T**race, **M**odify

Use the right arrow key to move the cursor to the "Store" field. Type "b" and press **Enter**.

You'll enter the pattern expression menu. Press the up arrow key until the **ispfunc** field directly opposite the pattern **b=** is highlighted. Type 0 into the field, and press **Enter**.

To save the specification, use **End Enter**. You'll return to the trace specification. Press **End Enter** to exit the trace specification display.

```
  ───────── Internal State Trace Specification ─────────
 1 While storing any state
   Trigger on a                  1 times


 2 Store          b




         Branches off      Count  time    Prestore off    Trigger position
                                                          start  of 512
    ←↑↓→ :Interfield movement    Ctrl ←→ :Field editing   TAB :Scroll choices
 STATUS: H8/570--Running user program        Emulation trace complete


 Use the TAB and Shift-TAB keys to select a trigger position or enter a number.
```

```
┌──────────────── Internal State Trace Specification ──────────────────┐
│                         ──────── Set 1 ────────                       │
│ Range (r) Label [addr  ]  =                    thru                   │
│ Pat ┌─────addr─────┬─────data─────┬─────stat─────────┬─ispaddr┬ispfunc┐│
│  a ▮                                                          6       ││
│  b ▮                                                                  ││
│  c ▮                                                                0 ││
│  d ▮                                                                  ││
│                         ──────── Set 2 ────────                       │
│  e ▮                                                                  ││
│  f ▮                                                                  ││
│  g ▮                                                                  ││
│  h ▮                                                                  ││
│ arm                                                                   │
│                         ──────── Expression ────────                  │
│ Expressions have the form: <set1> and/or <set2>. Where set1 consists of <a,│
│ b,c,d,r,!r> and set2 consists of <e,f,g,h,arm>. Patterns within a set can be│
│ joined with |(or) or ~(nor), but not both. Example: !r ~ a or e | f | g | h │
│ Pattern Expression: [b                                               ]│
└──────────────────────────────────────────────────────────────────────┘
 STATUS: H8/570--Running user program        Emulation trace complete
```

Start the trace, and modify memory to let the ISP function jump to
the address specified by the "Trigger on" specification.

    Analysis, **B**egin

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

Enter "Cmd_Input= 41", and press **Enter**. Now display the trace
list:

    Analysis, **D**isplay

Type **6** in the "Ending state to display" field, and press **Enter**. You
will see a display similar to the following:

```
┌──────────────────────Analysis──────────────────────────────────┐
│                                                                 │
│  Line    addr,H   H8/570-ISP Mnemonic              count,R  seq │
│  ─────   ──────   ──────────────────────────────   ───────  ─── │
│    0              006 00  READ.B DR0,MAB                ---   +  │
│                           NEXT (!C) 006,007                     │
│    1              007 00  ADD.W 0,#0001,DR0         0.200 uS  .  │
│                           NEXT () 008                           │
│    2              008 00  WRITE.B DR1,MAB           0.200 uS  .  │
│                           NEXT (!C) 008,009                     │
│    3              008 00  WRITE.B DR1,MAB           0.200 uS  .  │
│                           NEXT (!C) 008,009                     │
│    4              008 00  WRITE.B DR1,MAB           0.200 uS  .  │
│                           NEXT (!C) 008,009                     │
│    5              009 00  ADD.W 0,#0001,DR1         0.200 uS  .  │
│                           NEXT () 00a                           │
│    6     0fc02    43xx  isp write mem byte          0.200 uS  .  │
│                   00a 00  SUB.W 0,#0001,DR2                     │
│                           NEXT (!Z) 00c,00d                     │
│                                                                 │
│                                                                 │
├─────────────────────────────────────────────────────────────────┤
│STATUS: H8/570--Running user program      Emulation trace complete│
Window  System  Register  Processor  Breakpoints  Memory  Isp  Config  Analysis
 Begin  Halt  CMB  Format  Trace  Display
```

As you can see, only instructions of ISP function 0 were traced.

Tracing CPU/ISP Execution

To trace execution of both CPU and ISP, configure the emulator as follows:

> Config, General

Select **both** in the "Trace CPU or ISP cycles" field, and press **End Enter** to save the configuration.

Suppose that you want to see all states after the instruction at **Write_Msg** label is executed. Select:

> Analysis, Trace, Reset

> Analysis, Trace, Modify

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 **Write_Msg** into the field, and press **Enter**. Move the cursor to the "status" field and select the **exec** status.

```
┌─────────── Internal State Trace Specification ───────────┐
│ 1 While storing any state                                │
│   Trigger on a              1 times                       │
│                                                          │
│                                                          │
│ 2 Store          any state                               │
│                                                          │
│                                                          │
│                                                          │
│                                                          │
│                                                          │
│      Branches off         Count time     Prestore off    Trigger position │
│                                                          start of 512      │
│      ←↑↓→ :Interfield movement    Ctrl ←→ :Field editing    TAB :Scroll choices │
└──────────────────────────────────────────────────────────┘
STATUS: H8/570--Running user program        Emulation trace complete
```

```
Use the TAB and Shift-TAB keys to select a trigger position or enter a number.
```

```
┌─────────── Internal State Trace Specification ───────────┐
│ ┌──────────────────── Set 1 ────────────────────┐        │
│ Range (r) Label addr    =           thru               │
│ Pat ┌──addr──┬──data──┬──────stat──────┬ispaddr┬ispfunc┐ │
│   a =        Write_Msg                      exec        │
│   b =                                                   │
│   c =                                                   │
│   d =                                                   │
│ └──────────────────── Set 2 ────────────────────┘        │
│   e =                                                   │
│   f =                                                   │
│   g =                                                   │
│   h =                                                   │
│ arm                                                     │
│ ┌──────────────── Expression ────────────────┐          │
│ Expressions have the form: <set1> and/or <set2>. Where set1 consists of <a, │
│ b,c,d,r,!r> and set2 consists of <e,f,g,h,arm>. Patterns within a set can be │
│ joined with |(or) or ~(nor), but not both. Example: !r ~ a or e ¦ f ¦ g ¦ h │
│ Pattern Expression: a                                   │
└──────────────────────────────────────────────────────────┘
STATUS: H8/570--Running user program        Emulation trace halted
```

```
TAB selects a simple pattern or enter an expression or move up to edit patterns.
```

3-22 Debugging ISP Functions

To save the specification, use **End Enter**. You'll return to the trace specification. Press **End Enter** to exit the trace specification display.

Start the trace, and modify memory to let the ISP function jump to the address specified by the "Trigger on" specification.

Analysis, **B**egin

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

Enter "Cmd_Input= 41", and press **Enter**. Now display the trace list:

Analysis, **D**isplay

Type **6** in the "Ending state to display" field, and press **Enter**. You will see a display similar to the following:

```
                              Analysis
  Line    addr,H  H8/570-ISP Mnemonic                    count,R  seq
  -----   ------  ------------------------------------   --------  ---
    0     01047   INSTRUCTION--opcode unavailable         0.120 uS  +
                  004 00  NEXT (ISFL0) 004,005
    1             001 01  MOV.W #0003,DR3                  0.080 uS  .
                          NEXT () 00e
    2     0104a   07fc  fetch mem                         0.120 uS  .
                  004 00  NEXT (ISFL0) 004,005
    3             002 02  MOV.W #0004,DR4                  0.080 uS  .
                          NEXT () 010
    4             004 00  NEXT (ISFL0) 004,005            0.120 uS  .
    5     0104c   0215  fetch mem                         0.080 uS  .
                  00e 01  MOV.W #0000,DR3
                          NEXT () 00f
    6             004 00  NEXT (ISFL0) 004,005            0.120 uS  .




STATUS: H8/570--Running user program        Emulation trace complete
Window  System  Register  Processor  Breakpoints  Memory  Isp  Config  Analysis
 Begin  Halt  CMB  Format  Trace  Display
```

Notes

4

"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

Caution	🖐	**DAMAGE TO THE EMULATOR CIRCUITRY MAY RESULT IF THESE PRECAUTIONS ARE NOT OBSERVED.** The following precautions should be taken while using the H8/570 emulator.

Power Down Target System. Turn off power to the user target system and to the H8/570 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 adaptor 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 H8/570 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 H8/570 emulator; when powering down, turn off the emulator first, then turn off power to the target system.

Pin Guard   HP 64730 H8/570 emulator is shipped with a non-conductive 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.

Target Sytem Adaptor   The HP 64730 emulator is shipped with a target system adaptor. The adaptor allows you to connect the emulation probe to your target system which is designed for the QFP package of H8/570 microprocessor.

Pin Protector   The HP 64730 emulator is shipped with a short pin protector that prevents damage to the target system adaptor when inserting and removing the emulation probe. **Do not** insert the probe without using a short pin protector.

Installing the Target System Probe
1. Attach the adaptor to your target system. You can use a screw to hlep attaching the adaptor to the target system.

2. Install the emulation probe using the pin protector as shown in Figure 4-1.
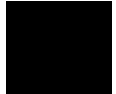
Note   You can order additional target system adaptor and short pin protector with part number 64732-61613 and 64732-61614, respectively. Contact your local HP sales representative to purcahse additional adaptor and protector.

Optional Pin Extender   If the target system probe is installed on a densely populated circuit board, there may not be a enough room to accommodate the plastic shoulders of the probe. If this occurs, you can use optional long pin protector and pin extender to avoid the conjunction with the target system components. Order the long pin protector and the pin extenders with part number 64732-61615 and 64732-61616, respectively.

Align the mark with pin 1 of the target system adaptor

Short pin protector

Target system adaptor

Pin 1 of the adaptor
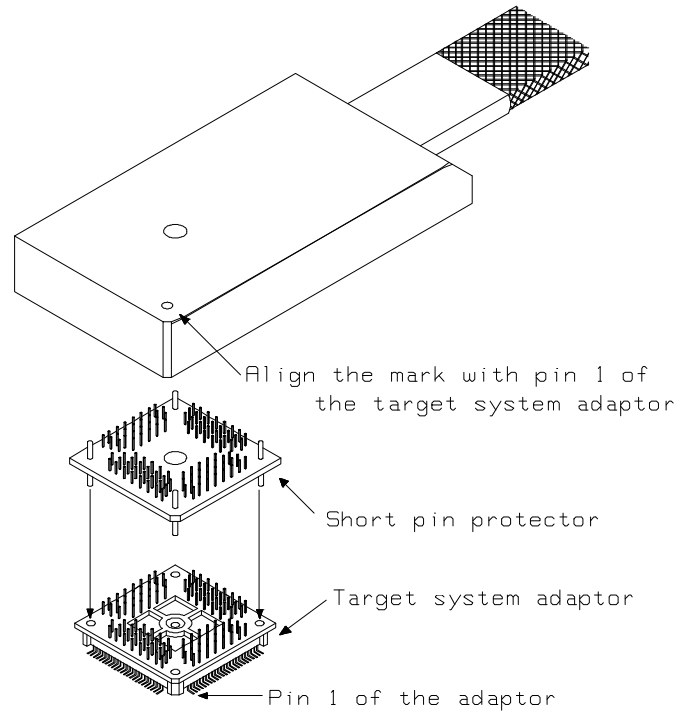
Figure 4-1. Installing the Probe

Target System Interface

Refer to the *H8/570 Terminal Interface User's Guide* for information on the target system interface of the emulator.

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, **G**o, **R**eset

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.

Notes

5
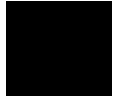
Configuring the Emulator

Introduction

The H8/570 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 H8/570 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 appears as follows:

```
┌────────────────────General Emulation Configuration────────────────────┐
│                                                                       │
│  Internal emulator clock?          [n]    Enable real-time mode?         [n]  │
│                                                                       │
│  Enable breaks on write to ROM?    [n]    Enable software breakpoints?   [n]  │
│                                                                       │
│  Enable CMB interaction?           [n]    Enable bus arbitration?        [n]  │
│                                                                       │
│  Drive background cycles to target? [n]   Enable NMI input from target?  [n]  │
│                                                                       │
│  Enable /RES input from target?    [n]    Drive reset to target?         [n]  │
│                                                                       │
│  Processer operation mode       -> ext    Trace CPU or ISP cycles?      cpu   │
│                                                                       │
│  Break ISP on CPU break?           [n]    Trace refresh cycles?          [n]  │
│                                                                       │
│  Memory data access width       -> bytes  Trace bus release cycles?      [n]  │
│                                                                       │
│  Reset value for Stack Pointer  -> 9▓▓▓▓▓▓▓▓▓▓▓▓                       │
│    ←↑↓→ :Interfield movement    Ctrl ←→ :Field editing    TAB :Scroll choices │
└───────────────────────────────────────────────────────────────────────┘
 STATUS: H8/570--Running in monitor           Trace Complete

 If enabled, the emulator uses the internal 10 MHz clock.  Otherwise, the clock
 input from the target system clocks the emulator.
```

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 H8/570 microprocessor. The maximum external clock speed is 12 MHz (system clock).

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.
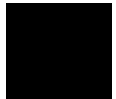
Enable Software Breakpoints?

When you define or enable a software breakpoint to a specified address, the emulator will replace the opcode with one of H8/570 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, **C**MB, **G**o, ...
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.

yes   Specifies that background cycles are driven to the target system. Emulation processor's address and control strobes (except /HWR and /LWR) 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   Memory cycles by ISP are always driven to the target system while the emulator in running in 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.

The emulator does not accept any interrupt during background execution. NMI is latched last one during in background, and such interrupt will occur when context is changed to foreground. IRQ0 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.

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 and reset by the Watchdog timer.

no   Specifies that the emulator will not drive the /RES signal during emulation reset and reset by the Watchdog timer.

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

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

Caution

If you intend to drive the reset signal to the target system, the driver of reset signal on the target **must** be an open collector or open drain. Otherwise, it may result in damage to target system or emulation circuitry.

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 1.

1 The emulator will operate in mode 1. (expanded minimum mode with 16 bit data bus)

3 The emulator will operate in mode 3. (expanded maximum mode with 16 bit data bus)

4 The emulator will operate in mode 4. (expanded minimum mode with 8 bit data bus)

5 The emulator will operate in mode 5. (expanded maximum mode with 16 bit data bus)

6 The emulator will operate in mode 6. (expanded maximum mode with 8 bit data bus)

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

Note In mode 3 and 6, ffe80 hex through fff7f hex is used as internal I/O register area. To display/modify this address by emulation commands, you need to map this area as target memory.

Trace CPU or ISP cycles?

This configuration allows you to select if the emulation analyzer traces CPU execution, or ISP execution, or both of them.

cpu   The emulation analyzer doesn't trace ISP execution. The following is a sample trace listing of this mode.

```
                              ┌Analysis┐
  Line    addr,H  H8/570-ISP Mnemonic                    count,R  seq
  -----   ------  ------------------------------------   --------- ---
      0   e_Msg     xx1d  fetch mem                          ---    +
      1   01048     fec2  fetch mem                       0.280 uS  .
      2   e_Msg   MOV:G.W #fc02,@cmd_rds2:ISP_DR1         0.120 uS  .
      3   0104a     07fc  fetch mem                       0.200 uS  .
      4   0104c     0215  fetch mem                       0.280 uS  .
      5   0104e     feb1  fetch mem                       0.400 uS  .
      6   P_DR1     fc02  write i/o word                  0.320 uS  .
      7   0104d   BCLR.B #0,@cmd_rds2:ISP_ISFL            0.080 uS  .
      8   01050     d015  fetch mem                       0.200 uS  .
      9   01052     feb1  fetch mem                       0.320 uS  .
     10   _ISFL     xx01  read  i/o byte                  0.280 uS  .
     11   _ISFL     xx00  write i/o byte                  0.320 uS  .
     12   t_ISP   BTST.B #0,@cmd_rds2:ISP_ISFL            0.080 uS  .
     13   01054     f027  fetch mem                       0.200 uS  .
     14   01056     fa1d  fetch mem                       0.320 uS  .
     15   Msgs      43xx  isp read  mem byte              0.280 uS  .


STATUS: H8/570--Running user program        Emulation trace complete
Window  System  Register  Processor  Breakpoints  Memory  Isp  Config  Analysis
 Begin  Halt  CMB  Format  Trace  Display
```

isp   The emulation analyzer traces only ISP execution and memory cycles by ISP. The following is a sample trace listing of this mode.

```
                        Analysis

   Line    addr,H   H8/570-ISP Mnemonic            count,R  seq
   -----   ------   -----------------------------  -------  ---
      0             006 00   READ.B DR0,MAB          0.080 uS   +
                             NEXT (!C) 006,007
      1             010 02   MOV.W #0000,DR4         0.120 uS   .
                             NEXT () 011
      2             007 00   ADD.W 0,#0001,DR0       0.080 uS   .
                             NEXT () 008
      3             00f 01   NEXT () 001             0.120 uS   .
      4             008 00   WRITE.B DR1,MAB         0.080 uS   .
                             NEXT (!C) 008,009
      5             011 02   NEXT () 002             0.120 uS   .
      6             008 00   WRITE.B DR1,MAB         0.080 uS   .
                             NEXT (!C) 008,009
      7    Msgs     43xx  isp read  mem byte         0.120 uS   .
                    001 01   MOV.W #0003,DR3
                             NEXT () 00e


STATUS: H8/570--Running user program       Emulation trace complete
Window  System  Register  Processor  Breakpoints  Memory  Isp  Config  Analysis
 Begin  Halt  CMB  Format  Trace  Display
```

The first column in the mnemonic field shows address of ISP microprogram memory. The second column is function number of the instruction. The third column is the mnemonic of the ISP instruction executed.

both   The emulation analyzer traces both of CPU and ISP execution. The following is a sample trace listing of this mode.

```
                      ┌─Analysis─────────────────────────────────────────────────────┐
                      │  Line   addr,H  H8/570-ISP Mnemonic              count,R  seq  │
                      │  ─────   ──────  ─────────────────────────────   ────────  ───  │
                      │     0   e_Msg    xx1d  fetch mem                     ---   +    │
                      │                  004 00  NEXT (ISFL0) 004,005                   │
                      │     1            011 02  NEXT () 002              0.080 uS  .    │
                      │     2            004 00  NEXT (ISFL0) 004,005     0.120 uS  .    │
                      │     3   01048    fec2  fetch mem                  0.080 uS  .    │
                      │                  001 01  MOV.W #0003,DR3                        │
                      │                          NEXT () 00e                           │
                      │     4   e_Msg   MOV:G.W #fc02,@cmd_rds2:ISP_DR1   0.120 uS  .    │
                      │                  004 00  NEXT (ISFL0) 004,005                   │
                      │     5            002 02  MOV.W #0004,DR4          0.080 uS  .    │
                      │                          NEXT () 010                           │
                      │     6   0104a    07fc  fetch mem                  0.120 uS  .    │
                      │                  004 00  NEXT (ISFL0) 004,005                   │
                      │     7            00e 01  MOV.W #0000,DR3          0.080 uS  .    │
                      │                          NEXT () 00f                           │
                      │     8            004 00  NEXT (ISFL0) 004,005     0.120 uS  .    │
                      │                                                                │
                      │                                                                │
                      └────────────────────────────────────────────────────────────────┘
 STATUS: H8/570--Running user program          Emulation trace complete
 Window  System  Register  Processor  Breakpoints  Memory  Isp  Config  Analysis
  Begin  Halt  CMB  Format  Trace  Display
```

Break ISP on CPU break?

This configuration allows you to select whether the emulator halts the ISP when CPU breaks into the monitor.

yes   When you enable this option, the emulator halts the ISP when CPU breaks into the monitor.

no   When you disable this option, the ISP continues execution when CPU breaks into the monitor. This configuration is useful if your target system requires constant execution of the ISP.

Configuring the Emulator 5-15

Trace Refresh Cycles?

You can direct the analyzer to trace refresh cycles or not.

yes   The analyzer will trace refresh cycles.

no   The analyzer will ignore refresh cycles.

Memory Data Access Width

This question allows you to specify the types of cycles that the emulation monitor use when accessing target system memory. When an emulation command requests the monitor to read or write target system memory locations, the monitor will either use byte or word instructions to accomplish the read/write.

bytes   Specifies that the emulator will access target system memory by byte access.

word   Specifies that the emulator will access target system memory by word access.

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).

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.
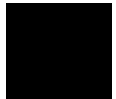
Storing an Emulator Configuration

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

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

To store the current emulator configuration, select:

Config, **S**tore

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, **L**oad

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

6

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, **G**eneral

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, **C**MB, **G**o

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, **T**rigger

The trigger configuration display contains two diagrams, one for each of the internal TRIG1 and TRIG2 signals.
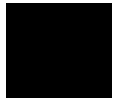
To use the internal TRIG1 signal to connect the analyzer trigger to the emulator break line, move the cursor to the highlighted "Analyzer" field in the TRIG1 portion of the display, and use the **Tab** key to select the "----->>" arrow which shows that the analyzer is driving TRIG1. Next, move the cursor 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.

```
┌──────────────────Cross Trigger Configuration──────────────────┐
│                                                                │
│                  TRIG1                          TRIG2          │
│          BNC ignore ═══════╗          BNC ignore ═══════╗      │
│                            ║                            ║      │
│          CMB ignore ═══════╣          CMB ignore ═══════╣      │
│                            ║                            ║      │
│     Emulator ≪───── ═══════╣     Emulator ignore ═══════╣      │
│                            ║                            ║      │
│     Analyzer ─────≫ ═══════╝     Analyzer ignore ═══════╝      │
│                                                                │
│                                                                │
│                                                                │
│    ←↑↓→ :Interfield movement   Ctrl ←→ :Field editing   TAB :Scroll choices │
└────────────────────────────────────────────────────────────────┘
STATUS: H8/570--Emulation reset            Emulation trace halted

The internal analyzer may drive (-----≫≫) , receive (≪≪-----) or ignore the
TRIG1 and TRIG2 signals.
```
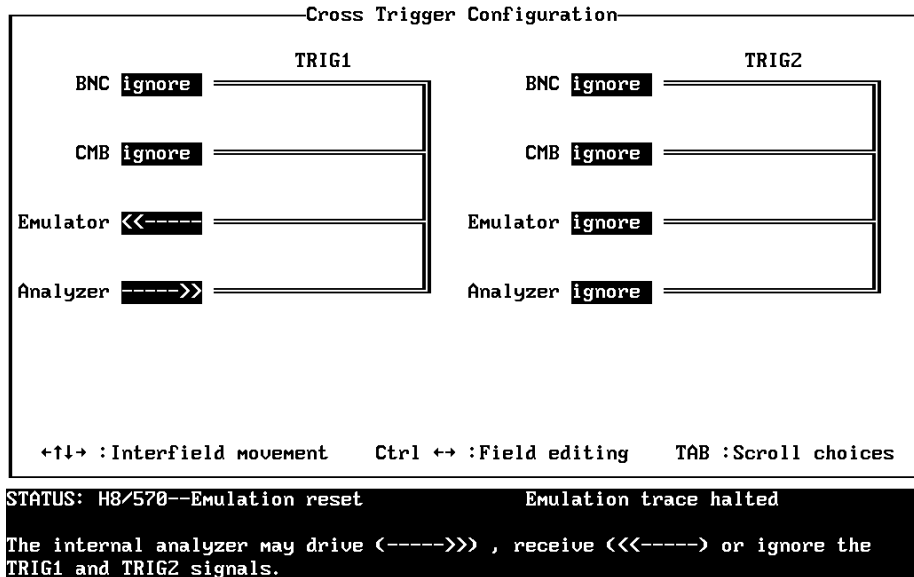
Note   If your emulator is configured with external analyzer, "Timing" cross trigger options are displayed.

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, **S**tore

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, **I**O, **D**isplay
        Processor, **I**O, **M**odify

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 **H8/570 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
mdcr	Mode control register
sbycr	Software stand-by control register
ramcr	RAM control register
syscr1	System control register 1

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

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

adc Class   A/D converter registers

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
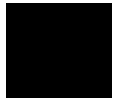
port Class   I/O port registers

Register name	Description
p1ddr	Port 1 data direction register
p5ddr	Port 5 data direction register
p6ddr	Port 6 data direction register
p8ddr	Port 8 data direction register
p9ddr	Port 9 data direction register
p10ddr	Port 10 data direction register
p11ddr	Port 11 data direction register
p12ddr	Port 12 data direction register
p1dr	Port 1 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
p10dr	Port 10 data register
p11dr	Port 11 data register
p12dr	Port 12 data register

pwm Class   PWM timer registers

tcr	Timer control register
tsr	Timer status register
odl	Output data latch
odr0	Output data register 0
odr1	Output data register 1
odr2	Output data register 2
ocr0	Output compare register 0
ocr1	Output compare register 1
ocr2	Output compare register 2
tmr	Timer

wdt Class   Watchdog timer registers

Register name	Description
wdtcsr	Timer control/status register
wdtcnt	Timer counter
rstcsr	Reset control/status register

sci Class   Serial communication interface registers.

rdr	Receive data register
tdr	Transmit data register
smr	Serial mode register
scr	Serial control register
ssr	Serial status register
brr	Bit rate register

adc Class   A/D converter registers

addra	A/D data register A
addrb	A/D data register B
addrc	A/D data register C
addrd	A/D data register D
adcsr	A/D control/status register
adcr	A/D control register
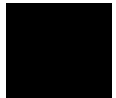
ispscm Class   ISP SCM

Register name	Description
ar0	ISP address register 0
ar1	ISP address register 1
ar2	ISP address register 2
:	:
:	:
ar9	ISP address register 9
ar10	ISP address register 10
ar11	ISP address register 11

ispdr Class   ISP data registers

dr0	ISP data register 0
dr1	ISP data register 1
dr2	ISP data register 2
dr3	ISP data register 3
:	:
:	:
dr30	ISP data register 30
dr31	ISP data register 31

ispf Class   ISP flags

icf	Interconnction flag
iof0	Input/output flag 0
iof1	Input/output flag 1
iof2	Input/output flag 2
egf	Edge flag
isf	Interrupt status flag

'

ispc Class ISP control registers

Register name	Description
ief	Interrupt enable flag
ioief	I/O interrupt enable flag
cle	Clear enable register
ever	Event enable register
ipr	ISP page register
icsr	ISP control status register
redge	Rising edge enable register
fedge	Falling egde enable register
syscr8	System control register 8
syscr9	System control register 9
syscr10	System control register 10

Notes

A
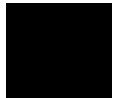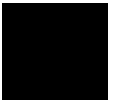
File Format Readers

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.
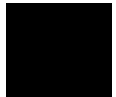
Using the Reader from MS-DOS

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."

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.
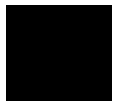
Including RHP64000 in a Make File

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 64730 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 H8/570 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."
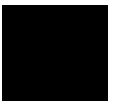
What the Reader Accomplishes

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 H8/570 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
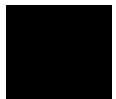
File Format Readers A-7

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 H8/570 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 H8/570 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 H8/570 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.

Using the IEEE-695 Reader

An IEEE-695 MUFOM (Microprocessor Universal Format for Object Modules) "reader" comes with the PC Interface. The IEEE-695 reader converts an IEEE-695 format file into two files that are usable with the emulator. This means you can use available language tools to create IEEE-695 absolute files, then load those files into the emulator from the PC Interface.

The IEEE-695 reader can operate from within the PC Interface or as a separate process. You may need to execute the reader as a separate process if there is not enough memory on your personal computer to run the PC Interface and the reader simultaneously.

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

What the Reader Accomplishes

The IEEE-695 reader accepts as input an IEEE-695 format absolute file in the form "< file> .< ext> " and creates two new files that are used by the PC Interface: an "absolute" file, and an ASCII symbol file.

The Absolute File

During execution of the IEEE-695 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 IEEE-695 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 your specific language tools to include symbolic ("debug") information in the IEEE-695 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
```

The space preceding module names is required. A single tab separates symbol and address.

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

The local symbols are scoped. This means that to access a variable named "count" in a function named "foo" in a source file module named "main.c," you would enter "main.c:foo.count." See the following table.

Module Name	Function Name	Variable Name	You Enter
main.c	foo	count	main.c:foo.count
main.c	bar	count	main.c:bar.count

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.

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

Location of the IEEE-695 Reader Program

The IEEE-695 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 IEEE-695 reader and PC Interface to operate properly. This is usually defined in the "\autoexec.bat" file.

Using the IEEE-695 Reader from MS-DOS

The command name for the IEEE-695 reader is **RIEEE695.EXE**. You can execute the IEEE-695 reader from the command line with the following command syntax:

```
C:\HP64700\BIN\RIEEE695 [-u] [-q] <filename>
<RETURN>
```

[-u]   Specifies that the first leading underscore of a symbol is not removed.

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

< filename>   Specifies the name of the file containing the IEEE-695 absolute program.

Using the IEEE-695 Reader from the PC Interface

The H8/570 Series PC Interface has a file format option under the "**M**emory **L**oad" command. After you select this option, the IEEE-695 reader will operate on the file you specify. After the reader completes successfully, the PC Interface will load the absolute and symbol files produced by the Reader.

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

1. Start the PC Interface.

2. Select "**M**emory **L**oad." The memory load menu will appear.

3. Specify the file format as "IEEE-695." This will appear as the default file format.

4. Specify the memory to be loaded (emulation, target, or both).

5. Specify a file in IEEE-695 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 IEEE-695 reader to process the symbols.

Using the IEEE-695 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 IEEE-695 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 IEEE-695 file creation date/time, the IEEE-695 reader re-creates 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 IEEE-695 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 you run the IEEE-695 reader at the MS-DOS command line prompt, the reader will always update the absolute and symbol files.

When the IEEE-695 reader operates on a file, a status message will be displayed indicating that it is reading an IEEE-695 file. When the reader completes its processing, another message will be displayed indicating the absolute file is being loaded.

If the IEEE-695 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. If this occurs, exit the PC Interface and execute the reader program at the MS-DOS command prompt.

Including RIEEE695 in a Make File

You may want to incorporate the "RIEEE695" 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, loading an IEEE-695 file will automatically create them.

Notes

Index