HP 64739

H8/536 Emulator
Terminal Interface

User’s Guide

HP Part No. 64739-97003
Printed in U.S.A.
February 1994
Edition 2
Notice

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

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

© Copyright 1994, Hewlett-Packard Company.

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

UNIX is a registered trademark of AT&T.

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

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

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in subparagraph (C) (1) (ii) of the Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013. Hewlett-Packard Company, 3000 Hanover Street, Palo Alto, CA 94304
New editions are complete revisions of the manual. The date on the title page changes only when a new edition is published.

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

**Edition 1**
64739-97000, February 1991

**Edition 2**
64739-97003, February 1994
This manual introduces you to the following emulators as used with the Terminal Interface.

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

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

<table>
<thead>
<tr>
<th>Model</th>
<th>Supported Microprocessors</th>
<th>Referred to as</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP 64739A (H8/536 emulator)</td>
<td>HD6475368CP</td>
<td>H8/536</td>
</tr>
<tr>
<td></td>
<td>HD6435368CP</td>
<td>H8/536</td>
</tr>
<tr>
<td></td>
<td>HD6475348CP</td>
<td>H8/534</td>
</tr>
<tr>
<td></td>
<td>HD6435348CP</td>
<td>H8/534</td>
</tr>
<tr>
<td>HP 64739B (H8/536S emulator)</td>
<td>HD6475368SCP</td>
<td>H8/536S</td>
</tr>
<tr>
<td></td>
<td>HD6435368SCP</td>
<td>H8/536S</td>
</tr>
<tr>
<td></td>
<td>HD6475348SCP</td>
<td>H8/534S</td>
</tr>
<tr>
<td></td>
<td>HD6435348SCP</td>
<td>H8/534S</td>
</tr>
</tbody>
</table>

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

- 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.

This manual will not:

- tell you how to use each and every emulator/analyzer command (refer to the User’s Reference manual)
**Organization**

**Chapter 1**  An introduction to the H8/536 emulator features and how they can help you in developing new hardware and software.

**Chapter 2**  A brief introduction to using the H8/536 Emulator. You will load and execute a short program, and make some measurements using the emulation analyzer.

**Chapter 3**  How to plug the emulator probe into a target system.

**Chapter 4**  Configuring the emulator to adapt it to your specific measurement needs.

**Appendix A**  Using a foreground monitor program; advantages and disadvantages.

**Appendix B**  H8/536 Emulator Specific Command Syntax

**Appendix C**  H8/536 Emulator Specific Error Messages
Notes
## Table of Contents

1 **Introduction to the H8/536 Emulator**
   - Purpose of the H8/536 Emulator ........................................... 1-1
   - Features of the H8/536 Emulator ........................................... 1-3
     - Supported Microprocessors ............................................... 1-3
     - Clock Speeds ...................................................................... 1-3
     - Emulation memory .............................................................. 1-4
     - Analysis ............................................................................. 1-5
     - Registers ........................................................................... 1-5
     - Single-Step ......................................................................... 1-5
     - Target System Interface ...................................................... 1-5
     - Breakpoints ........................................................................ 1-5
     - Reset Support ...................................................................... 1-5
     - Real Time Operation ............................................................ 1-6
     - Limitations, Restrictions ...................................................... 1-6
     - DMA Support ...................................................................... 1-6
     - Monitor Break at Sleep/Standby Mode ................................. 1-6
     - Watch Dog Timer in Background .......................................... 1-6
     - RAM Enable Bit .................................................................... 1-6

2 **Getting Started**
   - Before You Begin .................................................................. 2-2
   - A Look at the Sample Program .............................................. 2-3
   - Using the Help Facility ......................................................... 2-6
   - Initialize the Emulator to a Known State .............................. 2-7
   - Set Up the Proper Emulation Configuration ............................ 2-8
     - Set Up Emulation Conditions .............................................. 2-8
     - Map Memory ....................................................................... 2-10
   - Transfer Code into Emulation Memory .................................. 2-11
     - Transferring Code from a Terminal In ................................. 2-11
     - Transferring Code From A Host, HP 64700 In .................... 2-11
     - Transparent Configuration .................................................. 2-13
   - Looking at Your Code ............................................................ 2-16
   - Familiarize Yourself with the System Prompts ..................... 2-17
Running the Sample Program ........................................ 2-18
Stepping Through the Program ...................................... 2-20
Tracing Program Execution .......................................... 2-21
Using Software Breakpoints ........................................ 2-24
  Displaying and Modifying the Break Conditions ......... 2-24
  Defining a Software Breakpoint ............................. 2-25
Searching Memory for Strings or Numeric Expressions .... 2-26
Making Program Coverage Measurements ......................... 2-26

3 Using the H8/536 Emulator In-Circuit
  Installing the Target System Probe ......................... 3-2
  Pin Guard ...................................................... 3-2
  Installing the Target System Probe ......................... 3-3
  Target System Interface ...................................... 3-4

4 Configuring the H8/536 Emulator
  Types of Emulator Configuration ............................... 4-1
  Emulation Processor to Emulator/Target System .......... 4-1
  Commands Which Perform an Action or Measurement ..... 4-1
  Coordinated Measurements .................................. 4-2
  Analyzer ...................................................... 4-2
  System ......................................................... 4-2
  Emulation Processor to Emulator/Target System .......... 4-3
    cf .......................................................... 4-3
    cf ba ...................................................... 4-4
    cf chip ..................................................... 4-5
    cf clk ....................................................... 4-5
    cf dbc ....................................................... 4-6
    cf drst ..................................................... 4-7
    cf mode ..................................................... 4-7
    cf mon ....................................................... 4-8
    cf nmi ....................................................... 4-10
    cf rrt ....................................................... 4-10
    cf rsp ....................................................... 4-11
    cf tbusrel .................................................. 4-12
    cf trst ....................................................... 4-12
  Memory Mapping ............................................... 4-13
  Break Conditions ................................................ 4-15
  Restrictions and Considerations ............................. 4-16
  Monitor Break at Sleep/Standby Mode ....................... 4-16
  Watch Dog Timer in Background ............................... 4-16

2-Contents
Introduction to the H8/536 Emulator

Introduction

The topics in this chapter include:

- Purpose of the H8/536 Emulator
- Features of the H8/536 Emulator

Purpose of the H8/536 Emulator

The HP 64739 H8/536 Emulator is designed to replace the H8/536 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 H8/536 emulator performs just like the H8/536 microprocessor, but is a device that allows you to control the H8/536 directly. These features allow you to easily debug software before any hardware is available, and ease the task of integrating hardware and software.
Figure 1-1. HP 64739 Emulator for the H8/536 Processor

1-2 Introduction to the H8/536 Emulator
Features of the H8/536 Emulator

Supported Microprocessors

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

Table 1-1. Supported Microprocessors

<table>
<thead>
<tr>
<th>Model</th>
<th>Supported Microprocessors</th>
<th>Referred to as</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP 64739A (H8/536 emulator)</td>
<td>HD6475368CP</td>
<td>H8/536</td>
</tr>
<tr>
<td></td>
<td>HD6435368CP</td>
<td>H8/536</td>
</tr>
<tr>
<td></td>
<td>HD6475348CP</td>
<td>H8/534</td>
</tr>
<tr>
<td></td>
<td>HD6435348CP</td>
<td>H8/534</td>
</tr>
<tr>
<td>HP 64739B (H8/536S emulator)</td>
<td>HD6475368CP</td>
<td>H8/536</td>
</tr>
<tr>
<td></td>
<td>HD6435368CP</td>
<td>H8/536</td>
</tr>
<tr>
<td></td>
<td>HD6475348CP</td>
<td>H8/534</td>
</tr>
<tr>
<td></td>
<td>HD6435348CP</td>
<td>H8/534</td>
</tr>
<tr>
<td></td>
<td>HD6475368SCP</td>
<td>H8/536S</td>
</tr>
<tr>
<td></td>
<td>HD6435368SCP</td>
<td>H8/536S</td>
</tr>
<tr>
<td></td>
<td>HD6475348SCP</td>
<td>H8/534S</td>
</tr>
<tr>
<td></td>
<td>HD6435348SCP</td>
<td>H8/534S</td>
</tr>
</tbody>
</table>

Clock Speeds

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

When you use an external crystal, you need to input conforming to the specification of microprocessor.
### Table 1-2. Clock Speeds

<table>
<thead>
<tr>
<th>Clock source</th>
<th>Model</th>
<th>Microprocessor</th>
<th>Clock Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal</td>
<td>HP 64739A (H8/536 emulator)</td>
<td>H8/536</td>
<td>10MHz (System clock)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>H8/534</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HP 64739B (H8/536S emulator)</td>
<td>H8/536</td>
<td>10MHz (System clock)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>H8/534</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>H8/536S</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>H8/534S</td>
<td></td>
</tr>
<tr>
<td>External</td>
<td>HP 64739A (H8/536 emulator)</td>
<td>H8/536</td>
<td>From 0.5 up to 10MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>H8/534</td>
<td>(System clock)</td>
</tr>
<tr>
<td></td>
<td>HP 64739B (H8/536S emulator)</td>
<td>H8/536</td>
<td>From 0.5 up to 10MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>H8/534</td>
<td>(System clock)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>H8/536S</td>
<td>From 0.5 up to 16MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>H8/534S</td>
<td>(System clock)</td>
</tr>
</tbody>
</table>

**Emulation memory**

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

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

You can define up to 16 memory ranges (at 256 byte boundaries and least 256 byte in length.) The emulator occupies 2K byte, which is used for monitor program, leaving 126K, 510K, 1022K byte of emulation memory which you may use. You can characterize memory range as emulation RAM (eram), emulation ROM (erom), target system RAM (tram), target system ROM (trom), or guarded memory (grd). The emulator generates an error message when accesses are made to guarded memory locations. You can also configure the emulator so that writes to memory defined as ROM cause emulator execution to break out of target program execution.

---

1-4 Introduction to the H8/536 Emulator
**Analysis**  The H8/536 emulator is used with one of the following analyzers which allows you to trace code execution and processor activity.

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

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

**Registers**  You can display or modify the H8/536 internal register contents. This includes the ability to modify the program counter (PC) and the code page register (CP) values 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.

**Target System Interface**  You can set the interface to the target system to be active or passive during background monitor operation.

**Breakpoints**  You can set the emulator/analyzer interaction so the emulator will break to the monitor program when the analyzer finds a specific state or states, allowing you to perform post-mortem analysis of the program execution. You can also set software breakpoints in your program using the `bp` command. This feature is realized by inserting a special instruction into user program. One of undefined opcodes (1B hex) is used as software breakpoint instruction. Refer to the "Using Software Breakpoints" section of "Getting Started" chapter for more information.

**Reset Support**  The emulator can be reset from the emulation system under your control; or your target system can reset the emulation processor.
Real Time Operation

Real-time signifies continuous execution of your program at full rated processor speed without interference from the emulator. (Such interference occurs when the emulator needs to break to the monitor to perform an action you requested, such as displaying target system memory.) Emulator features performed in real time include: running and analyzer tracing. Emulator features not performed in real time include: display or modify of target system memory; load/dump of any memory, display or modification of registers, and single step.

Limitations, Restrictions

- **DMA Support**
  Direct memory access to emulation memory is not allowed.

- **Monitor Break at Sleep/Standby Mode**
  When the emulator breaks into the emulation monitor, sleep or software standby mode is released.

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

- **RAM Enable Bit**
  The internal RAM of H8/536 processor can be enabled/disabled by RAME (RAM enable bit). However, once you map the internal RAM area to emulation RAM, the emulator accesses emulation RAM even if the internal RAM is disabled by RAME.
Introduction
This chapter will lead you through a basic, step by step tutorial designed to familiarize you with the use of the HP 64700 emulator for the H8/536 microprocessor. When you have completed this chapter, you will be able to perform these tasks:

- Set up an emulation configuration for out of circuit emulation use
- Map memory
- Transfer a small program into emulation memory
- Use run/stop controls to control operation of your program
- Use memory manipulation features to alter the program’s operation
- Use analyzer commands to view the real time execution of your program
- Use software breakpoint feature to stop program execution at specific address
- Search memory for strings or numeric expressions
- Make program coverage measurements
Before You Begin

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

1. Completed hardware installation of the HP 64700 emulator in the configuration you intend to use for your work:
   - Standalone configuration
   - Transparent configuration
   - Remote configuration

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

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

   U>
   R>
   M>

   If you do not see one of these command prompts, retrace your steps through the hardware and software installation procedures outlined in the manuals above, verifying all connections and procedural steps. If you are still unable to get a command prompt, refer to the HP 64700 Support Services Guide. The guide gives basic troubleshooting procedures. If this fails, call the local HP sales and service office listed in the Support Services Guide.

   In any case, you must have a command prompt on your terminal screen before proceeding with the tutorial.
A Look at the Sample Program

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

Data Declarations

INPUT_POINTER and OUTPUT_POINTER define the address locations of an input area and an output area to be used by the program. MESSAGE_A, MESSAGE_B and INVALID_INPUT are the messages used by the program to respond to various command inputs.

Initialization

The locations of the input and output areas are moved into address registers for use by the program. Next, the CLEAR routine clears the command byte (the first byte location pointed to by the input area address - fc00 hex).

READ_INPUT

This routine continuously reads the byte at location fc00 hex until it is something other than a null character (00 hexadecimal); when this occurs, the PROCESS_COMM routine is executed.

PROCESS_COMM

Compares the input byte (now something other than a null) to the possible command bytes of "A" (ASCII 41 hex) and "B" (ASCII 42 hex), then jumps to the appropriate set up routine for the command message. If the input byte does not match either of these values, a branch to a set up routine for an error message is executed.

COMMAND_A, COMMAND_B, UNRECOGNIZED

These routines set up the proper parameters for writing the output message: the number of bytes in the message is moved to the R6 register and the base address of the message in the data area is moved to address register R0.
Figure 2-1. Sample Program Listing

2-4 Getting Started
OUTPUT

First the base address of the output area is copied to R1 (this preserves R3 for use in later program passes). Then the CLEAR_OLD routine writes nulls to 32 bytes of the output area (this serves both to initialize the area and to clear old messages written during previous program passes).

Finally, the proper message is written to the output area by the OUTPUT_LOOP routine. When done, OUTPUT_LOOP jumps back to CLEAR and the command monitoring process begins again.

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 utilizing the memory access commands provided by the HP 64700 command set.
**Using the Help Facility**

If you need a quick reference to the Terminal Interface syntax, you can use the built-in `help` facilities. For example, to display the top level `help` menu, type:

```
R> help
```

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

--- VALID <group> NAMES ---
gram          - system grammar
proc          - processor specific grammar
sys           - system commands
emul          - emulation commands
trc           - analyzer trace commands
*             - all command groups
```

You can type the `?` symbol instead of typing `help`. For example, if you want a list of commands in the `emul` command group, type:

```
R> ? emul
```

To display help information for any command, just type `help` (or `?`) and the command name. For example:

```
R> help load
```
Initialize the Emulator to a Known State

To initialize the emulator to a known state for this tutorial:

**Note**

It is especially important that you perform the following step if the emulator is being operated in a standalone mode controlled by only a data terminal. The only program entry available in this mode is through memory modification; consequently, if the emulator is reinitialized, emulation memory will be cleared and a great deal of tedious work could be lost.

1. Verify that no one else is using the emulator or will have need of configuration items programmed into the emulator.

2. Initialize the emulator by typing the command:

   ```
   R> init -p
   ```
Set Up the Proper Emulation Configuration

To set the emulator’s configuration values to the proper state for this tutorial, do this:

1. Type:
   
   ```
   R> cf
   ```

   You should see the following configuration items displayed:

   ```
   cf ba=en
   cf chip=536
   cf clk=int
   cf dbc=en
   cf drst=dis
   cf mode=ext
   cf mon=bg
   cf nmi=en
   cf rrt=dis
   cf rsp=9
   cf tbusrel=en
   cf trst=en
   ```

   **Note**
   The individual configuration items won’t be explained in this example; refer to Chapter 4 of this manual and the *User’s Reference* manual for details.

   2. If the configuration items displayed on your screen don’t match the ones listed above, here is how to make them agree:

   For each configuration item that does not match, type:

   ```
   R> cf <config_item>=<value>
   ```
For example, if you have the following configuration items displayed (those in **bold** indicate items different from the list above):

```
cf ba=en
cf chip=536
**cf clk=ext**
cf dbc=en
cf drst=dis
cf mode=ext
cf mon=bg
cf nmi=en
**cf rrt=en**
cf rsp=9
cf tbusrel=en
cf trst=en
```

To make these configuration values agree with the desired values, type:
```
R> cf clk=int
R> cf rrt=dis
```

3. Now, you need to set up stack pointer.
Type:
```
R> cf rsp=0fe00
```

4. Let’s go ahead and set up the proper break conditions.
Type:
```
R> bc
```

You will see:
```
bc -d bp #disable
bc -d rom #enable
bc -d bnct #disable
bc -d cmbt #disable
bc -d trig1 #disable
bc -d trig2 #disable
```

For each break condition that does not match the one listed, use one of the following commands:

To enable break conditions that are currently disabled, type:
```
R> bc -e <breakpoint type>
```
To disable break conditions that are currently enabled, type:
R> bc -d <breakpoint type>

For example, if typing bc gives the following list of break conditions:

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

(items in **bold** indicate improper values for this example)

Type the following commands to set the break conditions correctly for this example:

R> bc -e rom
   (this enables the write to ROM break)
R> bc -d trig1 trig2
   (this disables break on triggers from the analyzer)

### Map Memory

The H8/536 emulator is provided with 126K byte of high-speed emulation memory which you can configure as you desire. This allows you to store programs and data used in development before target system memory is available. You can define emulation memory as emulation RAM, emulation ROM, target RAM, target ROM or guarded memory. For this example, map the address 0 hex through 2fff hex as emulation ROM, and fc00 hex through feff hex as emulation RAM.

Type:

```
R> map 0..2fff erom
R> map 0fc00..0feff eram
```

To verify that memory blocks are mapped properly, type:

```
R> map
```

You will see:

```
# remaining number of terms : 14
# remaining emulation memory : 1c500h bytes
map 000000..002fff erom  # term 1
map 00fc00..00feff eram  # term 2
map other tram
```

2-10 Getting Started
Transfer Code into Emulation Memory

Transferring Code from a Terminal In Standalone Configuration

To transfer code into emulation memory from a data terminal running in standalone mode, you must use the modify memory commands. This is necessary because you have no host computer transfer facilities to automatically download the code for you (as if you would if you were using the transparent configuration or the remote configuration.)

To minimize the effects of typing errors, you will modify only one row of memory at a time in this example. Do the following:

1. Enter the data information for the program by typing the following commands:

   R> m 1000..100f=0fc,00,0fd,00,54,48,49,53,20,49,53,20,4d,45,53,53
   R> m 1010..101f=41,47,45,20,41,54,48,49,53,20,49,53,20,4d,45,53
   R> m 1020..102f=53,41,47,45,20,42,49,4e,56,41,4c,49,44,20,43,4f
   R> m 1030..1034=4d,4d,41,4e,44

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

   You could also type the following line instead:

   R> m 1000=0fc,00,0fd,00,"THIS IS MESSAGE ATHIS IS MESSAGE BINVALID COMMAND"

2. You should now verify that the data area of the program is correct by typing:

   R> m 1000..1034

   You should see:

   001000..00100f      fc 00 fd 00 54 48 49 53 20 49 53 20 4d 45 53 53
   001010..00101f      41 47 45 20 41 54 48 49 53 20 49 53 20 4d 45 53
   001020..00102f      53 41 47 45 20 42 49 4e 56 41 4c 49 44 20 43 4f
   001030..001034      4d 4d 41 4e 44

   If this is not correct, you can correct the errors by re-entering only the modify memory commands for the particular rows of memory that are wrong.
For example, if row 1000..100f shows these values:

```
001000..00100f  0fc 00 00 0fd 00 54 48 49 53 20 49 53 20 4d 45 53
```

you can correct this row of memory by typing:

```
R> m 1000..100f=0fc,00,0fd,00,54,48,49,53,20,49,53,20,4d,45,53,53
```

Or, you might need to modify only one location, as in the instance where address 100f equals 55 hex rather than 53 hex. Type:

```
R> m 100f=53
```

3. Enter the program information by typing the following commands:

```
M> m 2000..200f=5f,0fe,00,1d,10,00,82,1d,10,02,83,0d2,06,00,0d2,04
M> m 2010..201f=00,27,0fb,0d2,04,41,27,07,0d2,04,42,27,09,20,0e,56
M> m 2020..202f=11,58,10,04,20,0c,56,11,58,10,15,20,05,56,0f,58
M> m 2030..203f=10,26,0ab,81,5d,00,20,0c1,06,00,0ad,0c,26,0f9,0a3,81
M> m 2040..2049=0c0,85,0c1,95,0ae,0c,26,0f8,20,0c1
```

If this is not correct, you can correct the errors by re-entering only the modify memory commands for the particular rows of memory that are wrong.

4. You should now verify that the program area is correct by typing:

```
R> m 2000..2049
```

You should see:

```
002000..00200f  5f fe 00 1d 10 00 82 1d 10 02 83 d2 06 00 d2 04
002010..00201f  00 27 fb d2 04 41 27 07 d2 04 42 27 09 20 0e 56
002020..00202f  11 58 10 04 20 0c 56 11 58 10 15 20 05 56 0f 58
002030..00203f  10 26 ab 81 5d 00 20 c1 06 00 ad 0c 26 f9 a3 81
002040..002049  c0 85 c1 95 ae 0c 26 f8 20 c1
```

2-12 Getting Started
The method provided in this example assumes that you are running an HP 64869 H8/500 Assembler/Linkage Editor on an HP 9000/300 computer running the HP-UX operating system. In addition, you must have the HP 64000 transfer software running on your host.

If you are not using an HP 64869 H8/500 Assembler/Linkage Editor, you may be able to adapt the methods below to load your code into the emulator (refer to the HP 64700 User’s Reference manual for help).

If you are not able to transfer code from your host to the emulator using one of these methods, use the method described previously under "Transferring Code From A Terminal In Standalone Mode", as it will work in all cases. However, transferring code using host transfer facilities is easier and faster than modifying memory locations, especially for large programs.

1. First, you must establish communications with your host computer through the transparent mode link provided in the HP 64700. Type:

   R> xp -s 02a

   This sets the second escape character to "*". (The first escape character remains at the HP 64700 powerup default of hex 01b, which is the ASCII <ESC> character.) The sequence "<ESC>*" toggles the transparent mode software within the HP 64700 for the duration of one command (that is, any valid line of HP 64700 commands (not exceed 254 characters) concatenated by semicolons and terminated by a <carriage return>). Refer to the User’s Reference manual for more information on the xp command.

   Enable the transparent mode link by typing:

   R> xp -e

   If you then press <RETURN> a few times, you should see:
This is the login prompt for an HP-UX host system. (Your prompt may differ depending on how your system manager has configured your system.)

2. Log in to your host system and start up an editor such as "vi". You should now enter the source code for the sample program shown at the beginning of the chapter. When finished, save the program to filename "sampprog.src".

Note
If you need help learning how to log in to your HP-UX host system or use other features of the system, such as editors, refer to the HP-UX Concepts and Tutorials guides and your HP-UX system administrator.

3. Assemble your code with the following command.

$ h8asm sampprog
If any assembly errors were reported, re-edit your file and verify that the code was entered correctly.

4. Link the program to the correct addresses and generate absolute file with the following command.

$ h8lnk sampprog

5. Convert the SYSROF absolute file generated above into HP format with the following command. This is needed to load the file into the emulator. Refer to the HP 64869 H8/500 Assembler/Linkage Editor manual for more details.

$ h8cnvhp -x sampprog
An HP format absolute file sampprog.X will be generated.

Now it's time to transfer your code into the emulator. Do the following:

1. Disable the transparent mode so that your terminal will talk directly to the emulator. Type:

$ <ESC>* xp -d

2-14 Getting Started
The "<ESC>*" sequence temporarily toggles the transparent mode so that the emulator will accept commands; "xp -d" then fully disables the transparent mode.

2. Load code into the emulator by typing:

   R> load -hbo
   transfer -rtb sampprog.X<ESC>* (NOTE: DO NOT TYPE CARRIAGE RETURN!)

   The system will respond:
   ##
   R>

   load -hbo tells the emulator to load code expected in HP binary file format and to expect the data from the other port (the one connected to the host). It then puts you in communication with the host; you then enter the transfer command to start the HP 64000 transfer utility. Typing "<ESC>*" tells the system to return to the emulator after transferring the code. The "##" marks returned by the system indicates that the emulator loaded two records from the host.

3. At this point you should examine a portion of memory to verify that your code was loaded correctly.

   Type:
   R> m 1000..1034
   You should see:

   If your system does not match, verify 1) that you entered the source code correctly; 2) that you entered the linker parameters correctly.
Now that you have loaded your code into emulation memory, you can display it in mnemonic format. Type:

```
R> m -dm 2000..2049
```

You will see:

```
002000  -  MOV:I.W #fe00,R7
002003  -  MOV:G.W @1000,R2
002007  -  MOV:G.W @1002,R3
00200b  -  MOV:G.B #00,0R2
00200e  -  CMP:G.B #00,0R2
002011  -  BEQ 0200e
002013  -  CMP:G.B #41,0R2
002016  -  BEQ 0201f
002018  -  CMP:G.B #42,0R2
00201b  -  BEQ 0202
00201d  -  BRA 0202d
00201f  -  MOV:E.B #11,R6
002021  -  MOV:I.W #1004,R0
002024  -  BRA 02032
002026  -  MOV:E.B #11,R6
002028  -  MOV:I.W #1015,R0
00202b  -  BRA 02032
00202d  -  MOV:E.B #0f,R6
00202f  -  MOV:I.W #1026,R0
002032  -  MOV:G.W R3,1R
002034  -  MOV:G.W #0020,R5
002037  -  MOV:G.B #00,0R1+
00203a  -  ADD:Q.W #-1,R5
00203c  -  BNE 02037
00203e  -  MOV:G.B R3,1R
002040  -  MOV:G.B @R0+,R5
002042  -  MOV:G.B R5,0R1+
002044  -  ADD:Q.W #-1,R6
002046  -  BNE 02040
002048  -  BRA 0200b
```
Familiarize Yourself with the System Prompts

The following steps are not intended to be complete explanations of each command; the information is only provided to give you some idea of the meanings of the various command prompts you may see and reasons why the prompt changes as you execute various commands.

You should gain some familiarity with the HP 64700 emulator command prompts by doing the following:

1. Ignore the current command prompt. Type:
   
   *\( \texttt{rst} \)
   
   You will see:
   
   R>
   
   The \texttt{rst} command resets the emulation processor and holds it in the reset state. The "R>" prompt indicates that the processor is reset.

2. Type:
   
   R> \texttt{r 2000}
   
   You will see:
   
   U>
   
   The \texttt{r} command runs the processor from address 2000 hex.

3. Type:
   
   U> \texttt{b}
   
   You will see:
The `b` command causes the emulation processor to "break" execution of whatever it was doing and begin executing within the emulation monitor. The "M>" prompt indicates that the emulator is running in the monitor.

---

**Running the Sample Program**

4. Type:

```
M> r 2000
```

The emulator changes state from background to foreground and begins running the sample program from location 2000 hex.

---

**Note**

The default number base for address and data values within HP 64700 is hexadecimal. Other number bases may be specified. Refer to the Tutorials chapter of this manual or the *HP 64700 User’s Reference* manual for further details.

---

5. Let's look at the registers to verify that the address registers were properly initialized with the pointers to the input and output areas. Type:

```
U> reg
```

You will see:

```
reg pc=2011 cp=00 sr=0704 dp=00 ep=00 br=00 r0=0000 r1=0000 r2=fc00
greg r3=fd00 r4=0000 r5=0000 r6=0000 r7=fe00 fp=0000 sp=fe00 mdcr=c7
```

Notice that R2 contains fc00 hex; R3 contains fd00 hex.
6. Verify that the input area command byte was cleared during initialization.

Type:

U> m -db 0fc00

You will see:

00fc00..00fc00     00

The input byte location was successfully cleared.

7. Now we will use the emulator features to make the program work. Remember that the program writes specific messages to the output area depending on what the input byte location contains. Type:

U> m 0fc00=41

This modifies the input byte location to the hex value for an ASCII "A". Now let’s check the output area for a message.

U> m 0fd00..0fd1f

You will see:

00FD00..00FD0f     54 48 49 53 20 49 53 20 4d 45 53 53 41 47 45 20
00FD10..00FD1f     41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

These are the ASCII values for MESSAGE_A.

Repeat the last two commands twice. The first time, use 42 instead of 41 at location fc00 and note that MESSAGE_B overwrites MESSAGE_A. Then try these again, using any number except 00, 41, or 42 and note that the INVALID_INPUT message is written to this area.
Stepping Through the Program

8. You can also direct the emulator processor to execute one instruction or number of instructions. Type:

   M> s 1 2000;reg

   This command steps 1 instruction from address 2000 hex, and displays registers. You will see:

   ![Assembly code output]

   Notice that PC contains 2003 hex.

9. To step one instruction from present PC, you only need to type s at prompt. Type:

   M> s;reg

   You will see:

   ![Assembly code output]

2-20 Getting Started
Tracing Program Execution

Now let’s use the emulation analyzer to trace execution of the program. Suppose that you would like to start the trace when the analyzer begins writing data to the message output area. You can do this by specifying analyzer trigger upon encountering the address fd00 hex. Furthermore, you might want to store only the data written to the output area. This can be accomplished by modifying what is known as the “analyzer storage specification”.

Note

For this example, you will be using the analyzer in the easy configuration, which simplifies the process of analyzer measurement setup. The complex configuration allows more powerful measurements, but requires more interaction from you to set up those measurements. For more information on easy and complex analyzer configurations and the analyzer, refer to the HP 64700 Analyzer User’s Guide and the User’s Reference.

Now, let’s set the trigger specification. Type:

```
M> tg addr=0fd00
```

To store only the accesses to the address range fd00 through fd11 hex, type:

```
M> tsto addr=0fd00..0fd11
```

Let’s change the data format of the trace display so that you will see the output message writes displayed in ASCII format:

```
M> tf addr,h data,A count,R seq
```

Start the trace by typing:

```
M> t
```

You will see:

```
Emulation trace started
```

To start the emulation run, type:

```
M> r 2000
```

Now, you need to have a "command" input to the program so that the program will jump to the output routines (otherwise the trigger will not be found, since the program will never access address fd00 hex). Type:

```
U> m 0fc00=41
```
To display the trace list, type:

U> t1 0..34

You will see:

If you look at the last lines of the trace listing, you will notice that the
analyzer seems to have stored only part of the output message, even
though you specified more than the full range needed to store all of the
message. The reason for this is that the analyzer has a storage pipeline,
which holds states that have been acquired but not yet written to trace
memory. To see all of the states, halt the analyzer by typing:

U> th

You will see:

Emulation trace halted

2-22 Getting Started
Now display the trace list:

\[ U> \text{tl 0..34} \]

You will see:

As you can see, all of the requested states have been captured by the analyzer.

<table>
<thead>
<tr>
<th>Line</th>
<th>addr,R</th>
<th>data,A</th>
<th>count,R</th>
<th>seq</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0fd00</td>
<td>..</td>
<td>---</td>
<td>+</td>
</tr>
<tr>
<td>1</td>
<td>0fd01</td>
<td>..</td>
<td>1.920 uS</td>
<td>.</td>
</tr>
<tr>
<td>2</td>
<td>0fd02</td>
<td>..</td>
<td>1.880 uS</td>
<td>.</td>
</tr>
<tr>
<td>3</td>
<td>0fd03</td>
<td>..</td>
<td>1.920 uS</td>
<td>.</td>
</tr>
<tr>
<td>4</td>
<td>0fd04</td>
<td>..</td>
<td>1.880 uS</td>
<td>.</td>
</tr>
<tr>
<td>5</td>
<td>0fd05</td>
<td>..</td>
<td>1.920 uS</td>
<td>.</td>
</tr>
<tr>
<td>6</td>
<td>0fd06</td>
<td>..</td>
<td>1.880 uS</td>
<td>.</td>
</tr>
<tr>
<td>7</td>
<td>0fd07</td>
<td>..</td>
<td>1.920 uS</td>
<td>.</td>
</tr>
<tr>
<td>8</td>
<td>0fd08</td>
<td>..</td>
<td>1.880 uS</td>
<td>.</td>
</tr>
<tr>
<td>9</td>
<td>0fd09</td>
<td>..</td>
<td>1.920 uS</td>
<td>.</td>
</tr>
<tr>
<td>10</td>
<td>0fd0a</td>
<td>..</td>
<td>1.880 uS</td>
<td>.</td>
</tr>
<tr>
<td>11</td>
<td>0fd0b</td>
<td>..</td>
<td>1.920 uS</td>
<td>.</td>
</tr>
<tr>
<td>12</td>
<td>0fd0c</td>
<td>..</td>
<td>1.880 uS</td>
<td>.</td>
</tr>
<tr>
<td>13</td>
<td>0fd0d</td>
<td>..</td>
<td>1.920 uS</td>
<td>.</td>
</tr>
<tr>
<td>14</td>
<td>0fd0e</td>
<td>..</td>
<td>1.880 uS</td>
<td>.</td>
</tr>
<tr>
<td>15</td>
<td>0fd0f</td>
<td>..</td>
<td>1.920 uS</td>
<td>.</td>
</tr>
<tr>
<td>16</td>
<td>0fd10</td>
<td>..</td>
<td>1.880 uS</td>
<td>.</td>
</tr>
<tr>
<td>17</td>
<td>0fd11</td>
<td>..</td>
<td>1.920 uS</td>
<td>.</td>
</tr>
<tr>
<td>18</td>
<td>0fd00</td>
<td>TT</td>
<td>28.68 uS</td>
<td>.</td>
</tr>
<tr>
<td>19</td>
<td>0fd01</td>
<td>HH</td>
<td>2.320 uS</td>
<td>.</td>
</tr>
<tr>
<td>20</td>
<td>0fd02</td>
<td>II</td>
<td>2.280 uS</td>
<td>.</td>
</tr>
<tr>
<td>21</td>
<td>0fd03</td>
<td>SS</td>
<td>2.320 uS</td>
<td>.</td>
</tr>
<tr>
<td>22</td>
<td>0fd04</td>
<td>..</td>
<td>2.280 uS</td>
<td>.</td>
</tr>
<tr>
<td>23</td>
<td>0fd05</td>
<td>II</td>
<td>2.320 uS</td>
<td>.</td>
</tr>
<tr>
<td>24</td>
<td>0fd06</td>
<td>SS</td>
<td>2.280 uS</td>
<td>.</td>
</tr>
<tr>
<td>25</td>
<td>0fd07</td>
<td>..</td>
<td>2.320 uS</td>
<td>.</td>
</tr>
<tr>
<td>26</td>
<td>0fd08</td>
<td>MM</td>
<td>2.280 uS</td>
<td>.</td>
</tr>
<tr>
<td>27</td>
<td>0fd09</td>
<td>EE</td>
<td>2.320 uS</td>
<td>.</td>
</tr>
<tr>
<td>28</td>
<td>0fd0a</td>
<td>SS</td>
<td>2.280 uS</td>
<td>.</td>
</tr>
<tr>
<td>29</td>
<td>0fd0b</td>
<td>SS</td>
<td>2.320 uS</td>
<td>.</td>
</tr>
<tr>
<td>30</td>
<td>0fd0c</td>
<td>AA</td>
<td>2.280 uS</td>
<td>.</td>
</tr>
<tr>
<td>31</td>
<td>0fd0d</td>
<td>GG</td>
<td>2.320 uS</td>
<td>.</td>
</tr>
<tr>
<td>32</td>
<td>0fd0e</td>
<td>EE</td>
<td>2.280 uS</td>
<td>.</td>
</tr>
<tr>
<td>33</td>
<td>0fd0f</td>
<td>..</td>
<td>2.320 uS</td>
<td>.</td>
</tr>
<tr>
<td>34</td>
<td>0fd10</td>
<td>AA</td>
<td>2.280 uS</td>
<td>.</td>
</tr>
</tbody>
</table>
You can stop program execution at specific address by using `bp` (software breakpoint) command. When you define a software breakpoint to a certain address, the emulator will replace the opcode with one of undefined opcode (1B hex) as software breakpoint instruction. When the emulator detects the special instruction, user program breaks to the monitor, and the original opcode will be placed at the breakpoint address. A subsequent run or step command will execute from this address.

If the special instruction was not inserted as the result of `bp` command (in other words, it is part of the user program), the "Undefined software breakpoint" message is displayed.

**Note**
You can 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 software breakpoint instruction, you cannot define software breakpoints in target ROM. You can, however, copy target ROM into emulation memory by `cim` command when you are using the background monitor. (Refer to HP 64700 Terminal Interface User’s Reference manual.)

**Displaying and Modifying the Break Conditions**
Before you can define software breakpoints, you must enable software breakpoints with the `bc` (break conditions) command. To view the default break conditions and change the software breakpoint condition, enter the following commands.

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

M> bc -e bp

Defining a Software Breakpoint

Now that the software breakpoint is enabled, you can define software breakpoints. Enter the following command to break on the address of the OUTPUT_LOOP label.

M> bp 2032

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

M> r 2000
U> m 0fc00=41
!ASYNC_STAT  615! Software break point: 002032

M> reg

reg pc=2032 cp=00 sr=0700 dp=00 ep=00 tp=00 br=00 r0=1004 r1=fd11 r2=fc00
reg r3=fd00 r4=0000 r5=0041 r6=0011 r7=fe00 fp=0011 sp=fe00 mdcr=c7

Notice that PC contains 2032.

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

M> bp

###BREAKPOINT FEATURE IS ENABLED###
bp 002032 #disabled

M> bp -e 2032
M> bp

###BREAKPOINT FEATURE IS ENABLED###
bp 002032 #enabled

M> r 2000
U> m 0fc00=41
!ASYNC_STAT  615! Software break point: 002032

M> bp

###BREAKPOINT FEATURE IS ENABLED###
bp 002032 #disabled
Searching Memory for Strings or Numeric Expressions

The HP 64700 Emulator provides you with tools that allow you to search memory for data strings or numeric expressions. For example, you might want to know exactly where a string is loaded. To locate the position of the string "THIS IS MESSAGE A" in the sample program. Type:

\[ M> \text{ser 0..1fff="THIS IS MESSAGE A"} \]

pattern match at address: 001004

You can also find numeric expressions. For example, you might want to find all of the BEQ instructions in the sample program. Since a BEQ instruction begins with 27 hex, you can search for that value by typing:

\[ M> \text{ser -db 2000..2049=27} \]

pattern match at address: 002011
pattern match at address: 002016
pattern match at address: 00201b

Making Program Coverage Measurements

In testing your program, you will often want to verify that all possible code segments are executed. With the sample program, we might want to verify that all of the code is executed if a command "A", command "B", and an unrecognized command are input to the program.

To make this measurement, we must first reset the coverage status.

\[ M> \text{cov -r} \]
You should **always** reset the coverage status before making a coverage measurement. Any emulator system command which accesses emulation memory will affect the coverage status bit, resulting in measurement errors if the coverage status is not reset.

Now, run the program and input the three commands:

```
M> r 2000
M> m 0fe00=41
M> m 0fe00=42
M> m 0fe00=43
```

Make the coverage measurement:

```
U> cov 2000..2049
percentage of memory accessed: % 100.0
```

You're now finished with the "Getting Started" example. You can proceed on with using the emulator and use this manual and the *HP 64700 Terminal Interface User's Reference* manual as needed to answer your questions.
Notes
Using the H8/536 Emulator In-Circuit

When you are ready to use the H8/536 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 H8/536 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 H8/536 Emulator before inserting the user plug to avoid circuit damage resulting from voltage transients or mis-insertion of the user plug.

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

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

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

**Pin Guard**

HP 64739 H8/536 emulator is shipped with a non-conductive pin guard over the target system probe. This guard is designed to prevent impact damage to the pins and should be left in place while you are not using the emulator.

3-2 In-Circuit Emulation
Installing the Target System Probe

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

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

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

Note

To make sure the contact between emulator probe and target system microprocessor socket, we recommend that you use ITT CANNON "LCS-84" series 84 pin PLCC socket.

Figure 3-1. Installing Probe into the Target System
Target System Interface

MD2  MD1  These signals are connected to 74HCT14 through 51 ohm series register and 10K ohm pull-up register.
MD0  /NMI
/STBY

P10  P11  These signals are connected to H8/536 emulation processor through 51 ohm series register and 10K ohm pull-up register.
P14–P17
P2  P3  P4  P5
P6  P7  P9

P12  P13  These signals are connected to H8/536 emulation processor and 74HCT27 through 51 ohm series register and 10K ohm pull-up register.
P12  (BZACK)
P13  (NREQD)
This signal is connected to H8/536 emulation processor through 51 ohm series register.
Notes

3-6 In-Circuit Emulation
Configuring the H8/536 Emulator

In this chapter, we will discuss:

- how to configure the HP 64700 emulator for H8/536 microprocessor to fit your particular measurement needs.
- some restrictions of HP 64700 emulator for H8/536 microprocessor.

Types of Emulator Configuration

The HP 64700 Emulator is different from other HP emulators (such as those in the HP 64000-UX system) in that there are several different classes of configuration commands.

Emulation Processor to Emulator/Target System

These are the commands which are generally thought of as "configuration" items in the context of other HP 64000 emulator systems. The commands in this group set up the relationships between the emulation processor and the target system, such as determining how the emulator responds to requests for the processor bus. Also, these commands determine how the emulation processor interacts with the emulator itself; memory mapping and the emulator’s response to certain processor actions are some of the items which can be configured.

These commands are the ones which are covered in this chapter.

Commands Which Perform an Action or Measurement

Several of the emulator commands do not configure the emulator; they simply start an emulator program run or other measurement, begin or halt an analyzer measurement, or allow you to display the results of such measurements.
These commands are covered in the examples presented in earlier manual chapters; they are also covered in the *HP 64700 Terminal Interface: User’s Reference manual*.

**Coordinated Measurements**

These commands determine how the emulator interacts with other measurement instruments, such as external analyzers, or other HP 64700 emulators connected via the CMB (Coordinated Measurement Bus).

These commands are covered in the *HP 64700 CMB User’s Guide* and in the *HP 64700 Terminal Interface: User’s Reference Manual*.

**Analyzer**

The analyzer configuration commands are those commands which actually specify what type of measurement the analyzer is to make.

Some of the analyzer commands are covered earlier in this manual. You can also refer to the *HP 64700 Terminal Interface: Analyzer User’s Guide* and the *HP 64700 Terminal Interface: User’s Reference manual*.

**System**

This last group of commands is used by you to set the emulator’s data communications protocol, load or dump contents of emulation memory, set up command macros, and so on.

These commands are covered earlier in this manual and in the manual titled *HP 64700 Terminal Interface: User’s Reference*.
As noted before, these commands determine how the emulation processor will interact with the emulator’s memory and the target system during an emulation measurement.

The `cf` command defines how the emulation processor will respond to certain target system signals. It also defines the type of emulation monitor to be used and optionally defines the location of that monitor in emulation memory.

To see the default configuration settings defined by the `cf` command, type:

```
M> cf
```

You will see:

```
 cf ba=en
 cf chip=536
 cf clk=int
 cf dbc=en
 cf drst=dis
 cf mode=ext
 cf mon:bg
 cf nmi=en
 cf rrt=dis
 cf rsp=9
 cf tbusrel=en
 cf trst=en
```

Let’s examine each of these emulator configuration options, with a view towards how they affect the processor’s interaction with the emulator.
**cf ba**

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

**M> cf ba=en**

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

---

**Note**

External DMA (Direct Memory Access) device is prohibited from accessing to emulation memory.

---

**M> cf ba=dis**

When you disable bus arbitration by entering the above command, the emulator ignores the /BREQ signal from the target system. The emulation processor will never drive the /BACK line true; nor will it place the address, data and control signals into the tri-state mode.

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

The *chip* option allows you to select processor to be emulated.

If you are going to emulate H8/536 processor, configure the emulator with the following command.

```
M> cf chip=536
```

If you are going to emulate H8/534 processor, configure the emulator with the following command.

```
M> cf chip=534
```

---

**Note**

Executing this command will drive the emulator into the reset state.

---

**cf clk**

The *clk* (clock) option allows you to select whether the emulation processor’s clock will be sourced by your target system or by the emulator.

```
M> cf clk=int
```

You can select the emulator’s internal 10 MHz system clock using the above command.

```
M> cf clk=ext
```

You can specify that the emulator should use the clock input to the emulator probe from the target system as the system clock. You must use a clock input conforming to the specifications for the H8/536 microprocessor.

---

**Note**

Executing this command will drive the emulator into the reset state.
The *dbc* (drive background cycles) option allows you to select whether or not the emulator will drive the target system bus on background cycles.

If you have selected to use a foreground monitor with the `cf mon=fg` command, emulator monitor cycles will appear at the target interface exactly as if they were user program cycles.

```
M> cf dbc=en
```

You can enable background cycle drive to target system by entering the above command. Emulation processor’s address and control strobes (except /WR) are driven during background cycles.

Background write cycles won’t appear to the target system. (/WR signal is always "high" when the *dbc* option is enabled.)

```
M> cf dbc=dis
```

If you specify the above command, background monitor cycles are not driven to the target system. When you select this option, the emulator will appear to the target system as if it continuously between bus cycles while it is operating in the background monitor.

You use the *dbc* option to avoid target system interaction problems. For example, your target system interaction scheme may depend on the constant repetition of bus cycles. In such case, using the *dbc* option will help avoid the problem.

---

**Note**

Executing this command will drive the emulator into the reset state.

---

4-6 Configuring the Emulator
**cf drst**  The **drst** (drive reset) configuration item allows you to specify whether or not the mulator drives the /RES signal to the target system during emulation reset.

\[
\text{M}> \text{cf drst}=\text{dis}
\]

The above command configures the emulator not to drive the reset signal to the target.

\[
\text{M}> \text{cf drst}=\text{en}
\]

The emulator will drive the reset signal to the target system during emulation reset.

To drive the reset signal, the emulator must be configured to respond to the target reset with the **cf trst=on** command.

**cf mode**  The **mode** (cpu operation mode) configuration item defines operation mode in which the emulator works.

\[
\text{M}> \text{cf mode}=\text{ext}
\]

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

\[
\text{M}> \text{cf mode}=<\text{mode_num}>
\]

When <mode_num> is selected, the emulator will operate in selected mode regardless of the mode setting by the target system.

Valid <mode_num> are following:

<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 minimum mode)</td>
</tr>
<tr>
<td>2</td>
<td>The emulator will operate in mode 2. (expanded minimum mode with internal ROM)</td>
</tr>
<tr>
<td>3</td>
<td>The emulator will operate in mode 3. (expanded maximum mode)</td>
</tr>
<tr>
<td>4</td>
<td>The emulator will operate in mode 4. (expanded maximum mode with internal ROM)</td>
</tr>
</tbody>
</table>

*Configuring the Emulator 4-7*
The emulator will operate in mode 7. (single chip mode)

Note

Executing this command will drive the emulator into the reset state.

**cf mon**

The **mon** (monitor) configuration item allows you to choose between a foreground monitor supplied by you or the background monitor supplied with the emulator.

The emulation monitor is the program that handles communication between the emulation controller and the emulation processor. For example, when you ask for a register display, the processor is broken to the monitor, executes some code to store its register contents in an array of memory locations, then returns to executing your program.

The background monitor provided with the emulator offers the greatest degree of transparency to your target system (that is, your target system should generally be unaffected by monitor execution). However, in some cases you may require an emulation monitor tailored to the requirements of your system. In this case, you will need to use a foreground monitor linked into your program modules. See Appendix A of this manual for more information on foreground monitors.

```
cf mon=bg
```

You select the use of the built-in background monitor through the above command. A memory overlay is created and the background monitor is loaded into that area.
You select the use of your foreground monitor using this command. XXXXX defines an hexadecimal address where the monitor will be located. (Note: this will not load the monitor, it only specifies its location). You can define the location on a 2 kbyte boundary (address ending in 000 hex or 800 hex) of 00800 hex through 0ff800 hex. (When you are using the emulator in mode1, mode2, or mode7, 00800 hex through 0f800 hex is available.) 00000 hex is not available as the location.

Remember that you must assemble and link your foreground monitor starting at the 2 kbyte boundary specified in the command above. You must also load the monitor into emulation memory.

---

**Note**

If you intend to use a foreground monitor, the monitor must be loaded before attempting to load any information into target system memory.

---

**Note**

When you use the foreground monitor, the trace exception vector in the target system must point to TRACE_ENTRY in the foreground monitor to use single step command. (Refer to Appendix A of this manual.)

---

A memory mapper term is automatically created when you execute the `cf mon=fg` command to reserve 2 kilobytes of memory space for the monitor.

The memory map is reset any time `cf mon=fg` is entered. It is only reset when `cf mon=bg` if the emulator is not already configured to use the background monitor.
The **nmi** (non maskable interrupt) configuration item determines whether or not the emulator responds to NMI signal from the target system during foreground operation.

```
M> cf nmi=en
```

Using the above command, you can specify that the emulator will respond to NMI from the target system.

```
M> cf nmi=dis
```

The emulator won’t respond to NMI from the target system.

If you are using the background monitor, the emulator does not accept any interrupt during background execution. NMI and /IRQ1 are latched last one during in background, and such interrupts will occur when context is changed to foreground. /IRQ0 and internal interrupts are ignored during in background operation.

---

**Note**

Executing this command will drive the emulator into the reset state.

---

The **rrt** (restrict to real time) option lets you configure the emulator so that commands which cause the emulator to break to monitor and return to the user program will be rejected by the emulator command interpreter.

```
M> cf rrt=en
```

You can restrict the emulator to accepting only commands which don’t cause temporary breaks to the monitor by entering the above command. Only the following emulator run/stop commands will be accepted:

- **rst** (resets emulation processor)
- **b** (breaks processor to background monitor until you enter another command)
- **r** (runs the emulation processor from a given location)
- **s** (steps the processor through a piece of code -- returns to monitor after each step)
Commands which cause the emulator to break to the monitor and return, such as `reg, m` (for target memory display), and others will be rejected by the emulator.

If your target system circuitry is dependent on constant execution of program code, you should set this option to `cf rrt=en`. This will help insure that target system damage doesn’t occur. However, remember that you can still execute the `rst, b` and `s` commands; you should use caution in executing these commands.

```
M> cf rrt=dis
```

When you use this command, all commands, regardless of whether or not they require a break to the emulation monitor, are accepted by the emulator.

```
cf rsp
```

The `rsp` (reset stack pointer) configuration item allows you to specify a value to which the stack pointer and stack page register will be set upon the transition from emulation reset into the emulation monitor.

```
R> cf rsp=XXXXX
```

where `XXXXX` is a 20-bit even address, will set the stack pointer and stack page register to that value upon entry to the emulation monitor after an emulation reset.

You cannot set `rsp` at the following location.

- Odd address
- Internal I/O register area
- Internal RAM which is disabled (in mode 7 only)

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

For example, to set the stack pointer to 0fe00 hex, type:

```
R> cf rsp=0fe00
```

Now, if you break the emulator to monitor using the `b` command, the stack pointer will be modified to the value 0fe00 hex.
Without a stack pointer, 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.

**cf tbusrel**

The **tbusrel** (trace bus release cycles) configuration item defines whether or not the emulator traces bus release cycles.

**M> cf tbusrel=en**

When you enable this item with the above command, each time bus release occurs, one emulation analyzer state will be generated to recognize the bus release cycle.

**M> cf tbusrel=dis**

When disabled, no analyzer state will be generated at the occurrence of bus release. Therefore, any bus release cycle will be ignored by the analyzer.

**cf trst**

The **trst** (target reset) configuration item allows you to specify whether or not the emulator responds to /RES and /STBY signals by the target system during foreground operation. While running the background monitor, the emulator ignores /RES and /STBY signals, otherwise the emulator status is “waiting for the target system reset (prompt is T>)”. (You can see the emulator status with es command.)

**M> cf trst=en**

When you enable target system reset with the above command, the emulator will respond to /RES and /STBY input during foreground operation.

**M> cf trst=dis**

When disabled, the emulator won’t respond to /RES and /STBY input form the target system.

**Note**

Executing this command will drive the emulator into the reset state.

---

4-12 Configuring the Emulator
Memory Mapping

Before you begin an emulator session, you must specify the location and type of various memory regions used by your programs and your target system (whether or not it exists). You do this for several reasons:

- the emulator must know whether a given memory location resides in emulation memory or in target system memory. This allows the emulator to properly orient buffers for the given data transfer.
- the emulator needs to know the size of any emulation memory blocks so it can properly reserve emulation memory space for those blocks.
- the emulator must know if a given space is RAM (read/write), ROM (read only), or doesn’t exist. This allows the emulator to determine if certain actions taken by the emulation processor are proper for the memory type being accessed. For example, if the processor tries to write to a emulation memory location mapped as ROM, the emulator will not permit the write (even if the memory at the given location is actually RAM). (You can optionally configure the emulator to break to the monitor upon such occurrence with the bc -e rom command.) Also, if the emulation processor attempts to access a non existent location (known as "guarded"), the emulator will break to the monitor.

You use the `map` command to define memory ranges and types for the emulator. The HP 64739 H8/536 emulator memory mapper allows you to define up to 16 different map terms; each map term has a minimum size of 256 bytes. If you specify a value less than 256 byte, the emulator will automatically allocate an entire block. You can specify one of five different memory types (`erom, eram, trom, tram, grd`).

For example, you might be developing a system with the following characteristics:

- input port at 0f000 hex
- output port at 0f100 hex
program and data from 2000 through 6fff hex

Suppose that the only thing that exists in your target system at this time are input and output ports and some control logic; no memory is available. You can reflect this by mapping the I/O ports to target system memory space and the rest of memory to emulation memory space. Type the following commands:

```plaintext
R> map 0f000..0f100 tram
R> map 2000..6fff eram
```

| # remaining number of terms | : 14
| # remaining emulation memory | : 1a800h bytes
| map 002000..006fff eram | # term 1
| map 00f000..00f1ff tram | # term 2
| map other tram |

As you can see, the mapper rounded up the second term to 256 bytes block, since those are minimum size blocks supported by the H8/536 emulator.

---

**Note**

When you use the internal memory, you **must** map that area to emulation memory. When you power on the emulator, all memory space is mapped to target RAM. Therefore, if you don’t map internal memory properly, you cannot access that area.

---

**Note**

You should map all memory ranges used by your programs **before** loading programs into memory. This helps safeguard against loads which accidentally overwrite earlier loads if you follow a `map/load` procedure for each memory range.

---

For further information on mapping, refer to the examples in earlier chapters of this manual and to the *HP 64700 Terminal Interface User’s Reference* manual.

---

**4-14 Configuring the Emulator**
**Break Conditions**

The `bc` command lets you configure the emulator’s response to various emulation system and external events.

**Write to ROM**

If you want the emulator to break into the emulation monitor whenever the user program attempts to write to a memory region mapped as ROM, enter:

```
M> bc -e rom
```

You can disable this function by entering:

```
M> bc -d rom
```

When disabled, the emulator will not break to the monitor upon a write to ROM; however, it will not modify the memory location if the memory at that location is actually RAM.

**Software Breakpoints**

The `bp` command allows you to insert software traps in your code which will cause a break to the emulation monitor when encountered during program execution. If you want to enable the insertion and use of software breakpoints by the `bp` command, enter:

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

To disable use of software breakpoints, type:

```
M> bc -d bp
```

Any breakpoints which previously existed in memory are disabled, but are not removed from the breakpoint table.

**Trigger Signals**

The HP 64700 emulator provides four different trigger signals which allow you to selectively start or stop measurements depending on the signal state. These are the `bnct` (rear panel BNC input), `cmbt` (CMB trigger input), `trig1` and `trig2` signals (provided by the analyzer).

You can configure the emulator to break to the monitor upon receipt of any of these signals. Simply type:

```
M> bc -e <signal>
```
For example, to have the emulator break to monitor upon receipt of the \texttt{trig1} signal from the analyzer, type:

\begin{verbatim}
M> bc -e trig1
\end{verbatim}

(Note: in this situation, you must also configure the analyzer to drive the \texttt{trig1} signal upon finding its trigger by entering \texttt{tgout trig1}).

## Restrictions and Considerations

### Monitor Break at Sleep/Standby Mode

When the emulator breaks into the monitor, sleep or software standby mode is released. For example, if you use the \texttt{reg} command at sleep mode, the emulator processor will go into normal state and start execution.

### Watch Dog Timer in Background

Watch dog timer suspends count up while the emulator is running in background monitor.
Using the Optional Foreground Monitor

By using and modifying the optional Foreground Monitor, you can provide an emulation environment which is customized to the needs of a particular target system.

Comparison of Foreground and Background Monitors

An emulation monitor is required to service certain requests for information about the target system and the emulation processor. For example, when you request a register display, the emulation processor is forced into the monitor. The monitor code has the processor dump its registers into certain emulation memory locations, which can then be read by the emulator system controller without further interference.

Background Monitors

A background monitor is an emulation monitor which overlays the processor’s memory space with a separate memory region. Entry into the monitor is normally accomplished by jamming the monitor addresses onto the processor’s address bus.

Usually, a background monitor will be easier to work with in starting a new design. The monitor is immediately available upon powerup, and you don’t have to worry about linking in the monitor code or allocating space for the monitor to use the emulator. No assumptions are made about the target system environment; therefore, you can test and debug hardware before any target system code has been written. All of the processor’s address space is available for target system use, since the monitor memory is overlaid on processor memory, rather than subtracted from processor memory. Processor resources such as interrupts are not taken by the background monitor.
However, all background monitors sacrifice some level of support for the target system. For example, when the emulation processor enters the monitor code to display registers, it will not respond to target system interrupt requests. This may pose serious problems for complex applications that rely on the microprocessor for real-time, non-intrusive support. Also, the background monitor code resides in emulator firmware and can’t be modified to handle special conditions.

**Foreground Monitors**

A foreground monitor may be required for more complex debugging and integration applications. A foreground monitor is a block of code that runs in the same memory space as your program. You link this monitor with your code so that when control is passed to your program, the emulator can still service real-time events, such as interrupts or watchdog timers. For most multitasking, interrupt intensive applications, you will need to use a foreground monitor.

You can tailor the foreground monitor to meet your needs, such as servicing target system interrupts. However, the foreground monitor does use part of the processor’s address space, which may cause problems in some target systems. You must also properly configure the emulator to use a foreground monitor (see Chapter 3 and the examples in this appendix); and, you must link the monitor with your other program code.

A-2 Using A Foreground Monitor
An Example Using the Foreground Monitor

In the following example, we will illustrate how to link a foreground monitor with the sample program from Chapter 2. By using the emulation analyzer, we will also show how the emulator switches from state to state using a foreground monitor.

Select A Monitor Suitable to Your Application

The H8/536 emulator is provided with four foreground monitor programs. You need to select appropriate monitor program as shown in the following table.

<table>
<thead>
<tr>
<th>Processor</th>
<th>Processor Mode</th>
<th>Foreground Monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>H8/536</td>
<td>Mode 1, 2, 7</td>
<td>fm536min.src</td>
</tr>
<tr>
<td>H8/536</td>
<td>Mode 3, 4</td>
<td>fm536max.src</td>
</tr>
<tr>
<td>H8/534</td>
<td>Mode 1, 2, 7</td>
<td>fm534min.src</td>
</tr>
<tr>
<td>H8/534</td>
<td>Mode 3, 4</td>
<td>fm534max.src</td>
</tr>
</tbody>
</table>

For this example, we will use the `fm536min.src` monitor program, and will locate the monitor at 8000 hex; the sample program will be located at 2000 hex with its data at 1000 hex.

Modify Location Declaration Statement

To use the monitor, you must modify the `.SECTION` statement just after the first comment section of the monitor program listing. You should see the line below:

```
LOCATE_ADRS:   .EQU    H'8000           ;start monitor on 2k boundary
.SECTION fm536min,CODE,LOCATE=LOCATE_ADRS
```

You can specify the monitor location by modifying this label LOCATE_ADRS. For example, if you want locate the monitor program at 6000 hex, make above line to as below:

```
LOCATE_ADRS:   .EQU    H'6000           ;start monitor on 2k boundary
.SECTION fm536min,CODE,LOCATE=LOCATE_ADRS
```
Notice that the .SECTION statement is indented from the left margin; if it is not indented, the assembler will attempt to interpret the .SECTION as a label and will generate an error when processing the address portion of the statement. You can load the `fm536min.src` monitor on a 2k byte boundary of 00800 hex through 0f800 hex. When you are going to use `fm536max.src`, you can load it on a 2k byte boundary of 00800 hex through 0ff800 hex.

In this example, we will locate the monitor at 8000 hex. Therefore, you don’t have to modify the monitor program.

You can also specify monitor location when you link it. If you prefer this way, do the following:

- Change .EQU statement and .SECTION statement just after the first comment section into comment line by inserting `;` at the first column of these lines.
- Make the next two lines (.EQU statement and .SECTION statement which doesn’t have LOCATE keyword) effective by deleting `;` at the first column of these lines.
- Specify the monitor location when you link the monitor program. To do this, you can use the START linker subcommand like this:

```
$h8lnk
:INPUT samp prog, fm536min
:OUTPUT testfg
:START fm536min(8000)
:EXIT
```

**Configure the Emulator**

Before configuring the emulator, you should initialize the emulator to a known state. Type:

```
M> init -p
```

You need to tell the emulator that you will be using a foreground monitor and allocate the memory space for the monitor. This is all done with one configuration command. To locate the monitor on a 2k boundary starting at 8000 hex, type:

```
R> cf mon=fg.8000
```

**A-4 Using A Foreground Monitor**
To see the new memory mapper term allocated for the foreground monitor, type:

```
R> map
```

```
# remaining number of terms : 15
# remaining emulation memory : 1f800h bytes
map 0008000..00087ff eram  # term 1
map other tram
```

Notice that a 2k byte block from 8000 through 87ff hex was mapped.

Now, you need to map memory space for the sample program. Type:

```
R> map 0..2fff erom
R> map 0fc00..0feff eram
```

### Set a Stack Pointer

You need to set up the stack pointer for use by the foreground monitor. The foreground monitor use the stack when transit from foreground monitor to user program. You can use the `cf rsp` command to define the stack pointer location; the stack pointer will be initialized on each transition from emulation reset to the monitor. Type:

```
R> cf rsp=0fe00
```

### Load the Program Code

Now it’s time to load the sample program and monitor. Link the sample program provided in chapter 2 and monitor program into absolute file named testfg.abs, and covert it into HP format absolute file named testfg.X. In the example shown, we’re loading the program from a host with the emulator in Transparent Configuration. If you’re using the standalone configuration with a data terminal, you will need to enter the data using the `m` command. (You can get the data from your assembly listings.) See Chapter 2 for information.

Load the program by typing:

```
R> load -hbs "transfer -tb testfg.X"
```

You can also load the sample program and the monitor separately. In this case, you don’t have to link the sample program with the monitor.
Before we forget, let’s initialize the stack pointer by breaking the emulator out of reset:

R> b

Now you can run the sample program with the following command:

M> r 2000

To use the s command to step through processor instructions with either of the monitors listed in this chapter, you must modify the processor’s exception vector table. The entry that you must modify is the trace exception vector. The vector must point to the identifier TRACE_ENTRY in the foreground monitor. You can know the location of TRACE_ENTRY from the assemble listing generated by the assembler.

Address Error During Step Operation

In operation of H8/536 microprocessor, the Stack Pointer (SP) must always contain an even value. Once it becomes to an odd value, an address error will occur. In step operation of H8/536 emulator, if the SP is forced to be an odd value by user program, the emulator will fail to perform step instruction. The emulation processor will read the address error exception vector, and it will continue executing from the address pointed by the vector. If your program doesn’t have proper routine to process the address error, the emulation monitor program may run away. (You will see ?> prompt.)

If the monitor program runs away, try to reset the emulator with rst command. When the emulator cannot restore control, all you can do is to initialize the emulator with init command. In this case, you will lose all the data in emulation memory.
You can avoid the program run away by using an emulation monitor routine. To use the routine, the address error exception vector in your program must point to ADRSERR_ENTRY of the monitor program.

When the address error occurs, the emulator can break into the monitor by using the routine. However, when the emulator breaks into the monitor in this manner, register values are unreliable. Besides, the SP will contain an odd value.

To continue your measurement, you have to do the following:

- Reset the emulator.

Or:

- Modify registers to proper values by yourself.

When you are using the background monitor, you don’t have to worry about this issue. The background monitor can handle it by itself.

---

**Limitations of Foreground Monitors**

**Synchronized measurements**

You cannot perform synchronized measurements over the CMB when using a foreground monitor. If you need to make such measurements, set the foreground/background configuration option to `cf mon=bg`.
A-8 Using A Foreground Monitor
H8/536 Emulator Specific Command Syntax

The following pages contain descriptions of command syntax specific to the H8/536 emulator. The following syntax items are included (several items are part of other command syntax):

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

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

- `<ADDRESS>`. May be specified in emulation commands which allow addresses to be entered.

- `<Register Classes and Names>`. May be specified in the `reg` (register) command.
CONFIG_ITEMS

Summary  H8/536 emulator configuration items.

Syntax
**Description**

The H8/536 emulator has several dedicated configuration items which allow you to specify the emulator’s interaction with the target system and the rest of the emulation system. These items are:

- **ba**  
  Enable/disable bus arbitration with target system.

- **chip**  
  Select Processor to be emulated.

- **clk**  
  Select internal/external clock source.

- **dbc**  
  Enable/disable to drive background cycles to target system.

- **drst**  
  Enable/disable to drive emulation reset to target system.

- **mode**  
  Determine emulator processor operation mode.

- **mon**  
  Select background or foreground monitor.

- **nmi**  
  Enable/disable NMI (non maskable interrupt) from target system.

- **rrt**  
  Restrict emulator to real time runs.

- **rsp**  
  Specify system stack pointer value to load upon each transition from emulation reset to the monitor.

- **tbusrel**  
  Enable/disable tracing bus release cycles.

- **trst**  
  Enable/disable target system reset.

Complete explanations of all configuration items are given in chapter 4 of this manual.
**Examples**  
To select an external clock, type:

\[ M> \textit{cf clk=ext} \]

You can obtain the status of configuration items by typing the item name without a value. You can also specify multiple configuration items on the same line. Type:

\[ M> \textit{cf mon=fg..08000 rrt=dis clk clk=int} \]

Here, we changed to a foreground monitor located at address 8000 hex, disabled the real-time runs restriction, and ask processor clock source. Notice that items which are changed do not have status printed; you could explicitly request the new status by repeating the configuration item on the command line after the change but without a value. For example:

\[ R> \textit{cf mon=fg..2000 mon cf mon=fg..2000} \]

**Related information**  
Refer to the \textit{cf} syntax pages in the \textit{User’s Reference} manual. Also, refer to chapter 3 of this manual for complete information about each configuration item.
ACCESS MODE and DISPLAY MODE

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

Syntax

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

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

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

Note
Only byte is valid to ACCESS MODE since H8/536 microprocessor have 8 bits external data bus width.
Defaults  The <DISPLAY_MODE> is b at power up initialization. Display mode specifications are saved; that is, when a command changes the display mode, the new display mode becomes the current default.

Related Information  Refer to the mo syntax information in the User’s Reference manual for further information on use of the mode command.

ADDRESS

Summary  Address specification used in emulation commands.

Description  The <ADDRESS> parameter used in emulation commands is specified in 20 bits address information.

Examples  m 1000

m 20000..200ff

B-6  H8/536 Emulator Specific Command Syntax
io Command

Syntax

Summary

The io command accesses devices on the target system with MOVFPE/MOVTP instruction.

Restrictions

- The io command accesses target system memory regardless of memory mapping.
- Access to internal memory with io command is not allowed.
- Address range cannot be specified in io command.
Register Classes and Names

**Summary**

H8/536 register designators.

**Description**

The following register classes and names are used with the display/modify registers commands in H8/536 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>cp</td>
<td>Code page register</td>
</tr>
<tr>
<td>sr</td>
<td>Status register</td>
</tr>
<tr>
<td>dp</td>
<td>Data page register</td>
</tr>
<tr>
<td>ep</td>
<td>Extended page register</td>
</tr>
<tr>
<td>tp</td>
<td>Stack page register</td>
</tr>
<tr>
<td>br</td>
<td>Base register</td>
</tr>
<tr>
<td>r0</td>
<td>Register R0</td>
</tr>
<tr>
<td>r1</td>
<td>Register R1</td>
</tr>
<tr>
<td>r2</td>
<td>Register R2</td>
</tr>
<tr>
<td>r3</td>
<td>Register R3</td>
</tr>
<tr>
<td>r4</td>
<td>Register R4</td>
</tr>
<tr>
<td>r5</td>
<td>Register R5</td>
</tr>
<tr>
<td>r6</td>
<td>Register R6</td>
</tr>
<tr>
<td>r7</td>
<td>Register R7</td>
</tr>
<tr>
<td>fp</td>
<td>Frame pointer</td>
</tr>
<tr>
<td>sp</td>
<td>Stack pointer</td>
</tr>
<tr>
<td>mdcr</td>
<td>Mode control register</td>
</tr>
</tbody>
</table>
### sys Class
System control registers

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wcr</td>
<td>Wait control register</td>
</tr>
<tr>
<td>ramcr</td>
<td>RAM control register</td>
</tr>
<tr>
<td>mdcr</td>
<td>Mode control register</td>
</tr>
<tr>
<td>sbycr</td>
<td>Software stand-by control register</td>
</tr>
</tbody>
</table>

### intc Class
Interrupt control registers

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipra</td>
<td>Interrupt priority register A</td>
</tr>
<tr>
<td>iprab</td>
<td>Interrupt priority register B</td>
</tr>
<tr>
<td>iprc</td>
<td>Interrupt priority register C</td>
</tr>
<tr>
<td>iprd</td>
<td>Interrupt priority register D</td>
</tr>
<tr>
<td>ipre</td>
<td>Interrupt priority register E</td>
</tr>
<tr>
<td>iprf</td>
<td>Interrupt priority register F</td>
</tr>
</tbody>
</table>

### dtc Class
Data transfer controller registers

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dtea</td>
<td>DT enable register A</td>
</tr>
<tr>
<td>dteb</td>
<td>DT enable register B</td>
</tr>
<tr>
<td>dtec</td>
<td>DT enable register C</td>
</tr>
<tr>
<td>dted</td>
<td>DT enable register D</td>
</tr>
<tr>
<td>dtee</td>
<td>DT enable register E</td>
</tr>
<tr>
<td>dtcf</td>
<td>DT enable register F</td>
</tr>
</tbody>
</table>
## port Class

I/O port registers

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1ddr</td>
<td>Port 1 data direction register</td>
</tr>
<tr>
<td>p2ddr</td>
<td>Port 2 data direction register</td>
</tr>
<tr>
<td>p3ddr</td>
<td>Port 3 data direction register</td>
</tr>
<tr>
<td>p4ddr</td>
<td>Port 4 data direction register</td>
</tr>
<tr>
<td>p5ddr</td>
<td>Port 5 data direction register</td>
</tr>
<tr>
<td>p6ddr</td>
<td>Port 6 data direction register</td>
</tr>
<tr>
<td>p7ddr</td>
<td>Port 7 data direction register</td>
</tr>
<tr>
<td>p9ddr</td>
<td>Port 9 data direction register</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</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>p1cr</td>
<td>Port 1 control register</td>
</tr>
<tr>
<td>p69cr</td>
<td>Port 69 control register</td>
</tr>
</tbody>
</table>

## frt1 Class

Free running timer 1 registers

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>frtcrl</td>
<td>Timer control register</td>
</tr>
<tr>
<td>frtcsr1</td>
<td>Timer control/status register</td>
</tr>
<tr>
<td>frc1</td>
<td>Free running counter</td>
</tr>
<tr>
<td>ocra1</td>
<td>Output compare register A</td>
</tr>
<tr>
<td>ocrb1</td>
<td>Output compare register B</td>
</tr>
<tr>
<td>icrl</td>
<td>Input capture register</td>
</tr>
</tbody>
</table>
### frt2 Class
Free running timer 2 registers

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>frtcr2</td>
<td>Timer control register</td>
</tr>
<tr>
<td>frtcsr2</td>
<td>Timer control/status register</td>
</tr>
<tr>
<td>frc2</td>
<td>Free running counter</td>
</tr>
<tr>
<td>ocra2</td>
<td>Output compare register A</td>
</tr>
<tr>
<td>ocrb2</td>
<td>Output compare register B</td>
</tr>
<tr>
<td>icr2</td>
<td>Input capture register</td>
</tr>
</tbody>
</table>

### frt3 Class
Free running timer 3 registers

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>frtcr3</td>
<td>Timer control register</td>
</tr>
<tr>
<td>frtcsr3</td>
<td>Timer control/status register</td>
</tr>
<tr>
<td>frc3</td>
<td>Free running counter</td>
</tr>
<tr>
<td>ocra3</td>
<td>Output compare register A</td>
</tr>
<tr>
<td>ocrb3</td>
<td>Output compare register B</td>
</tr>
<tr>
<td>icr3</td>
<td>Input capture register</td>
</tr>
</tbody>
</table>

### tmr Class
Timer registers

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcr</td>
<td>Timer control register</td>
</tr>
<tr>
<td>tcsr</td>
<td>Timer control/status register</td>
</tr>
<tr>
<td>tcora</td>
<td>Timer constant register A</td>
</tr>
<tr>
<td>tcorb</td>
<td>Timer constant register B</td>
</tr>
<tr>
<td>tcnt</td>
<td>Timer counter</td>
</tr>
</tbody>
</table>

### pwm1 Class
PWM timer1 registers

<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>

H8/536 Emulator Specific Command Syntax  B-11
**pwm2 Class**  
PWM timer2 registers

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pwmtcr2</td>
<td>Timer control register</td>
</tr>
<tr>
<td>dtr2</td>
<td>Duty register</td>
</tr>
<tr>
<td>pwmtcnt2</td>
<td>Timer counter</td>
</tr>
</tbody>
</table>

**pwm3 Class**  
PWM timer3 registers

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pwmtcr3</td>
<td>Timer control register</td>
</tr>
<tr>
<td>dtr3</td>
<td>Duty register</td>
</tr>
<tr>
<td>pwmtcnt3</td>
<td>Timer counter</td>
</tr>
</tbody>
</table>

**wdt Class**  
Watchdog timer registers

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wdtcsr</td>
<td>Timer control/status register</td>
</tr>
<tr>
<td>wdtcnt</td>
<td>Timer counter</td>
</tr>
<tr>
<td>rstcsr</td>
<td>Reset control/status register</td>
</tr>
</tbody>
</table>

**sci1 Class**  
Serial communication interface 1 registers.

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdr1</td>
<td>Receive data register</td>
</tr>
<tr>
<td>tdr1</td>
<td>Transmit data register</td>
</tr>
<tr>
<td>smr1</td>
<td>Serial mode register</td>
</tr>
<tr>
<td>scr1</td>
<td>Serial control register</td>
</tr>
<tr>
<td>ssr1</td>
<td>Serial status register</td>
</tr>
<tr>
<td>brr1</td>
<td>Bit rate register</td>
</tr>
</tbody>
</table>
**sci2 Class**  
Serial communication interface 2 registers.

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdr2</td>
<td>Receive data register</td>
</tr>
<tr>
<td>tdr2</td>
<td>Transmit data register</td>
</tr>
<tr>
<td>smr2</td>
<td>Serial mode register</td>
</tr>
<tr>
<td>scr2</td>
<td>Serial control register</td>
</tr>
<tr>
<td>ssr2</td>
<td>Serial status register</td>
</tr>
<tr>
<td>brr2</td>
<td>Bit rate register</td>
</tr>
</tbody>
</table>

**adc Class**  
A/D converter registers

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addra</td>
<td>A/D data register A</td>
</tr>
<tr>
<td>addrb</td>
<td>A/D data register B</td>
</tr>
<tr>
<td>addrc</td>
<td>A/D data register D</td>
</tr>
<tr>
<td>addrd</td>
<td>A/D data register D</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>
H8/536 Emulator Specific Error Messages

The following pages document the error messages which are specific to the HP 64739 H8/536 emulator. The cause of the error is described, as well as the action you must take to remedy the situation.

Message 140 : Stack pointer not initialized

Cause

This error occurs when you attempt to execute user program (with r or s command) without set up the stack pointer. You will see this error message only when you are operating with the foreground monitor. When you are using the background monitor, another error message will be reported.

Action

Set up the stack pointer with cf rsp command. Refer to chapter 3 of this manual for more information.

Message 141 : Stack is in I/O registers

Cause

This error occurs when you attempt to execute user program after you set up the stack pointer at internal I/O register area.
Action
Set up the stack pointer at proper location with \texttt{cf rsp} command. Refer to chapter 3 of this manual for more information.

Message 142 : Stack is in disabled RAM

Cause
This error occurs when you attempt to execute user program after you set up the stack pointer at internal RAM area which is disabled. This error occurs only when you are using the emulator in mode 7.

Action
Set up the stack pointer at proper location with \texttt{cf rsp} command. Refer to chapter 3 of this manual for more information.

Message 143 : Invalid access for disabled RAM

Cause
This error occurs when you attempt to access internal RAM which is disabled and mapped to target memory. This error occurs only when you are using the emulator in mode 7.

Action
Map the area to emulation RAM (eram).
**Message 144** : Invalid address for run or step in minimum mode

**Cause**
This error occurs when you attempt to run or step from address other than page 0 during the emulator operates in minimum mode.

**Message 145** : Code page register not writable in minimum mode

**Cause**
This error occurs when you attempt to modify the Code Page Register during the emulator operates in minimum mode.

**Message 146** : Emulation processor is in single chip mode

**Cause**
This error occurs when you attempt to use the `io` command during the emulator operates in the single chip mode. The `io` command is not available when the emulator operates in the single chip mode.
Notes
Index

A
- Address error during step operation A-6
- ADDRESS syntax B-6
- Analyzer
  - configuration 2-21
  - configuration commands 4-2
  - halting 2-22
  - pipeline 2-22
  - storage specification 2-21
  - trace 2-21
  - trace list display 2-22
  - trace list format 2-21
  - trigger specification 2-21
- Analyzer trace
  - starting 2-21

B
- b Command 2-17
- Background monitor A-1
- bc Command 2-9, 2-24, 4-15
- Before using the emulator 2-2
- bp Command 2-24, 4-15
- Break
  - write to ROM 4-15
- Break condition 2-24
- Breaks 4-15
- Bus arbitration
  - configure emulator’s response 4-4
  - using configuration to isolate target problem 4-4
- Bus release cycles
  - enable/disable tracing bus release cycles 4-12

C
- cf ba Command 4-4
- cf chip Command 4-5
- cf clk Command 4-5
- cf Command 2-8, 4-3
- cf dbc Command 4-6
- cf drst Command 4-7
cf mode Command 4-7
cf mon Command 4-8
cf ami Command 4-10
cf rrt Command 4-10
cf rsp Command 4-11
cf tbusrel Command 4-12
cf trst Command 4-12
cim Command 2-24
Clock selection for microprocessor 4-5
Command help 2-6
Command prompts 2-17
Command syntax, specific to H8/536 emulator B-1
Commands
   analyzer configuration 4-2
   b 2-17
   bc 2-9, 2-24, 4-15
   bp 2-24, 4-15
   cf 2-8, 4-3
   cf ba 4-4
   cf chip 4-5
   cf clk 4-5
   cf dbc 4-6
   cf drst 4-7
   cf mode 4-7
   cf mon 4-8
   cf nmi 4-10
   cf rrt 4-10
   cf rsp 4-11
   cf tbusrel 4-12
   cf trst 4-12
   cim 2-24
   configuration 4-1
   coordinated measurement 4-2
   cov 2-26
   es 4-12
   help 2-6
   init 2-7
   io B-7
   load 2-15
   m 2-11, 2-19
   map 2-10, 4-13
Comparison of foreground/background monitors A-1
CONFIG ITEMS syntax B-2
Configuration
  analyzer 4-2
  breaks 4-15
  bus arbitration 4-4
  clock selection 4-5
  displaying 4-3
  drive background cycles to target 4-6
  drive emulation reset to target 4-7
  enable/disable target interrupts 4-10
  enable/disable target system reset 4-12
  enable/disable to trace bus release cycles 4-12
  for getting started 2-8
  foreground/background monitor 4-8
  measurement commands 4-1
  memory mapping 4-13
  microprocessor operation mode 4-7
  processor to emulator/target system 4-1, 4-3
  restrict to real-time runs 4-10
  select processor 4-5
  stack pointer 4-11
  system 4-2
  to access the internal memory 4-14
  types of 4-1
Coordinated measurement commands 4-2
cov Command 2-26
Coverage measurement 2-26

D  Displaying
   configuration 4-3
   memory 2-19
   registers 2-18
   trace list 2-22
DMA limitations 4-4

E  E clock synchronous target access B-7
Emulator
   configuration 2-8
   initialization 2-7
   purpose 1-1
Emulator features 1-3
   analyzer 1-5
   breakpoints 1-5
   clock speeds 1-3
   emulation memory 1-4
   processor reset control 1-5
   register display/modify 1-5
   restrict to real-time runs 1-6
   single-step processor 1-5
   supported microprocessors 1-3
Emulator limitations 1-6
   DMA support 1-6
   RAM enable bit 1-6
   Sleep/standby mode 1-6
Emulator limitations and restrictions 4-4
Emulator specific command syntax B-1
Emulator status 4-12
es Command 4-12

F  Foreground monitor
   address requirements 4-9
   to use single step command 4-9
Foreground monitors A-2
   example of using A-3
   select an appropriate monitor A-3
   single-step processor A-6
H  Halting the analyzer 2-22
    Help 2-6
    help Command 2-6

I  Information help 2-6
    init Command 2-7
    Initializing the Emulator 2-7
    Installing target system probe
        target system probe 3-2
    Internal memory access 4-14
    Interrupts
        enable/disable from target system 4-10
    io Command B-7
    io Command (syntax) B-7

L  Limitations
    DMA 4-4
        monitor break at sleep mode 4-16
        monitor break at standby mode 4-16
        Watch dog timer in background 4-16
    load Command 2-15
    Loading programs 2-11
        for Standalone Configuration 2-11
        for Transparent Configuration 2-13
        load command 2-15
        transfer utility 2-13

M  m Command 2-11, 2-19
    map Command 2-10, 4-13
    Measurement commands 4-1
    Memory Display 2-19
        mnemonic format 2-16
    Memory mapping 4-13
        defining memory type to emulator 4-13
        for getting started program 2-10
        sequence of map/load commands 4-14
    Memory search 2-26
    Mnemonic display format 2-16
    Monitor
        select foreground/background monitor 4-8
    Monitors
        background A-1
        comparison of foreground/background A-1

Index-5
MOVFPE instruction B-7
MOVTPE instruction B-7

N Non-conductive pin guard
target system probe 3-2

P Predefining stack pointer 4-11
Prerequisites for using the emulator 2-2
Processor clock selection 4-5
Program loads 2-11
Program tracing 2-21
Prompts
emulator command 2-17
Purpose of the Emulator 1-1

R r Command 2-17 - 2-18
Real-time runs
restricting emulator to 4-10
reg Command 2-18
Register class B-8
Register Display 2-18
Register name B-8
Restrict to real time runs 4-10
permissible commands 4-10
target system dependency 4-11
rst Command 2-17

S s Command 2-20
Sample programs
for getting started 2-3
ser Command 2-26
Single step 2-20
in foreground monitor A-6
Sleep mode
unavailable commands 4-16
Software breakpoints 2-24, 4-15
defining in target ROM 2-24
Stack pointer
predefining 4-11
Standby mode
unavailable commands 4-16
Starting a trace 2-21
Storage qualifier 2-21
Syntax (command), specific to H8/536 emulator B-1
System commands 4-2

T  t Command 2-21
   Target system dependency on executing code 4-11
   Target system interrupts
   enable/disable 4-10
   Target system probe
   cautions for installation 3-2
   installation 3-2
   installation procedure 3-3
   non-conductive pin guard 3-2
   Target system reset 4-12
   tf Command 2-21
   tg Command 2-21
   th Command 2-22
   tl Command 2-22
   Trace list display 2-22
   Trace list format 2-21
   Tracing program execution 2-21
   Transfer utility 2-13
   Transparent mode 2-13
   Trigger signals
   break upon 4-15
   tsto Command 2-21
   Types of configuration 4-1

W  Watch dog timer
   in background monitor 4-16

X  xp Command 2-13