HP 64793
H8/338/329 Emulator
PC Interface

User’s Guide
Notice

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

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

© Copyright 1992, Hewlett-Packard Company.

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

UNIX is a registered trademark of AT&T.

Torx is a registered trademark of Camcar Division of Textron, Inc.

Hewlett-Packard Company
Logic Systems Division
8245 North Union Boulevard
Colorado Springs, CO 80920, U.S.A.
New editions are complete revisions of the manual. The date on the title page changes only when a new edition is published.

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

Edition 1 64793-97001, October 1992
Using This Manual

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

- HP 64793A H8/338 emulator
- HP 64793B H8/329 emulator

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

<table>
<thead>
<tr>
<th>Model</th>
<th>Supported Microprocessors</th>
<th>Referred to as</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP 64793A (H8/338 emulator)</td>
<td>HD6473388CP</td>
<td>H8/338</td>
</tr>
<tr>
<td></td>
<td>HD6433388CP</td>
<td>H8/338</td>
</tr>
<tr>
<td></td>
<td>HD6413388CP</td>
<td>H8/338</td>
</tr>
<tr>
<td></td>
<td>HD6473378CP</td>
<td>H8/337</td>
</tr>
<tr>
<td></td>
<td>HD6433378CP</td>
<td>H8/337</td>
</tr>
<tr>
<td></td>
<td>HD6413378CP</td>
<td>H8/337</td>
</tr>
<tr>
<td></td>
<td>HD6433368CP</td>
<td>H8/336</td>
</tr>
<tr>
<td>HP 64793B (H8/329 emulator)</td>
<td>HD6473298P</td>
<td>H8/329</td>
</tr>
<tr>
<td></td>
<td>HD6473298C</td>
<td>H8/329</td>
</tr>
<tr>
<td></td>
<td>HD643298P</td>
<td>H8/329</td>
</tr>
<tr>
<td></td>
<td>HD6413298P</td>
<td>H8/329</td>
</tr>
<tr>
<td></td>
<td>HD643288P</td>
<td>H8/328</td>
</tr>
<tr>
<td></td>
<td>HD6473278P</td>
<td>H8/327</td>
</tr>
<tr>
<td></td>
<td>HD6473278C</td>
<td>H8/327</td>
</tr>
<tr>
<td></td>
<td>HD643278P</td>
<td>H8/327</td>
</tr>
<tr>
<td></td>
<td>HD6413278P</td>
<td>H8/327</td>
</tr>
<tr>
<td></td>
<td>HD6433268P</td>
<td>H8/326</td>
</tr>
</tbody>
</table>

For the most part, the H8/338 and H8/329 emulators all operate the same way. Differences between the emulators are described where they exist. Both the H8/338 and H8/329 emulators will be referred to as the "H8/338 emulator". In the specific instances where H8/329
emulator differs from H8/338 emulator, it will be described as the “H8/329 emulator”.

This manual:

- Shows you how to use emulation commands by executing them on a sample program and describing their results.

- Shows you how to use the emulator in-circuit (connected to a target system).

- Shows you how to configure the emulator for your development needs. Topics include: restricting the emulator to real-time execution, selecting a target system clock source, and allowing the target system to insert wait states.

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

<table>
<thead>
<tr>
<th>Chapter 1</th>
<th>Introduction to the H8/338 Emulator. This chapter lists the H8/338 emulator features and describes how they can help you in developing new hardware and software.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 2</td>
<td>Getting Started. This chapter shows you how to use emulation commands by executing them on a sample program. This chapter describes the sample program and how to: load programs into the emulator, map memory, display and modify memory, display registers, step through programs, run programs, set software breakpoints, search memory for data, and use the analyzer.</td>
</tr>
<tr>
<td>Chapter 3</td>
<td>&quot;In-Circuit&quot; Emulation. This chapter shows you how to plug the emulator into a target system, and how to use the &quot;in-circuit&quot; emulation features.</td>
</tr>
<tr>
<td>Chapter 4</td>
<td>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.</td>
</tr>
<tr>
<td>Chapter 5</td>
<td>Using the Emulator. This chapter describes emulation topics which are not covered in the &quot;Getting Started&quot; chapter (for example, coordinated measurements and storing memory).</td>
</tr>
<tr>
<td>Appendix A</td>
<td>File Format Readers. This chapter shows you what the &quot;Reader&quot; program accomplishes, and how to use it.</td>
</tr>
</tbody>
</table>
# Contents

## 1 Introduction to the H8/338 Emulator

<table>
<thead>
<tr>
<th>Introduction</th>
<th>1-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose of the H8/338 Emulator</td>
<td>1-1</td>
</tr>
<tr>
<td>Features of the H8/338 Emulator</td>
<td>1-3</td>
</tr>
<tr>
<td>Supported Microprocessors</td>
<td>1-3</td>
</tr>
<tr>
<td>Clock Speeds</td>
<td>1-4</td>
</tr>
<tr>
<td>Emulation Memory</td>
<td>1-4</td>
</tr>
<tr>
<td>Analysis</td>
<td>1-4</td>
</tr>
<tr>
<td>Registers</td>
<td>1-4</td>
</tr>
<tr>
<td>Single-Step</td>
<td>1-4</td>
</tr>
<tr>
<td>Breakpoints</td>
<td>1-4</td>
</tr>
<tr>
<td>Reset Support</td>
<td>1-5</td>
</tr>
<tr>
<td>Real-Time Operation</td>
<td>1-5</td>
</tr>
<tr>
<td>Limitations, Restrictions</td>
<td>1-5</td>
</tr>
<tr>
<td>Foreground Monitor</td>
<td>1-5</td>
</tr>
<tr>
<td>Sleep and Software Standby Mode</td>
<td>1-5</td>
</tr>
<tr>
<td>Store Condition and Trace</td>
<td>1-6</td>
</tr>
<tr>
<td>Step Command and Interrupts</td>
<td>1-6</td>
</tr>
<tr>
<td>RAM Enable Bit</td>
<td>1-6</td>
</tr>
</tbody>
</table>

## 2 Getting Started

<table>
<thead>
<tr>
<th>Introduction</th>
<th>2-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before You Begin</td>
<td>2-2</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>2-2</td>
</tr>
<tr>
<td>A Look at the Sample Program</td>
<td>2-2</td>
</tr>
<tr>
<td>Sample Program Assembly</td>
<td>2-6</td>
</tr>
<tr>
<td>Linking the Sample Program</td>
<td>2-6</td>
</tr>
<tr>
<td>Starting Up the PC Interface</td>
<td>2-7</td>
</tr>
<tr>
<td>Selecting PC Interface Commands</td>
<td>2-8</td>
</tr>
<tr>
<td>Emulator Status</td>
<td>2-8</td>
</tr>
<tr>
<td>Modifying Configuration</td>
<td>2-8</td>
</tr>
<tr>
<td>Defining the Reset Value for the Stack Pointer</td>
<td>2-8</td>
</tr>
<tr>
<td>Selecting your Processor</td>
<td>2-8</td>
</tr>
<tr>
<td>Saving the Configuration</td>
<td>2-9</td>
</tr>
</tbody>
</table>
Mapping Memory ................................................. 2-9
Which Memory Locations Should Be Mapped? .............. 2-9
Loading Programs into Memory ................................. 2-12
File Format ....................................................... 2-12
Memory Type ...................................................... 2-13
Force Absolute File Read ....................................... 2-13
Absolute File Name .............................................. 2-13
Using Symbols .................................................... 2-14
Displaying Global Symbols ..................................... 2-14
Displaying Local Symbols ...................................... 2-15
Transfer Symbols to the Emulator .............................. 2-17
Displaying Memory in Mnemonic Format ..................... 2-18
Stepping Through the Program ................................. 2-19
Specifying a Step Count ....................................... 2-20
Modifying Memory .............................................. 2-21
Running the Program ............................................ 2-22
Searching Memory for Data ..................................... 2-23
Breaking into the Monitor ....................................... 2-23
Using Software Breakpoints .................................... 2-24
Defining a Software Breakpoint ............................... 2-25
Displaying Software Breakpoints .............................. 2-26
Setting a Software Breakpoint ................................. 2-26
Clearing a Software Breakpoint ............................... 2-26
Using the Analyzer ............................................. 2-27
Resetting the Analysis Specification ......................... 2-27
Specifying a Simple Trigger .................................. 2-27
Starting the Trace .............................................. 2-29
Displaying the Trace .......................................... 2-29
For a Complete Description .................................. 2-31
Trace Analysis Considerations ................................. 2-32
How to Specify Trigger Condition ............................ 2-32
Store Condition And Trace .................................... 2-35
Triggering the Analyzer by Data .............................. 2-38
Using a Command File ........................................... 2-39
Resetting the Emulator ......................................... 2-41
Exiting the PC Interface ....................................... 2-42

3 Using the H8/338 Emulator In-Circuit

Installing the Target System Probe ......................... 3-2
Pin Guard ......................................................... 3-2
Pin Protector (H8/329 Only) .................................. 3-3

2-Contents
4 Configuring the Emulator

Introduction ............................. 4-1
Accessing the Emulator Configuration Options .................. 4-2
Internal Emulator Clock .................... 4-3
Enable Real-Time Mode ..................... 4-3
Enable Breaks on Writes to ROM ............. 4-4
Enable Software Breakpoints ................ 4-5
Enable CMB Interaction ................... 4-5
Enable /NMI Input from Target ............... 4-6
Enable / RES Input from Target ............. 4-7
Processor Type (H8/338 emulator) ............ 4-7
Processor Type (H8/329 emulator) ............ 4-8
Processor Operation Mode .................. 4-8
Reset Value for Stack Pointer .............. 4-9
Storing an Emulator Configuration .......... 4-10
Loading an Emulator Configuration .......... 4-11

5 Using the Emulator

Introduction ................................ 5-1
Making Coordinated Measurements .......... 5-2
   Running the Emulator at /EXECUTE ........ 5-3
   Breaking on the Analyzer Trigger .......... 5-3
Storing Memory Contents to an Absolute File .............. 5-5
Register Classes and Names (H8/338 Emulator) ............. 5-6
Register Classes and Names (H8/329 Emulator) ............. 5-11

A File Format Readers

Using the HP 64000 Reader .................. A-1
   What the Reader Accomplishes ............. A-1
   Location of the HP 64000 Reader Program .... A-3
   Using the Reader from MS-DOS ............... A-4
   Using the Reader from the PC Interface ....... A-4
   If the Reader Won’t Run ..................... A-5
   Including RHP64000 in a Make File ........... A-6
Using the HP 64869 Reader ............................................. A-6
What the Reader Accomplishes ................................ A-6
Location of the HP 64869 Reader Program ...................... A-8
Using the HP 64869 Reader from MS-DOS ....................... A-8
Using the HP 64869 Reader from the PC Interface .............. A-9
If the Reader Won’t Run ................................................. A-10
Including RD64869 in a Make File ................................. A-10

Illustrations

Figure 1-1. HP 64793 Emulator for the H8/338 Processor ...... 1-2
Figure 2-1. Sample Program Listing ................................. 2-3
Figure 2-2. Linkage Editor Subcommand File ..................... 2-6
Figure 2-3. PC Interface Display ...................................... 2-7
Figure 2-4. Sample Program Load Map Listing .................... 2-10
Figure 2-5. Memory Configuration Display ......................... 2-11
Figure 2-6. Modifying the Trace Specification ...................... 2-28
Figure 2-7. Modifying the Pattern Specification ................... 2-29
Figure 3-1. Installing the Probe (H8/338 emulator) ................ 3-4
Figure 3-2. Installing the Probe (H8/329 emulator) .............. 3-5
Figure 4-1. Emulator Configuration Display ....................... 4-2
Figure 5-1. Cross Trigger Configuration ............................ 5-4
Introduction to the H8/338 Emulator

Introduction

The topics in this chapter include:

- Purpose of the H8/338 emulator.
- Features of the H8/338 emulator.

Purpose of the H8/338 Emulator

The H8/338 emulator is designed to replace the H8/338 microprocessor in your target system so you can control operation of the microprocessor in your application hardware (usually referred to as the target system). The H8/338 emulator performs just like the H8/338 microprocessor, but is a device that allows you to control the H8/338 directly. These features allow you to easily debug software before any hardware is available, and ease the task of integrating hardware and software.
1-2 Introduction to the H8/338 Emulator

Figure 1-1. HP 64793 Emulator for the H8/338 Processor

- RS-232/RS-422 Connection
- Power Switch
- Green Status Right
- Probe Cable
- Target System (typically contains memory, CPU, and I/O circuitry)
- Emulator Probe
Features of the H8/338 Emulator

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

Supported Microprocessors

The HP 64793A H8/338 emulator and HP 64793B H8/329 emulators support the microprocessors listed in the following table.

<table>
<thead>
<tr>
<th>Model</th>
<th>Supported Microprocessor</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP 64793A (H8/338 emulator)</td>
<td>HD6473388CP (H8/338)</td>
</tr>
<tr>
<td></td>
<td>HD6433388CP (H8/338)</td>
</tr>
<tr>
<td></td>
<td>HD6413388CP (H8/338)</td>
</tr>
<tr>
<td></td>
<td>HD6473378CP (H8/337)</td>
</tr>
<tr>
<td></td>
<td>HD6433378CP (H8/337)</td>
</tr>
<tr>
<td></td>
<td>HD6413378CP (H8/337)</td>
</tr>
<tr>
<td></td>
<td>HD6433368CP (H8/336)</td>
</tr>
<tr>
<td>HP 64793B (H8/329 emulator)</td>
<td>HD6473298P (H8/329)</td>
</tr>
<tr>
<td></td>
<td>HD6473298C (H8/329)</td>
</tr>
<tr>
<td></td>
<td>HD6433298P (H8/329)</td>
</tr>
<tr>
<td></td>
<td>HD6413298P (H8/329)</td>
</tr>
<tr>
<td></td>
<td>HD6433288P (H8/328)</td>
</tr>
<tr>
<td></td>
<td>HD6473278P (H8/327)</td>
</tr>
<tr>
<td></td>
<td>HD6473278C (H8/327)</td>
</tr>
<tr>
<td></td>
<td>HD6433278P (H8/327)</td>
</tr>
<tr>
<td></td>
<td>HD6413278P (H8/327)</td>
</tr>
<tr>
<td></td>
<td>HD6433268P (H8/326)</td>
</tr>
</tbody>
</table>

Each model provides with an emulation probe designed for its support microprocessors. By replacing the emulation probe, the HP64893 can support processors other than its original support processors. Contact Hewlett-Packard to replace the emulation probe.
Clock Speeds

Maximum clock speed is 10 MHz (system clock).

Emulation Memory

H8/338 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 byte 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 the memory defined as ROM cause emulator execution to break out of target program execution.

Analysis

The H8/338 emulator is used with one of the following analyzer 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 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 H8/338 internal register contents. This includes the ability to modify the program counter (PC) value so you can control where the emulator starts a program run.

Single-Step

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

Breakpoints

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

1-4 Introduction to the H8/338 Emulator
You can also define software breakpoints in your program. The emulator uses one of H8/338 undefined opcodes (5770 hex) as software breakpoint interrupt instruction. When you define a software breakpoint, the emulator places the breakpoint interrupt instruction (5770 hex) at the specified address; after the breakpoint interrupt instruction causes emulator execution to break out of your program, the emulator replaces the original opcode. Refer to the "Using Software Breakpoints" section of "Getting Started" chapter for more information.

**Reset Support**

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

**Real-Time Operation**

Real-time signifies continuous execution of your program without interference from the emulator. (Such interference occurs when the emulator temporarily breaks into the monitor so that it can access register contents or target system memory.) Emulator features performed in real time include: running and analyzer tracing. Emulator features not performed in real time include: display or modify of target system memory; load/dump of any target memory, display or modification of registers, and single step.

---

**Limitations, Restrictions**

**Foreground Monitor**

Foreground monitor is not supported for the H8/338 emulator.

**Sleep and Software Standby Mode**

When the emulator breaks into the emulation monitor, sleep or software standby mode is released.

**Store Condition and Trace**

Disassembling of program execution in the trace list is unreliable when the analyzer is used with store condition. Refer to the "Trace Analysis Considerations" section in Chapter 2.

*Introduction to the H8/338 Emulator 1-5*
Step Command and Interrupts

Step execution cannot be performed in the following cases.

- When the emulator is in the monitor and a suspended interrupt is existed.
- When the emulator is in the monitor and a level sensed interrupt is existed (including interrupts from internal I/O device).

In these cases, step command will fail, and the contents of registers will be the same as that before the issue of the step command.

RAM Enable Bit

The internal RAM of H8/338 processor can be enabled/disabled by RAME (RAM enable bit). However, once you map the internal RAM area to emulation RAM, the emulator still accesses emulation RAM even if the internal RAM is disabled by RAME.
Getting Started

Introduction

This chapter leads you through a basic, step by step tutorial that shows how to use the HP H8/338 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.
- Create and use a command file.
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 set up 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 *Installation/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 H8/338 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 "Memory Modify" emulation command.
.GLOBAL Init,Msgs,Cmd_Input
.GLOBAL Msg_Dest

.SECTION Table,DATA

Msgs
Msg_A .SDATA "THIS IS MESSAGE A"
Msg_B .SDATA "THIS IS MESSAGE B"
Msg_I .SDATA "INVALID COMMAND"
End_Msgs

.SECTION Prog,CODE

;***************************************************************
; Set up the Stack Pointer
;***************************************************************
Init MOV.W #Stack,R7
;***************************************************************
; Clear previous command
;***************************************************************
Clear MOV.B #H'00,R0L
MOV.B R0L,Cmd_Input
;***************************************************************
; Read command input byte. If no command has been
; entered, continue to scan for it.
;***************************************************************
Scan MOV.B @Cmd_Input,R2L
CMP.B #H'00
BEQ Scan
;***************************************************************
; A command has been entered. Check if it is
; command A, command B, or invalid command.
;***************************************************************
Exe_Cmd CMP.B #H'41
BEQ Cmd_A
CMP.B #H'42
BEQ Cmd_B
BRA Cmd_I
;***************************************************************
; * Command A is entered. R3L = the number of bytes
; in message A. R4 = location of the message.
; * Jump to the routine which writes the message.
;***************************************************************
Cmd_A MOV.B #Msg_B-Msg_A,R3L
MOV.W #Msg_A,R4
BRA Write_Msg
;***************************************************************
; * Command B is entered.
;***************************************************************
Cmd_B MOV.B #Msg_I-Msg_B,R3L
MOV.W #Msg_B,R4
BRA Write_Msg
;***************************************************************
; * An invalid command is entered.
;***************************************************************
Cmd_I MOV.B #End_Msgs-Msg_I,R3L
MOV.W #Msg_I,R4

Figure 2-1. Sample Program Listing
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.

Figure 2-1. Sample Program Listing (Cont’d)
Reading Input

The instruction at the Clear label clears any random data or previous commands from the Cmd_Input byte. The Scan loop continually reads the Cmd_Input byte to see if a command is entered (a value other than 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 R3L 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.

Prior to writing the message, Clear_Old clears the destination area. After the message is written, the program branches back to read the next command.
Sample Program Assembly

The sample program is written for and assembled with the HP 64876 H8/300 Assembler/Linkage Editor. For example, the following command was used to assemble the sample program.

\[ \text{C}> \text{h8asm } \text{cmd_rds.src } /\text{debug} \text{<RETURN>} \]

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 the absolute file. The contents of "cmd_rds.k" linkage editor subcommand file is shown in Figure 2-2.

\[ \text{C}> \text{h8lnk } /\text{subcommand=cmd_rds.k}<RETURN> \]

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

```
debug
input cmd_rds
start Table(1100),Prog(1000),Data(0fe80)
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 H8/338 PC Interface by entering the following command from the MS-DOS prompt:

```
pch8338 <emulname>
```

where `<emulname>` is `emul_com1` if your emulator is connected to the COM1 port or `emul_com2` if it is connected to the COM2 port. If you edited the `\hp64700\tables\64700tab` file to change the emulator name, substitute the appropriate name for `<emulname>` in the above command.

In the command above, `pch8338` is the command to start the PC Interface; "<emulname>" is the logical emulator name given in the emulator device table. (To start the version of the PC Interface that supports external timing analysis, substitute `pth8338` for `pch8338` in this command.) If this command is successful, you will see the display shown in Figure 2-3. Otherwise, you will be given an error message and returned to the MS-DOS prompt.

![Figure 2-3. PC Interface Display](image-url)
Selecting PC Interface Commands

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

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

Emulator Status

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

Modifying Configuration

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

```
Config, General
```

Defining the Reset Value for the Stack Pointer

Even though the H8/338 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 0ff80 and press Enter.

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

Selecting your Processor

You need to select the processor you are going to emulate. Use the arrow keys to move the cursor to the "Processor type?" field. Use the TAB key to select the processor you are going to emulate. The default emulator configuration select H8/338 and H8/329 processor, when you use H8/338 and H8/329 emulator, respectively.

2-8 Getting Started
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 H8/338 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 to specify whether a certain range of memory is present in the target system or whether you will be using emulation memory for that address range. You can also specify whether the target system memory is ROM or RAM, and you can specify that emulation memory be treated as ROM or RAM. You can include function code information with address ranges to further characterize the memory block.

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

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

Caution

When you use the H8/338 internal ROM and RAM, you must map memory space where internal ROM and RAM are located as each emulation ROM and RAM.

Which Memory Locations Should Be Mapped?

Typically, assemblers generate relocatable files and linkers combine relocatable files to form the absolute file. The linker load map listing will show what locations your program will occupy in memory. A part of linker load map listing for the sample program (cmd_rds.map) is shown in Figure 2-4.
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 104f hex. The data area, which contains the ASCII values of the messages the program displays, is occupies locations 1100 hex through 1130 hex. The destination area, which contains the command input byte and the locations of the message destination and the stack, occupies locations fe80 hex through ff7f 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.

**Figure 2-4. Sample Program Load Map Listing**

<table>
<thead>
<tr>
<th>SECTION</th>
<th>NAME</th>
<th>START</th>
<th>END</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODULE NAME</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATTRIBUTE : CODE NOSHR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prog</td>
<td>H’00001000 - H’0000104F</td>
<td>H’00000050</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cmd_rds</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>* TOTAL ADDRESS *</td>
<td>H’00001000 - H’0000104F</td>
<td>H’00000050</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATTRIBUTE : DATA NOSHR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Table</td>
<td>H’00001100 - H’00001130</td>
<td>H’00000031</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cmd_rds</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>* TOTAL ADDRESS *</td>
<td>H’00001100 - H’00001130</td>
<td>H’00000031</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATTRIBUTE : DATA SHR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td>H’0000FE80 - H’0000FF7F</td>
<td>H’00000100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cmd_rds</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>* TOTAL ADDRESS *</td>
<td>H’0000FE80 - H’0000FF7F</td>
<td>H’00000100</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2-10 Getting Started
To map memory for the sample program, select:

**Config, Map, Modify**

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

`1000..1fff`

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

`0fe80..0ff7f`

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

---

Figure 2-5. Memory Configuration Display

---

**STATUS**: H8/338—Emulation reset  
**Emulation trace halted**

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

The memory mapper re-assigns blocks of emulation memory after the insertion or deletion of mapper terms. For example, if you modified the contents of fe80 hex through feff hex above, deleted term 1, and displayed locations fe80 hex through feff hex, you would notice the contents of those locations are not the same as they were before deleting the mapper term.

---

**Loading Programs into Memory**

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

```
Memory, Load
```

**File Format**

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

- HP 64876 absolute.
- HP absolute.
- Raw HP64000 absolute.
- Intel hexadecimal.
- Tektronix hexadecimal.
- Motorola S-records.

The HP 64876 absolute file is generated with HP 64876 H8/300 Assembler/Linkage Editor. For this tutorial, choose the HP 64876 format.
**HP 64876 Format:** When you load HP 64876 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 H8/338 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 64876 Format Reader" section in Appendix A for more information.

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

**Memory Type**

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

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

**Force Absolute File Read**

This option is only available for HP 64876 and HP64000 formats. It forces the file format readers to regenerate the emulator absolute file (.hpa) and symbol database (.hps) before loading the code. Normally, these files are only regenerated whenever the file you specify (the output of your language tools) is newer than the emulator absolute file and symbol database.

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 Reference*.

**Displaying Global Symbols**

When you load HP 64876 or HP64000 format absolute files into the emulator, the corresponding symbol database is also loaded.

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

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

To display global symbols, select:

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

The symbols window automatically becomes the active window as a result of this command. You can press <CTRL>z to zoom the window. The resulting display follows.
The global symbols display has two parts. The first part 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-sensitive. For example, if you wish to make a measurement using the symbol `Cmd_Input`, you must specify `Cmd_Input`. The strings `cmd_input` or `CMD_INPUT` are not valid symbol names here.

### Displaying Local Symbols

To display local symbols, select:

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

Enter the name of the module you want to specify (from the first part of the global symbols display; in this case, `cmd_rds`) and press Enter. The resulting display follows.
After you display local symbols with the “System Symbols Local Display” 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 “System Symbols Local Display” command.

Local symbols must be from assembly modules that form the absolute whose symbol database is currently loaded. Otherwise, no symbols will be found (even if the named assembler symbol file exists and contains information).
One thing to note: It is possible for a symbol to be local in one module and global in another, which may result in some confusion. For example, suppose symbol “XYZ” is a global in module A and a local in module B and that these modules link to form the absolute file. After you load the absolute file (and the corresponding symbol database), entering “XYZ” in an expression refers to the symbol from module A. Then, if you display local symbols from module B, entering “XYZ” in an expression refers to the symbol from module B, **not the global symbol**. Now, if you again want to enter “XYZ” to refer to the global symbol from module A, you must display the local symbols from module A (since the global symbol is also local to that module). Loading local symbols from a third module, if it was linked with modules A and B and did not contain an “XYZ” local symbol, would also cause “XYZ” to refer to the global symbol from module A.

**Transfer Symbols to the Emulator**

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

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

Transfer the local symbol information for all modules by entering:

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

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

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

**Memory, Display, Mnemonic**

Enter the address range "1000..1026". (You could also specify this address range using symbols, for example, "Init..Init+26".) The emulation window automatically becomes the active window as a result of this command. You can press <CTRL>z to zoom the memory window. The resulting display follows.

<table>
<thead>
<tr>
<th>Address</th>
<th>Symbol</th>
<th>Mnemonic</th>
</tr>
</thead>
<tbody>
<tr>
<td>01000</td>
<td>Init</td>
<td>MOV.W #ff00,R7</td>
</tr>
<tr>
<td>01004</td>
<td>cmd_rds:Clear</td>
<td>MOV.B #00,RBL</td>
</tr>
<tr>
<td>01006</td>
<td></td>
<td>MOV.B ROL,0Cmd Input</td>
</tr>
<tr>
<td>0100a</td>
<td>cmd_rds:Scan</td>
<td>MOV.B 0Cmd Input,R2L</td>
</tr>
<tr>
<td>0100e</td>
<td></td>
<td>CMP.B #00,R2L</td>
</tr>
<tr>
<td>01010</td>
<td></td>
<td>BEQ cmd_rds:Scan</td>
</tr>
<tr>
<td>01012</td>
<td>cmd_rds:Exe_Cmd</td>
<td>CMP.B #41,R2L</td>
</tr>
<tr>
<td>01014</td>
<td></td>
<td>BEQ cmd_rds:Cmd_A</td>
</tr>
<tr>
<td>01015</td>
<td></td>
<td>CMP.B #42,R2L</td>
</tr>
<tr>
<td>01018</td>
<td></td>
<td>BEQ cmd_rds:Cmd_B</td>
</tr>
<tr>
<td>0101a</td>
<td></td>
<td>BRA cmd_rds:Cmd_I</td>
</tr>
<tr>
<td>0101c</td>
<td>cmd_rds:Cmd_A</td>
<td>MOV.B #11,R3L</td>
</tr>
<tr>
<td>0101e</td>
<td></td>
<td>MOV.W #1100,R4</td>
</tr>
<tr>
<td>01022</td>
<td></td>
<td>BRA cmd_rds:Write_Msg</td>
</tr>
<tr>
<td>01024</td>
<td>cmd_rds:Cmd_B</td>
<td>MOV.B #11,R3L</td>
</tr>
<tr>
<td>01026</td>
<td></td>
<td>MOV.W #1111,R4</td>
</tr>
</tbody>
</table>

If you wish to view the rest of the sample program memory locations, you can select "Memory Display Mnemonic" command again and enter the range "102a..104e".

2-18 Getting Started
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, Step, Address**

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.

```
01010  -  BEQ cmd_rds:Scan
01012  cmd_rds:Exe_Cmd  CMP.B #41,R2L
01014  -  BEQ cmd_rds:Cmd_A
01016  -  CMP.B #42,R2L
01018  -  BEQ cmd_rds:Cmd_B
0101a  -  BRA cmd_rds:Cmd_I
0101c  cmd_rds:Cmd_A  MOV.B #11,R3L
0101e  -  MOV.W #1100,R4
01022  -  BRA cmd_rds:Write_Msg
01024  cmd_rds:Cmd_B  MOV.B #11,R3L
01026  -  MOV.W #1111,R4

01000  Init  MOV.W #ff80,R7
PC = 01004
r0 = 0000  r1 = 0000  r2 = 0000  r3 = 0000
r4 = 0000  r5 = 0000  r6 = 0000  r7 = ff00
pc = 1004 ccr = 00  sp = ff80  mdr = 00
```

STATUS: H0/338—Running in monitor  Emulation trace halted
Window System Register Processor Breakpoints Memory Config Analysis
Go Break Reset CHE Step

Getting Started 2-19
You cannot display registers if the processor is reset. Use the "Processor Break" command to cause the emulator to start executing in the monitor.

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

To continue stepping through the program, you can select:

Processor, Step, Pc

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

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

Specifying a Step Count

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

Processor, Step, Pc

The previous step count is displayed in the "number of instructions" field. You can enter a number from 1 through 99 to specify the number of times to step. Type 5 into the field, and press Enter. The resulting display follows.
When you specify step counts greater than 1, only the last register contents are displayed.

### Modifying Memory

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

**Memory, Modify, Bytes**

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** (fe80 hex), select:

**Memory, Display, Bytes**
Type the address "0fe80" or the symbol **Cmd_Input**, and press **Enter**. This command will automatically activate the memory window. The resulting display is shown below.

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

---

**Running the Program**

To start the emulator executing the sample program, select:

- **Processor**, **Go**, **PC**

The status line will show that the emulator is "Running user program".
Searching Memory for Data

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

Memory, Find

Enter the range of the memory locations to be searched, 0fe82 hex through 0fea1 hex, and enter the data 20 hex and 41 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 and 42 hex), or you can modify the command input byte to "C" and search for "IN" (49 hex and 4e hex).

Breaking into the Monitor

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

Processor, Break

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

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

Software breakpoints are implemented in the H8/338 emulator by replacing opcodes with one of undefined opcodes (5770 hex) as software breakpoint instruction. In the following explanation, we call this code as special instruction. In the H8/338 emulator, software breakpoints are implemented by replacing opcodes with the special instruction. When you set a software breakpoint, the emulator replaces the opcode at the address specified with the special instruction. When the emulator executes this instruction in the user program, execution breaks to the monitor.

If the special instruction (undefined opcode, 5770 hex) was not inserted as the result of a "Breakpoints" command (in other words, it is part of the user program), the "Undefined software breakpoint" message is displayed above the status line. 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 (5770 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).
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 (102C hex), select:

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

You can add multiple breakpoints in a single command by separating each one with a semicolon. For example, you could type "1012;101C;1032" to set three breakpoints.

Run the program by selecting:

- **Processor, Go, Pc**
  The status line shows that the emulator is running the user program. Modify the command input byte to an invalid command by selecting:

- **Memory, Modify, Bytes**
  Enter an invalid command, such as "Cmd_Input=75". The following messages result:

  | ALERT: Software breakpoint: 0102C |
  | STATUS: H8/338--Running in monitor |

To continue program execution, select:

- **Processor, Go, Pc**
Displaying Software Breakpoints

To view the status of the breakpoint, select:

Breakpoints, Display

The resulting display shows that the breakpoint has been cleared.

Setting a Software Breakpoint

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

Breakpoints, Set, Single

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

Clearing a Software Breakpoint

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

Breakpoints, Clear, Single

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

The H8/338 emulation analyzer has 48 trace signals which monitor internal emulation lines (address, data, and status lines). Optionally, you may have an additional 16 trace signals which monitor external input lines. The analyzer collects data at each pulse of a clock signal, and saves the data (a trace state) if it meets a "storage qualification" condition.

Note

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

Resetting the Analysis Specification

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

Analysis, Trace, Reset

Specifying a Simple Trigger

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

Analysis, Trace, Modify

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

You'll enter the pattern expression menu. Press the up arrow key until the addr field directly opposite the pattern a= is highlighted. Type the address of the command input byte, using either the global symbol Cmd_Input or address 0fe80, 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.

Now the "Status" field is highlighted. Use the TAB key to view the status qualifiers which may be entered.
H8/338 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.

<table>
<thead>
<tr>
<th>Qualifier</th>
<th>Status Bits (32..44)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>backgrnd</td>
<td>0 xxxx xxxx xxxxB</td>
<td>Background cycle</td>
</tr>
<tr>
<td>byte</td>
<td>x 1xxx xxxx xx1xB</td>
<td>Byte access</td>
</tr>
<tr>
<td>foregrnd</td>
<td>1 xxxx xxxx xxxxB</td>
<td>Foreground cycle</td>
</tr>
<tr>
<td>grd</td>
<td>x 10xx xxxx xxxxB</td>
<td>Guarded memory access</td>
</tr>
<tr>
<td>ifetch</td>
<td>x 1xxx xxxx 1101B</td>
<td>Fetch from internal ROM</td>
</tr>
<tr>
<td>intack</td>
<td>x xxx0 xxxx xxxxB</td>
<td>Interrupt acknowledge cycle</td>
</tr>
<tr>
<td>io</td>
<td>x 1xxx xxxx 0xxxB</td>
<td>Internal I/O access</td>
</tr>
<tr>
<td>memory</td>
<td>x 1xxx xxxx 1xxxB</td>
<td>Memory access</td>
</tr>
<tr>
<td>read</td>
<td>x 1xxx xxxx xxxx1B</td>
<td>Read cycle</td>
</tr>
<tr>
<td>word</td>
<td>x 1xxx xxxx xx0xB</td>
<td>Word access</td>
</tr>
<tr>
<td>write</td>
<td>x 1xxx xxxx xxxxOB</td>
<td>Write cycle</td>
</tr>
<tr>
<td>wrrom</td>
<td>x 1x0x xxxx xxxxOB</td>
<td>Write to ROM cycle</td>
</tr>
</tbody>
</table>

Select the read status and press Enter. Figure 2-6 and 2-7 shows 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 trigger apec field. Press Enter to exit the trace specification.

![Internal State Trace Specification]

2-28 Getting Started
To start the trace, select:

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

**Memory, Modify, Bytes**
Enter "Cmd_Input=42". The status line now shows that the trace is complete.

To display the trace, select:

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

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

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

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

To list the next lines of the trace, press the PgDn or Next key.
The resulting display shows the \texttt{Cmd\_B} instructions and the branch to \texttt{Write\_Msg} and the beginning of the instructions which move the "THIS IS MESSAGE B" message to the destination locations.

\textbf{For a Complete Description}

For a complete description of using the HP 64700 Series analyzer with the PC Interface, refer to the \textit{HP 64700 Emulators PC Interface: Analyzer User's Guide}.
Trace Analysis Considerations

There are some points to be noticed when you use the emulation analyzer of H8/338 emulator.

How to Specify Trigger Condition

You need to be careful to specify the condition on which the emulation analyzer should start the trace. Suppose you would like to start the trace when the program begins executing `Exe_Cmd` routine. To reset the analysis specification:

```
Analysis, Trace, Reset
```

Select the following command and modify the analysis specification as shown in the following displays.

```
Analysis, Trace, Modify
```

![Internal State Trace Specification](image)

**Status:**
- H8/338—Running user program
- Emulation trace complete

**Tab** selects a pattern or press ENTER to modify this field and the pattern values.
Start the trace and modify memory so that the program execution jumps to the \texttt{Exe_Cmd} routine:

\begin{verbatim}
Analysis, Begin
Memory, Modify, Bytes
Enter "Cmd_Input=41".
\end{verbatim}

Now display the trace:

\begin{verbatim}
Analysis, Display
Press \texttt{Enter} three times.
\end{verbatim}
This is not what we were expecting to see. As you can see at the first line of the trace, the address of **Exe_Cmd** routine appears on the address bus during the program executing **Scan** loop. This made the analyzer start trace. To avoid mis-trigger by this cause, set the trigger condition to the second instruction of the routine you want to trace:

```
Analysis, Trace, Modify
```

Enter the pattern specification display, and modify the address specification to "Exe_Cmd+2".

To start the trace and complete the measurement:

```
Analysis, Begin
Memory, Modify, Bytes
```

Enter "Cmd_Input=41".

```
Analysis, Display
```

Press Enter three times.
As you can see, the analyzer captured the execution of \texttt{Exe_Cmd} routine. If you need to see the execution of the instruction at the \texttt{Exe_Cmd} label, set the trigger position to "Center" when you modify the analysis specification.

When you specify store condition, disassembling of program execution is unreliable.

When you specify store condition, disassembling of program execution is unreliable.

\textbf{Analysis, Trace, Reset}  
\textbf{Analysis, Begin}  
\textbf{Analysis, Display}  
Press \texttt{Enter} three times. You will see a display similar to the following.
The program is executing the Scan loop.

To trace only accesses to the address range Init through Init+0ffh:

Analysis, Trace, Modify

Modify the trace specification as shown in the following displays.
Internal State Trace Specification

1 While storing any state
   Trigger on any state 1 times

2 Store

Branches off  Count  time  Prestore off  Trigger position
+11+ : Interfield movement  Ctrl ++ : Field editing  TAB : Scroll choices

STATUS: M87/338—Running user program  Emulation trace complete

TAB selects a pattern or press ENTER to modify this field and the pattern value.

Internal State Trace Specification

Set 1

Range (>) Label addr = Init thru Init+Off
Pat addr data stat
a b c d

Set 2

Expression

Expressions have the form: <set1> and/or <set2>. Where set1 consists of <a, b, c, d, r, t> 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:

STATUS: M87/338—Running user program  Emulation trace complete

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

Getting Started 2-37
Start the trace and display the trace listing:

\textbf{Analysis, Begin}  \\
\textbf{Analysis, Display}  \\
Press Enter three times. You will see a display similar to the following.

![Analysis display](image)

As you can see, the executions of CMP.B instruction are not disassembled. This occurs when the analyzer cannot get necessary information because of the store condition. Be careful when you use the analyzer with store condition.

\textbf{Triggering the Analyzer by Data}

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

There are some points to be noticed when you trigger the analyzer in this way. You always need to specify the "Data" with 16 bits value even when access to the data is performed by byte access. This is because the analyzer is designed so that it can capture data on internal data bus (which has 16 bits width). The following table shows the way to specify the trigger condition by data.
For example, to trigger the analyzer when the processor performs word access to data 1234 hex in internal ROM, you can use 1234h, 12xxh, and 0xx34h as the "Data" specification.

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

Notice that you always need to specify "xx" as the lower 8 bits value to capture byte access of the processor. Be careful to trigger the analyzer by data.

### 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 maps memory, modifies configuration and loads program into memory for the sample program. To create such a command file:

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

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

---

<table>
<thead>
<tr>
<th>Location of data</th>
<th>Access</th>
<th>&quot;Data&quot; Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal ROM, RAM</td>
<td>word access</td>
<td>hhll *1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hhxx *2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>xxll *2</td>
</tr>
<tr>
<td></td>
<td>byte access</td>
<td>ddxx *2</td>
</tr>
<tr>
<td>Others</td>
<td>byte access *3</td>
<td>ddxx</td>
</tr>
</tbody>
</table>

*1 hhll means 16 bits data
*2 dd, hh, ll mean 8 bits data
*3 H8/338 processor performs word access (MOV.W etc..) to external memory and internal I/O by two byte accesses.
First, to map the memory:

**Config, Map, Memory**
Map 1000 hex through 1fff hex to erom and fe80 hex through ff7f hex to eram. (As shown in Figure 2-5.)

To set up the reset value for the stack pointer:

**Config, General**
Use the arrow keys to move the cursor to the "Reset value for Stack Pointer" field, type 0ff80h, and press Enter.

To load the program into memory:

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

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

**System, Log, Input, Disable**
The command file cmd_rds.cmd will no longer accept command input. The file looks like this:

```
cmm
@tram
@1000H..2FFFH
@erom
@0FE00H..0FEFFH
@ram
@Empty
@grad
@Empty
@grad
@Empty
@grad
@Empty
@grad
@Empty
@grad
@Empty
@grad
@Empty
@grad
@Empty
@grad
@Empty
@grad
@Empty
@grad
@Empty
@grad
```

2-40 Getting Started
You can see a @symbol in front of some lines in the file. These represents data values, as opposed to commands.

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

System, Command_file
Enter "cmd_rds.cmd", press Enter. Watch the command file commands execute. As you can see, the sequence of commands you entered is automatically executed.

### Resetting the Emulator

To reset the emulator, select:

Processor, Reset, Hold
The emulator is held in a reset state (suspended) until a "Processor Break", "Processor Go", or "Processor Step" command is entered. A CMB execute signal will also cause the emulator to run if reset.

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

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

**System, Exit, Locked**

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

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

**System, Exit, Unlocked**
Using the H8/338 Emulator In-Circuit

When you are ready to use the H8/338 Emulator in conjunction with actual target system hardware, there are some special considerations you should keep in mind.

- installing the emulator probe
- properly configure the emulator

We will cover the first topic in this chapter. For complete details on in-circuit emulation configuration, refer to Chapter 4.
Installing the Target System Probe

Caution

The following precautions should be taken while using the H8/338 Emulator. Damage to the emulator circuitry may result if these precautions are not observed.

Power Down Target System. Turn off power to the user target system and to the H8/338 Emulator before inserting the user plug to avoid circuit damage resulting from voltage transients or mis-insertion of the user plug.

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

Protect Against Static Discharge. The H8/338 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 H8/338 Emulator; when powering down, turn off the emulator first, then turn off power to the target system.

Pin Guard

The H8/338/329 emulator is shipped with a pin guard to prevent impact damage to the target system probe pins. The guard should be left in place while you are not using the emulator.

3-2 In-Circuit Emulation
H8/338 Emulator

H8/338 emulator is shipped with a non-conductive pin guard over the target system probe.

H8/329 Emulator

H8/329 emulator is shipped with a conductive plastic pin guard over the target system probe pins. When you do use the emulator, either for normal emulation tasks, or to run performance verification on the emulator, you must remove this conductive pin guard to avoid intermittent failures due to the target system probe lines being shorted together.

Pin Protector (H8/329 Only)

The target system probe of the H8/329 emulator has a pin protector that prevents damage to the probe when inserting and removing the probe from the target system microprocessor socket. Do not use the probe without a pin protector installed. If the target system probe is installed on a densely populated circuit board, there may not be enough room for the plastic shoulders of the probe socket. If this occurs, another pin protector may be stacked onto the existing pin protector.

Installing the Target System Probe

1. Remove the H8/338 microprocessor from the target system socket. Note the location of pin 1 on the processor and on the target system socket.

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

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

4.
When you are using the H8/338 emulator, we recommend that you use ITT CANNON "LCS-84" series 84 pin PLCC socket to make sure the contact between emulator probe and target system microprocessor socket.

Figure 3-1. Installing the Probe (H8/338 emulator)
While the emulator is running the background monitor, probe pins are in the following state.

<table>
<thead>
<tr>
<th>Pin</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Bus</td>
<td>Same as foreground</td>
</tr>
<tr>
<td>Data Bus</td>
<td>Always high impedance otherwise you direct the emulator to modify target memory.</td>
</tr>
<tr>
<td>AS</td>
<td>Same as foreground</td>
</tr>
<tr>
<td>RD</td>
<td>Same as foreground</td>
</tr>
<tr>
<td>WR</td>
<td>Always high otherwise you direct the emulator to modify target memory.</td>
</tr>
<tr>
<td>Others</td>
<td>Same as foreground</td>
</tr>
</tbody>
</table>
Target System Interface (H8/338)

MD1 MD0
RES NMI
STBY

These signals are connected to 74HCT14 through 51 ohm series resister and 10K ohm pull-up resister.

P1(7:0) P5(2:0) These signals are connected to H8/338 emulation processor through 51 ohm series resister and 10K ohm pull-up resister.
P2(7:0) P6(7:0)
P3(7:0) P8(6:0)
P4(7:0) P9(6:0)

3-6 In-Circuit Emulation
This signal are connected to H8/338 emulation processor and GAL20V8 through 51 ohm series resister.

These signals are connected to H8/338 emulation processor through 51 ohm series resister.
Target System Interface (H8/329)

These signals are connected to 74HCT14 through 51 ohm series resistor and 10K ohm pull-up resistor.

These signals are connected to H8/329 emulation processor through 51 ohm series resistor and 10K ohm pull-up resistor.

3-8 In-Circuit Emulation
P47  This signal are connected to H8/329 emulation processor and GAL20V8 through 51 ohm resistor.

P7(7:0)  These signals are connected to H8/329 emulation processor through 51 ohm series resister.
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. To specify a run from reset state, select:

Processor, Go, Reset

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

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

Introduction

The H8/338 emulator is designed to help you in all stages of target system development. For instance, you can run the emulator out-of-circuit when developing and debugging your target system software and in-circuit when integrating your target system software with hardware. You can use the emulator’s internal clock or your target system clock. Emulation memory can be used along with your target system memory, and it can be mapped as RAM or ROM. And, there are many more options available.

The emulator is a flexible instrument and may be configured to suit your needs at any stage of the development process. This chapter describes the options available when configuring your 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.
To enter the general configuration menu, Select:

**Config, General**
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.

**General Emulation Configuration**

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal emulator clock?</td>
<td>[0]</td>
</tr>
<tr>
<td>Enable real-time mode?</td>
<td>[1]</td>
</tr>
<tr>
<td>Enable breaks on write to ROM?</td>
<td>[1]</td>
</tr>
<tr>
<td>Enable software breakpoints?</td>
<td>[1]</td>
</tr>
<tr>
<td>Enable CMB interaction?</td>
<td>[1]</td>
</tr>
<tr>
<td>Enable /RES input from target?</td>
<td>[1]</td>
</tr>
<tr>
<td>Processor type</td>
<td>8088</td>
</tr>
<tr>
<td>Processor operation mode</td>
<td>8087</td>
</tr>
<tr>
<td>Reset value for Stack Pointer</td>
<td>0</td>
</tr>
</tbody>
</table>

+↓↑+: Interfield movement  Ctrl + : Field editing  TAB : Scroll choices

**STATUS:** HP/338—Emulation reset  Emulation trace halted
Specify the value that the stack pointer is set to whenever the monitor is entered after an emulation reset. The stack pointer must be set to an even address.

**Figure 4-1. Emulator Configuration Display**

4-2 Configuring the Emulator
## Internal Emulator Clock?

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

<table>
<thead>
<tr>
<th>Yes</th>
<th>Selects the internal clock oscillator as the emulator clock source. The emulators’ internal clock speed is 10 MHz (system clock).</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Selects the clock input to the emulator probe from the target system. You must use a clock input conforming to the specifications for the H8/338 microprocessor. The maximum clock speed is 10 MHz (system clock).</td>
</tr>
</tbody>
</table>

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

## Enable Real-Time Mode?

The "restrict to real-time" question lets you configure the emulator so that commands which cause the emulator to break to monitor and return to the user program are refused.

<table>
<thead>
<tr>
<th>No</th>
<th>All commands, regardless of whether or not they require a break to the emulation monitor, are accepted by the emulator.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>When runs are restricted to real-time and the emulator is running the user program, all commands that cause a break (except &quot;Processor Reset&quot;, &quot;Processor Break&quot;, &quot;Processor Go&quot;, and &quot;Processor Step&quot;) are refused. For example, the following commands are not allowed when runs are restricted to real-time:</td>
</tr>
</tbody>
</table>
Display/modify registers.
Display/modify target system memory.
Display/modify internal I/O registers.
Load/store target system memory.

Caution

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

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.

4-4 Configuring the Emulator
Enable Software Breakpoints?

This question allows you to enable or disable the software breakpoints feature. The H8/338 emulator uses undefined opcode (5770 hex) as software breakpoint.

**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**
Allows you to use the software breakpoints feature. When you set a software breakpoint, an undefined opcode (5770 hex) will be placed at the address specified. When the opcode 5770 hex is executed, program execution will break into the monitor.

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

Enable CMB Interaction?

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

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

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

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

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

---

**Note**

CMB interaction will also be enabled when the Processor, CMB, Execute command is entered.

---

**Enable /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. Edge sensed interrupts are suspended while running the background monitor, and such interrupts will occur when context is changed to foreground. Level sensed interrupts and internal interrupts are ignored during in background operation.
**Enable / RES Input from Target?**

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

While running the background monitor, the emulator ignores /RES signal except that the emulator’s status is "Awaiting target reset". (see the "Running the Emulation from Target Reset" section in the "In-Circuit Emulation" chapter).

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>The emulator will respond to /RES input during foreground operation.</td>
</tr>
<tr>
<td>No</td>
<td>The emulator will not respond to /RES input from the target system.</td>
</tr>
</tbody>
</table>

**Processor Type? (H8/338 emulator)**

This configuration defines the processor to be emulated by the H8/338 emulator.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>338</td>
<td>When you are going to emulate H8/338 microprocessor, select this item. (The default emulation configuration selects this.)</td>
</tr>
<tr>
<td>337</td>
<td>When you are going to emulate H8/337 microprocessor, select this item.</td>
</tr>
<tr>
<td>336</td>
<td>When you are going to emulate H8/336 microprocessor, select this item.</td>
</tr>
</tbody>
</table>
## Processor Type? (H8/329 emulator)

This configuration defines the processor to be emulated by the H8/329 emulator.

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>329</td>
<td>When you are going to emulate H8/329 microprocessor, select this item. (The default emulation configuration selects this.)</td>
</tr>
<tr>
<td>328</td>
<td>When you are going to emulate H8/328 microprocessor, select this item.</td>
</tr>
<tr>
<td>327</td>
<td>When you are going to emulate H8/327 microprocessor, select this item.</td>
</tr>
<tr>
<td>326</td>
<td>When you are going to emulate H8/326 microprocessor, select this item.</td>
</tr>
</tbody>
</table>

## Processor Operation Mode?

This configuration defines operation mode in which the emulator works.

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ext</td>
<td>The emulator will work using the mode setting by the target system. The target system must supply appropriate input to MD0 and MD1. If you are using the emulator out of circuit when &quot;external&quot; is selected, the emulator will operate in mode 3.</td>
</tr>
<tr>
<td>&lt;mode_num&gt;</td>
<td>When &lt;mode_num&gt; is selected, the emulator will operate in selected mode regardless of the mode setting by the target system.</td>
</tr>
</tbody>
</table>

4-8 Configuring the Emulator
Valid <mode_num> are following:

<table>
<thead>
<tr>
<th>&lt;mode_num&gt;</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The emulator will operate in mode 1. (expanded mode without internal ROM)</td>
</tr>
<tr>
<td>2</td>
<td>The emulator will operate in mode 2. (expanded mode with internal ROM)</td>
</tr>
<tr>
<td>3</td>
<td>The emulator will operate in mode 3. (single chip mode)</td>
</tr>
</tbody>
</table>

**Reset Value for Stack Pointer?**

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

---

Storing an Emulator Configuration

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

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

To store the current emulator configuration, select:

`Config, Store`

Enter the name of file to which the emulator configuration will be saved.
Loading an Emulator Configuration

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

```
Config, Load
```

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

4-12 Configuring the Emulator
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 the H8/338 internal I/O 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.

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.

5-2 Using the Emulator
Running the Emulator at /EXECUTE

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

Config, General

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

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

Processor, CMB, Go

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

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

Breaking on the Analyzer Trigger

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

Config, Trigger

The trigger configuration display contains two diagrams, one for each of the internal TRIG1 and TRIG2 signals.
To use the internal TRIG1 signal to connect the analyzer trigger to the emulator break line, move the cursor to the highlighted "Analyzer" field in the TRIG1 portion of the display, and use the Tab key to select the "----->>" arrow which shows that the analyzer is driving TRIG1. Next, move the cursor to the highlighted "Emulator" field and use the Tab key to select the arrow pointing towards the emulator (<<-----); this specifies that emulator execution will break into the monitor when the TRIG1 signal is driven. The trigger configuration display is shown in figure 5-1.

Figure 5-1. Cross Trigger Configuration

Note

If your emulator is not configured with external analyzer, the "Timing" cross trigger option is not displayed.

5-4 Using the Emulator
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.

\textbf{Memory, Store}

\underline{Note}

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

\underline{Caution}

The "Memory Store" command writes over an existing file if it has the same name that is specified with the command. You may wish to verify beforehand that the specified filename does not already exist.
Register Classes and Names (H8/338 Emulator)

The following register classes and names are used with the display/modify registers commands in H8/338 emulator.

**basic (*) class**

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pc</td>
<td>Program counter</td>
</tr>
<tr>
<td>ccr</td>
<td>Condition code register</td>
</tr>
<tr>
<td>r0</td>
<td>Register 0</td>
</tr>
<tr>
<td>r1</td>
<td>Register 1</td>
</tr>
<tr>
<td>r2</td>
<td>Register 2</td>
</tr>
<tr>
<td>r3</td>
<td>Register 3</td>
</tr>
<tr>
<td>r4</td>
<td>Register 4</td>
</tr>
<tr>
<td>r5</td>
<td>Register 5</td>
</tr>
<tr>
<td>r6</td>
<td>Register 6</td>
</tr>
<tr>
<td>r7</td>
<td>Register 7</td>
</tr>
<tr>
<td>sp</td>
<td>Stack pointer</td>
</tr>
<tr>
<td>mdcr</td>
<td>Mode control register *1</td>
</tr>
</tbody>
</table>

**sys class** (System control registers)

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stcr</td>
<td>Serial timer control register</td>
</tr>
<tr>
<td>syscr</td>
<td>System control register</td>
</tr>
<tr>
<td>mdcr</td>
<td>Mode control register</td>
</tr>
<tr>
<td>iscr</td>
<td>IRQ sense control register *1</td>
</tr>
<tr>
<td>ier</td>
<td>IRQ enable register</td>
</tr>
</tbody>
</table>
**port class** (I/O port)

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1ddr</td>
<td>Port 1 data direction register *2</td>
</tr>
<tr>
<td>p2ddr</td>
<td>Port 2 data direction register *2</td>
</tr>
<tr>
<td>p3ddr</td>
<td>Port 3 data direction register *2</td>
</tr>
<tr>
<td>p4ddr</td>
<td>Port 4 data direction register *2</td>
</tr>
<tr>
<td>p5ddr</td>
<td>Port 5 data direction register *2</td>
</tr>
<tr>
<td>p6ddr</td>
<td>Port 6 data direction register *2</td>
</tr>
<tr>
<td>p8ddr</td>
<td>Port 8 data direction register *2</td>
</tr>
<tr>
<td>p9ddr</td>
<td>Port 9 data direction register *2</td>
</tr>
<tr>
<td>p1dr</td>
<td>Port 1 data register</td>
</tr>
<tr>
<td>p2dr</td>
<td>Port 2 data register</td>
</tr>
<tr>
<td>p3dr</td>
<td>Port 3 data register</td>
</tr>
<tr>
<td>p4dr</td>
<td>Port 4 data register</td>
</tr>
<tr>
<td>p5dr</td>
<td>Port 5 data register</td>
</tr>
<tr>
<td>p6dr</td>
<td>Port 6 data register</td>
</tr>
<tr>
<td>p7dr</td>
<td>Port 7 data register *1</td>
</tr>
<tr>
<td>p8dr</td>
<td>Port 8 data register</td>
</tr>
<tr>
<td>p9dr</td>
<td>Port 9 data register</td>
</tr>
<tr>
<td>p1pcr</td>
<td>Port 1 input pull up MOS control register</td>
</tr>
<tr>
<td>p2pcr</td>
<td>Port 2 input pull up MOS control register</td>
</tr>
<tr>
<td>p3pcr</td>
<td>Port 3 input pull up MOS control register</td>
</tr>
</tbody>
</table>

**frt class** (16 bit free running timer)

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tier</td>
<td>Timer interrupt enable register</td>
</tr>
<tr>
<td>frtcsr</td>
<td>Timer control/status register</td>
</tr>
<tr>
<td>frc</td>
<td>Free running counter</td>
</tr>
<tr>
<td>ocra</td>
<td>Output compare register A</td>
</tr>
<tr>
<td>ocrb</td>
<td>Output compare register B</td>
</tr>
<tr>
<td>frtcr</td>
<td>Timer control register</td>
</tr>
<tr>
<td>tocr</td>
<td>Timer output compare control register</td>
</tr>
<tr>
<td>icra</td>
<td>Input capture register A *1</td>
</tr>
<tr>
<td>icrb</td>
<td>Input capture register B *1</td>
</tr>
<tr>
<td>icrc</td>
<td>Input capture register C *1</td>
</tr>
<tr>
<td>icrd</td>
<td>Input capture register D *1</td>
</tr>
</tbody>
</table>

---

*Using the Emulator 5-7*
**tmr0 class** (8 bit timer 0)

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcr0</td>
<td>Timer control register</td>
</tr>
<tr>
<td>tcsr0</td>
<td>Timer control/status register</td>
</tr>
<tr>
<td>tcora0</td>
<td>Timer constant register A</td>
</tr>
<tr>
<td>tcorb0</td>
<td>Timer constant register B</td>
</tr>
<tr>
<td>tcnt0</td>
<td>Timer counter</td>
</tr>
</tbody>
</table>

**tmr1 class** (8 bit timer 1)

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcr1</td>
<td>Timer control register</td>
</tr>
<tr>
<td>tcsr1</td>
<td>Timer control/status register</td>
</tr>
<tr>
<td>tcora1</td>
<td>Timer constant register A</td>
</tr>
<tr>
<td>tcorb1</td>
<td>Timer constant register B</td>
</tr>
<tr>
<td>tcnt1</td>
<td>Timer counter</td>
</tr>
</tbody>
</table>

**pwm0 class** (PWM timer 0)

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pwmtcr0</td>
<td>Timer control register</td>
</tr>
<tr>
<td>dtr0</td>
<td>Duty register</td>
</tr>
<tr>
<td>pwmtcnt0</td>
<td>Timer counter</td>
</tr>
</tbody>
</table>

**pwm1 class** (PWM timer 1)

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pwmtcr1</td>
<td>Timer control register</td>
</tr>
<tr>
<td>dtr1</td>
<td>Duty register</td>
</tr>
<tr>
<td>pwmtcnt1</td>
<td>Timer counter</td>
</tr>
</tbody>
</table>

5-8 Using the Emulator
**sci0 class**  (Serial communication interface 0)

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>smr0</td>
<td>Serial mode register</td>
</tr>
<tr>
<td>brr0</td>
<td>Bit rate register</td>
</tr>
<tr>
<td>scr0</td>
<td>Serial control register</td>
</tr>
<tr>
<td>tdr0</td>
<td>Transmit data register</td>
</tr>
<tr>
<td>ssr0</td>
<td>Serial status register</td>
</tr>
<tr>
<td>rdr0</td>
<td>Receive data register *1</td>
</tr>
</tbody>
</table>

**sci1 class**  (Serial communication interface 1)

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>smr1</td>
<td>Serial mode register</td>
</tr>
<tr>
<td>brr1</td>
<td>Bit rate register</td>
</tr>
<tr>
<td>scr1</td>
<td>Serial control register</td>
</tr>
<tr>
<td>tdr1</td>
<td>Transmit data register</td>
</tr>
<tr>
<td>ssr1</td>
<td>Serial status register</td>
</tr>
<tr>
<td>rdr1</td>
<td>Receive data register *1</td>
</tr>
</tbody>
</table>

**adc class**  (A/D converter)

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addra</td>
<td>A/D data register A *1</td>
</tr>
<tr>
<td>addrb</td>
<td>A/D data register B *1</td>
</tr>
<tr>
<td>addrc</td>
<td>A/D data register C *1</td>
</tr>
<tr>
<td>addrd</td>
<td>A/D data register D *1</td>
</tr>
<tr>
<td>adcsr</td>
<td>A/D control/status register</td>
</tr>
<tr>
<td>adcr</td>
<td>A/D control register</td>
</tr>
</tbody>
</table>

*1 indicates marked register.
### dac class (D/A converter)

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dadr0</td>
<td>D/A data register 0</td>
</tr>
<tr>
<td>dadr1</td>
<td>D/A data register 1</td>
</tr>
<tr>
<td>dacr</td>
<td>D/A control register</td>
</tr>
</tbody>
</table>

### NO CLASS
The following register names are not included in any register class.

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>r0h</td>
<td>Register 0 H</td>
</tr>
<tr>
<td>r0l</td>
<td>Register 0 L</td>
</tr>
<tr>
<td>r1h</td>
<td>Register 1 H</td>
</tr>
<tr>
<td>r1l</td>
<td>Register 1 L</td>
</tr>
<tr>
<td>r2h</td>
<td>Register 2 H</td>
</tr>
<tr>
<td>r2l</td>
<td>Register 2 L</td>
</tr>
<tr>
<td>r3h</td>
<td>Register 3 H</td>
</tr>
<tr>
<td>r3l</td>
<td>Register 3 L</td>
</tr>
<tr>
<td>r4h</td>
<td>Register 4 H</td>
</tr>
<tr>
<td>r4l</td>
<td>Register 4 L</td>
</tr>
<tr>
<td>r5h</td>
<td>Register 5 H</td>
</tr>
<tr>
<td>r5l</td>
<td>Register 5 L</td>
</tr>
<tr>
<td>r6h</td>
<td>Register 6 H</td>
</tr>
<tr>
<td>r6l</td>
<td>Register 6 L</td>
</tr>
<tr>
<td>r7h</td>
<td>Register 7 H</td>
</tr>
<tr>
<td>r7l</td>
<td>Register 7 L</td>
</tr>
</tbody>
</table>

*1 Display only  
*2 Modification only

5-10 Using the Emulator
The following register classes and names are used with the display/modify registers commands in H8/329 emulator.

### basic (*) class

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pc</td>
<td>Program counter</td>
</tr>
<tr>
<td>ccr</td>
<td>Condition code register</td>
</tr>
<tr>
<td>r0</td>
<td>Register 0</td>
</tr>
<tr>
<td>r1</td>
<td>Register 1</td>
</tr>
<tr>
<td>r2</td>
<td>Register 2</td>
</tr>
<tr>
<td>r3</td>
<td>Register 3</td>
</tr>
<tr>
<td>r4</td>
<td>Register 4</td>
</tr>
<tr>
<td>r5</td>
<td>Register 5</td>
</tr>
<tr>
<td>r6</td>
<td>Register 6</td>
</tr>
<tr>
<td>r7</td>
<td>Register 7</td>
</tr>
<tr>
<td>sp</td>
<td>Stack pointer</td>
</tr>
<tr>
<td>mdcr</td>
<td>Mode control register *1</td>
</tr>
</tbody>
</table>

### sys class (System control registers)

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stcr</td>
<td>Serial timer control register</td>
</tr>
<tr>
<td>syscr</td>
<td>System control register</td>
</tr>
<tr>
<td>mdcr</td>
<td>Mode control register</td>
</tr>
<tr>
<td>iscr</td>
<td>IRQ sense control register *1</td>
</tr>
<tr>
<td>ier</td>
<td>IRQ enable register</td>
</tr>
</tbody>
</table>
### port class (I/O port)

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1ddr</td>
<td>Port 1 data direction register *2</td>
</tr>
<tr>
<td>p2ddr</td>
<td>Port 2 data direction register *2</td>
</tr>
<tr>
<td>p3ddr</td>
<td>Port 3 data direction register *2</td>
</tr>
<tr>
<td>p4ddr</td>
<td>Port 4 data direction register *2</td>
</tr>
<tr>
<td>p5ddr</td>
<td>Port 5 data direction register *2</td>
</tr>
<tr>
<td>p6ddr</td>
<td>Port 6 data direction register *2</td>
</tr>
<tr>
<td>p7ddr</td>
<td>Port 7 data direction register *2</td>
</tr>
<tr>
<td>p1dr</td>
<td>Port 1 data register</td>
</tr>
<tr>
<td>p2dr</td>
<td>Port 2 data register</td>
</tr>
<tr>
<td>p3dr</td>
<td>Port 3 data register</td>
</tr>
<tr>
<td>p4dr</td>
<td>Port 4 data register</td>
</tr>
<tr>
<td>p5dr</td>
<td>Port 5 data register</td>
</tr>
<tr>
<td>p6dr</td>
<td>Port 6 data register</td>
</tr>
<tr>
<td>p7dr</td>
<td>Port 7 data register *1</td>
</tr>
<tr>
<td>p1pcr</td>
<td>Port 1 input pull up MOS control register</td>
</tr>
<tr>
<td>p2pcr</td>
<td>Port 2 input pull up MOS control register</td>
</tr>
<tr>
<td>p3pcr</td>
<td>Port 3 input pull up MOS control register</td>
</tr>
</tbody>
</table>

### frt class (16 bit free running timer)

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tier</td>
<td>Timer interrupt enable register</td>
</tr>
<tr>
<td>frtcsr</td>
<td>Timer control/status register</td>
</tr>
<tr>
<td>frc</td>
<td>Free running counter</td>
</tr>
<tr>
<td>ocra</td>
<td>Output compare register A</td>
</tr>
<tr>
<td>ocrb</td>
<td>Output compare register B</td>
</tr>
<tr>
<td>frtcr</td>
<td>Timer control register</td>
</tr>
<tr>
<td>tocr</td>
<td>Timer output compare control register</td>
</tr>
<tr>
<td>icra</td>
<td>Input capture register A *1</td>
</tr>
<tr>
<td>icrb</td>
<td>Input capture register B *1</td>
</tr>
<tr>
<td>icrc</td>
<td>Input capture register C *1</td>
</tr>
<tr>
<td>icrd</td>
<td>Input capture register D *1</td>
</tr>
</tbody>
</table>

5-12 Using the Emulator
### tmr0 class (8 bit timer 0)

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcrl</td>
<td>Timer control register</td>
</tr>
<tr>
<td>tcsr0</td>
<td>Timer control/status register</td>
</tr>
<tr>
<td>tcora0</td>
<td>Timer constant register A</td>
</tr>
<tr>
<td>tcorb0</td>
<td>Timer constant register B</td>
</tr>
<tr>
<td>tcnt0</td>
<td>Timer counter</td>
</tr>
</tbody>
</table>

### tmr1 class (8 bit timer 1)

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcrl</td>
<td>Timer control register</td>
</tr>
<tr>
<td>tcsr1</td>
<td>Timer control/status register</td>
</tr>
<tr>
<td>tcora1</td>
<td>Timer constant register A</td>
</tr>
<tr>
<td>tcorb1</td>
<td>Timer constant register B</td>
</tr>
<tr>
<td>tcnt1</td>
<td>Timer counter</td>
</tr>
</tbody>
</table>

### sci class (Serial communication interface)

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>smr</td>
<td>Serial mode register</td>
</tr>
<tr>
<td>brr</td>
<td>Bit rate register</td>
</tr>
<tr>
<td>scr</td>
<td>Serial control register</td>
</tr>
<tr>
<td>tdr</td>
<td>Transmit data register</td>
</tr>
<tr>
<td>ssr</td>
<td>Serial status register</td>
</tr>
<tr>
<td>rdr</td>
<td>Receive data register *1</td>
</tr>
</tbody>
</table>

Using the Emulator 5-13
### adc class  (A/D converter)

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addra</td>
<td>A/D data register A *1</td>
</tr>
<tr>
<td>addrb</td>
<td>A/D data register B *1</td>
</tr>
<tr>
<td>addrc</td>
<td>A/D data register C *1</td>
</tr>
<tr>
<td>addrd</td>
<td>A/D data register D *1</td>
</tr>
<tr>
<td>adcsr</td>
<td>A/D control/status register</td>
</tr>
<tr>
<td>adcr</td>
<td>A/D control register</td>
</tr>
</tbody>
</table>

### NO CLASS  The following register names are not included in any register class.

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>r0h</td>
<td>Register 0 H</td>
</tr>
<tr>
<td>r0l</td>
<td>Register 0 L</td>
</tr>
<tr>
<td>r1h</td>
<td>Register 1 H</td>
</tr>
<tr>
<td>r1l</td>
<td>Register 1 L</td>
</tr>
<tr>
<td>r2h</td>
<td>Register 2 H</td>
</tr>
<tr>
<td>r2l</td>
<td>Register 2 L</td>
</tr>
<tr>
<td>r3h</td>
<td>Register 3 H</td>
</tr>
<tr>
<td>r3l</td>
<td>Register 3 L</td>
</tr>
<tr>
<td>r4h</td>
<td>Register 4 H</td>
</tr>
<tr>
<td>r4l</td>
<td>Register 4 L</td>
</tr>
<tr>
<td>r5h</td>
<td>Register 5 H</td>
</tr>
<tr>
<td>r5l</td>
<td>Register 5 L</td>
</tr>
<tr>
<td>r6h</td>
<td>Register 6 H</td>
</tr>
<tr>
<td>r6l</td>
<td>Register 6 L</td>
</tr>
<tr>
<td>r7h</td>
<td>Register 7 H</td>
</tr>
<tr>
<td>r7l</td>
<td>Register 7 L</td>
</tr>
</tbody>
</table>

*1 Display only  
*2 Modification only
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.
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.

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.

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Variable Name</th>
<th>You Enter:</th>
</tr>
</thead>
<tbody>
<tr>
<td>main.c</td>
<td>count</td>
<td>main.c:count</td>
</tr>
<tr>
<td>main.c</td>
<td>line number 23</td>
<td>main.c: line 23</td>
</tr>
</tbody>
</table>

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.
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
```

The PC Interface has a file format option under the “Memory Load” 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 “Memory Load.” The memory load menu will appear.
3. Specify the file format as “HP64000.” This will appear as the default file format.
4. Specify the name of an HP 64000 linker symbol file (TESTFILE.L for example).

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

- It checks to see if two files with the same base name and extensions .HPS and .HPA already exist (for example, TESTFILE.HPS and TESTFILE.HPA).
If TESTFILE.HPS and TESTFILE.HPA don’t exist, the HP 64000 Reader produces them. The new absolute file, TESTFILE.HPA, is then loaded into the emulator.

If TESTFILE.HPS and TESTFILE.HPA already exist but the create dates and times are earlier than the HP 64000 linker symbol file creation date/time, the HP 64000 Reader recreates them. The new absolute file, TESTFILE.HPA, is then loaded into the emulator.

If TESTFILE.HPS and TESTFILE.HPA already exist but the dates and times are later than the creation date and time for the HP 64000 linker symbol file, the HP 64000 Reader will not recreate TESTFILE.HPA. The current absolute file, TESTFILE.HPA, is then loaded into the emulator.

---

**Note**

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

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

---

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

The PC Interface executes the Reader with the “-q” (quiet) option by default.

---

**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 described 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 64876 format file into two files that are usable with the HP 64736 emulator. This means you can use available language tools to create HP 64876 format absolute files, then load those files into the emulator using the H8/338 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 64876 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 H8/330 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.
You must use the required options for your specific language tools to include symbolic ("debug") information in the HP 64876 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
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.

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

The space preceding module names is required. Although formatted for readability here, a single tab separates symbol and address.
The local symbols are scoped. When accessing the variable named "count" in the source file module named "main.c", you would enter "main:count". Notice that the module name of the source file "main.c" is "main". see Table A-2.

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Variable Name</th>
<th>You Enter:</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>count</td>
<td>main:count</td>
</tr>
<tr>
<td>main</td>
<td>line number 23</td>
<td>main: line 23</td>
</tr>
</tbody>
</table>

**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 64876 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 H8/338 PC Interface has a file format option under the "Memory, Load" command. After you select this option, the HP 64869 Reader will operate on the file you specify. After this completes successfully, the H8/338 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 H8/338 PC Interface.
2. Select "Memory, Load". The memory load menu will appear.
3. Specify the file format as "HP64876". This will appear as the default file format.
4. Specify a file in HP 64876 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 64876 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 64876 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 64876 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 64876 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 64876 format file will automatically create them.
# Index

<table>
<thead>
<tr>
<th>A</th>
<th>absolute files</th>
</tr>
</thead>
<tbody>
<tr>
<td>.HPA created by HP 64869 Reader, A-6</td>
<td></td>
</tr>
<tr>
<td>.hpa created by HP 64000 Reader, A-1</td>
<td></td>
</tr>
<tr>
<td>loading, 2-12</td>
<td></td>
</tr>
<tr>
<td>storing, 5-5</td>
<td></td>
</tr>
<tr>
<td>analysis begin, 2-29</td>
<td></td>
</tr>
<tr>
<td>analysis display, 2-29</td>
<td></td>
</tr>
<tr>
<td>analysis specification</td>
<td></td>
</tr>
<tr>
<td>resetting the, 2-27</td>
<td></td>
</tr>
<tr>
<td>saving, 2-29</td>
<td></td>
</tr>
<tr>
<td>trigger condition, 2-27</td>
<td></td>
</tr>
<tr>
<td>triggering by data, 2-38</td>
<td></td>
</tr>
<tr>
<td>analyzer</td>
<td></td>
</tr>
<tr>
<td>features of, 1-4</td>
<td></td>
</tr>
<tr>
<td>status qualifiers, 2-28</td>
<td></td>
</tr>
<tr>
<td>triggering by data, 2-38</td>
<td></td>
</tr>
<tr>
<td>using the, 2-27</td>
<td></td>
</tr>
<tr>
<td>ASCII symbol file (&lt;file&gt;.hps), A-1</td>
<td></td>
</tr>
<tr>
<td>ASCII symbol files</td>
<td></td>
</tr>
<tr>
<td>.HPS created by HP 64869 Reader, A-6</td>
<td></td>
</tr>
</tbody>
</table>

| B | BNC connector, 5-2 |
| break command, 2-20, 2-23, 2-41 |
| breakpoints |
| software, 2-24 |
| breaks, 1-4 |
| guarded memory accesses, 2-9 |
| on analyzer trigger, 5-3 |
| software breakpoints, 1-4 |
| write to ROM, 4-4 |
| writes to ROM, 2-9 |

| C | cautions |
| filenames in the memory store command, 5-5 |
| internal memory must be assigned as emulation memory, 2-9 |
real-time dependent target system circuitry, 4-4
characterization of memory, 2-9
cim, Terminal Interface command, 2-24
clock source
  external, 4-3
  internal, 4-3
clock source selection, emulator configuration, 4-3
clock speed, 1-4
CMB (coordinated measurement bus), 5-2
  enabling interaction, 4-5
  execute signal while emulator is reset, 2-41
  signals, 5-2
command file
  creating and using, 2-39
commands (PC Interface), selecting, 2-8
Configuration
  for sample program, 2-8
  reset value for stack pointer, 2-8
configuration (emulator)
  accessing, 4-2
  break processor on write to ROM, 4-4
  clock selection, 4-3
  enable /NMI input, 4-6
  enable CMB interaction, 4-5
  enable software breakpoints, 4-5
  honor target reset, 4-7
  loading, 4-11
  processor mode, 4-8
  processor type, 4-7 - 4-8
  restrict to real-time runs, 4-3
  stack pointer, 4-9
  storing, 4-10
configuration(hardware), installing the emulator, 2-2
coordinated measurements
  break on analyzer trigger, 5-3
  definition, 5-2
  multiple emulator start/stop, 4-5
  run at /EXECUTE, 5-3
  count, step command, 2-20
D device table, emulator, 2-7
displaying the trace, 2-29
emulation analyzer, 1-4
emulation memory, 1-4
  RAM and ROM, 2-9
  size of, 2-9
emulator
  before using, 2-2
  clock speed, 1-4
  device table, 2-7
  features of, 1-3
  memory mapper resolution, 2-9
  prerequisites, 2-2
  purpose of, 1-1
  reset, 2-41
  running from target reset, 3-10
  status, 2-8
  supported microprocessors, 1-3
Emulator limitations, 1-5
eram, memory characterization, 2-9
erom, memory characterization, 2-9
EXECUTE
  CMB signal, 5-2
  run at, 5-3
executing programs, 2-22
exiting the PC Interface, 2-42
external clock source, 4-3
features of the emulator, 1-3
file format
  HP 64876, A-6
file formats
  HP64000, A-4
find data in memory, 2-23
function codes
  memory mapping, 2-9
getting started
  prerequisites, 2-2
  global symbols, 2-14, 2-19
  grd, memory characterization, 2-9
  guarded memory accesses, 2-9
hardware installation, 2-2
  HP 64000 Reader, A-1
using with PC Interface, A-4
HP 64000 Reader command (RHP64000.EXE), A-4
HP 64869 Reader
using with PC Interface, A-9
HP 64869 Reader command (RD64869.EXE), A-8
HP 64876 format, 2-13
   loading, 2-13
HP64000 file format, A-4
HP64000 format, 2-13
HPTABLES environment variable, 2-7

I  in-circuit emulation, 4-1
   installation
      hardware, 2-2
      software, 2-2
   Installing target system probe
      target system probe, 3-2
      internal clock source, 4-3

L  Limitations
   foreground monitor, 1-5
   RAME enable bit is not effective, 1-6
   sleep and software standby mode, 1-5
   step command and interrupts, 1-6
   store condition and trace, 1-6
   limitations of the emulator, 1-5
   line numbers, 2-30
   link the sample program, 2-6
   linkers, 2-9
   load map, 2-9
   loading absolute files, 2-12
   local symbols, 2-15, 2-25, A-3, A-8
   locked, PC Interface exit option, 2-42
   logging of commands, 2-39

M  make file, A-1, A-6
   mapping memory, 2-9
   memory
      displaying in mnemonic format, 2-18
      emulation, 1-4
      mapper resolution, 1-4
      mapping, 2-9
      modifying, 2-21

4-Index
re-assignment of emulation memory blocks, 2-12
searching for data, 2-23
memory characterization, 2-9
memory mapping
  function codes, 2-9
  ranges, maximum, 2-9
microprocessors, supported by HP 64736 emulator, 1-3

N  non-maskable interrupt, 4-6
notes
  "Timing" option only with external analyzer, 5-4
  absolute file names for stored memory, 5-5
  CMB interaction enabled on execute command, 4-6
  config. option for reset stack pointer recommended, 4-10
  date checking only in PC Interface, A-5, A-10
  displaying complete traces, 2-30
  re-assignment of emul. mem. blocks by mapper, 2-12
  register command, 2-20
  setting software bkpts. while running user code, 2-25
  software breakpoint locations, 2-24
  software breakpoints and ROM code, 2-24
  terminal window to modify emul. config., 4-2
  use required options to include symbols, A-2, A-7
  write to ROM analyzer status, 4-4

O  out-of-circuit emulation, 4-1

P  PC Interface
  exiting the, 2-42
  HP 64000 Reader, A-4
  HP 64869 Reader, A-9
  selecting commands, 2-8
  starting the, 2-7
pin guard
  conductive pin guard for H8/329 emulator, 3-3
  non-conductive pin guard for H8/338 emulator, 3-3
pin protector
  target system probe, 3-3
predefining stack pointer, 4-9
prerequisites for getting started, 2-2
processor operation mode, 4-8
processor type, 4-7 - 4-8
purpose of the emulator, 1-1

Q  qualifiers, analyzer status, 2-28

R  RAM, mapping emulation or target, 2-9
Raw HP64000 format, 2-13
reader
  RD64869, A-6
READY, CMB signal, 5-2
real-time operation, 1-5
real-time runs
  restricting the emulator to, 4-3
register class
  H8/329, 5-11
  H8/338, 5-6
register display/modify, 2-20
register name
  H8/329, 5-11
  H8/338, 5-6
registers, 1-4
relocatable files, 2-9
reset, 2-41
reset (emulator)
  running from target reset, 3-10
reset(emulator), 1-5
resetting the analyzer specifications, 2-27
restrict to real-time runs
  emulator configuration, 4-3
  permissible commands, 4-3
  target system dependency, 4-4
ROM
  mapping emulation or target, 2-9
writes to, 2-9
run at /EXECUTE, 5-3
run from target reset, 3-10
running programs, 2-22

S  s Command
  step command and interrupts, 1-6
sample program, linking, 2-6
sample programs
  for getting started, 2-2
saving analysis specifications, 2-29
searching for data in memory, 2-23
selecting PC Interface commands, 2-8
simple trigger, specifying, 2-27
single-step, 1-4
software breakpoints, 1-4, 2-24
clearing, 2-26
defining (adding), 2-25
displaying, 2-26
enabling, 4-5
setting, 2-26
software installation, 2-2
specifications
   See analysis specification
stack pointer
   reset value, 2-8
stack pointer, defining, 4-9
starting the trace, 2-29
status (analyzer) qualifiers, 2-28
status line, 2-8
status qualifiers, H8/338, 2-28
step, 2-19
count specification, 2-20
supervisor stack pointer
   required for proper operation, 4-10
symbols, 2-14
   .HPS file format, A-2, A-7
global, 2-19
   local, 2-25, A-2, A-7

T
target reset
   running from, 3-10
target system, 1-1
   interface (H8/329), 3-8
   interface (H8/338), 3-6
target system dependency on executing code, 4-4
Target system probe
   cautions for installation, 3-2
   installation, 3-2
   installation procedure, 3-3
   pin guard, 3-2
   pin protector, 3-3

Index-7
target system RAM and ROM, 2-9
trace
  analyzer signals, 2-27
description of listing, 2-30
displaying the, 2-29
starting the, 2-29
  store condition and its effect, 2-35
tram, memory characterization, 2-9
TRIG1, TRIG2 internal signals, 5-3
trigger, 2-27
  breaking into monitor on, 5-3
  specifying a simple, 2-27
trigger state, 2-30
TRIGGER, CMB signal, 5-2
trom, memory characterization, 2-9

U  undefined software breakpoint, 2-24
  unlocked, PC Interface exit option, 2-42
  using the HP 64000 file reader, A-1

W  write to ROM break, 4-4

Z  zoom, window, 2-14, 2-18