
HP 64784

H8/3003 Emulator Terminal Interface

User's Guide



HP Part No. 64784-97010

August 1995

Edition 6

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 1995, 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.

HP is a trademark of Hewlett-Packard Company.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

H8/3003™ is a registered trademark of Hitachi Ltd.

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 U.S.A. Right for non-DOD U.S. Government Department and Agencies are as set forth in FAR 52.227-19(c)(1,2).

Printing History

New editions are complete revisions of the manual. The date on the title page changes only when a new edition is published.

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

Edition 1	64784-97000, July 1993
Edition 2	64784-97002, March 1994
Edition 3	64784-97004, August 1994
Edition 4	64784-97006, April 1995
Edition 5	64784-97008, May 1995
Edition 6	64784-97010, Aug 1995

Using This Manual

This manual is designed to give you an introduction to the HP 64784 H8/3003 Emulator. This manual will also help define how these emulators differ from other HP 64700 Emulators.

This manual will:

- give you an introduction to using the emulator
- explore various ways of applying the emulator to accomplish your tasks
- show you emulator commands which are specific to the H8/3003 Emulator

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 H8/3003 emulator features and how they can help you in developing new hardware and software.
- Chapter 2** A brief introduction to using the H8/3003 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** H8/3003 Emulator Specific Command Syntax and Error Message

Contents

1 Introduction to the H8/3003 Emulator

Introduction	1-1
Purpose of the H8/3003 Emulator	1-1
Features of the H8/3003 Emulator	1-3
Supported Microprocessors	1-3
Clock Speeds	1-5
Emulation memory	1-7
Analysis	1-7
Registers	1-7
Breakpoints	1-7
Reset Support	1-8
Real Time Operation	1-8
Limitations, Restrictions	1-9
Foreground Monitor	1-9
DMA Support	1-9
Internal RAM of H8/3005	1-9
Watch Dog Timer in Background	1-9
Monitor Break at Sleep/Standby Mode	1-9
Hardware Standby Mode	1-9
Interrupts in Background Cycles	1-9
Reset Output Enable Bit	1-9
Evaluation chip	1-10

2 Getting Started

Introduction	2-1
Before You Begin	2-2
A Look at the Sample Program	2-3
Using the Help Facility	2-7
Initialize the Emulator to a Known State	2-8
Set Up the Proper Emulation Configuration	2-9
Set Up Emulation Conditions	2-9
Mapping Memory	2-11

Transfer Code into Emulation Memory	2-12
Transferring Code from a Terminal In Standalone Configuration	2-12
Transferring Code From A Host, HP 64700 In Transparent Configuration	2-15
Looking at Your Code	2-18
Familiarize Yourself with the System Prompts	2-19
Running the Sample Program	2-20
Stepping Through the Program	2-22
Tracing Program Execution	2-22
Predefined Trace Labels	2-22
Predefined Status Equates	2-23
Specifying a Trigger	2-23
Using Software Breakpoints	2-27
Displaying and Modifying the Break Conditions	2-27
Defining a Software Breakpoint	2-28
Searching Memory for Strings or Numeric Expressions	2-29
Making Program Coverage Measurements	2-29
Trace Analysis Considerations	2-30
How to Specify the Trigger Condition	2-30
Store Condition and Disassembling	2-32
Triggering the Analyzer by Data	2-34
3 In-Circuit Emulation	
Installing the Target System Probe	3-2
QFP adaptor	3-3
PGA adaptor	3-3
QFP socket/adaptor	3-4
Installing the QFP Adaptor	3-5
Installing the 64784E PGA adaptor	3-6
Installing the H8/3003 microprocessor	3-9
Using Low Voltage Adaptor	3-10
Specification	3-10
Installing the 64797B PGA adaptor	3-11
Run from Target System Reset	3-12
Electrical Characteristics	3-13
Target System Interface	3-26
4 Configuring the H8/3003 Emulator	
Types of Emulator Configuration	4-1
Emulation Processor to Emulator/Target System	4-1

Commands Which Perform an Action or Measurement	4-2
Coordinated Measurements	4-2
Analyzer	4-2
System	4-2
Emulation Processor to Emulator/Target System	4-3
cf ba	4-4
cf dbc	4-7
cf drst	4-8
Memory Mapping	4-15
Break Conditions	4-18
Where to Find More Information	4-19

A H8/3003 Emulator Specific Command Syntax

CONFIG_ITEMS	A-2
Summary	A-2
Syntax	A-2
Description	A-3
Examples	A-4
Related information	A-4
ACCESS MODE and DISPLAY MODE	A-4
Summary	A-4
Syntax	A-4
Defaults	A-5
Related Information	A-5
ADDRESS	A-6
Summary	A-6
Description	A-6
Examples	A-6
REGISTER CLASS and NAME	A-7
Summary	A-7
Emulator Specific Error Messages	A-15
Message	A-15

Illustrations

Figure 1-1. HP 64784 Emulator for the H8/3003	1-2
Figure 2-1. Sample Program Listing	2-5
Figure 3-1. Installing the QFP adaptor	3-5
Figure 3-2 Installing the PGA adaptor (General)	3-6
Figure 3-3 Installing the PGA adaptor (3001 mode 3/4)	3-7
Figure 3-4 Installing the PGA adaptor (3004/5 mode 3)	3-8
Figure 3-5 Installing the H8/3003 microprocessor	3-9
Figure 3-6 Installing the PGA adaptor (General)	3-11

Tables

Table 1-1. Supported Microprocessors	1-3
Table 1-2. Clock Speeds	1-6
Table 3-1. DC Characteristics of input high voltage	3-10
Table 3-2. Bus timing ($V_{cc} = 5.0V$, $f = 16MHz$)	3-13
Table 3-3. Refresh controller timing ($V_{cc} = 5.0V$, $f = 16MHz$)	3-16
Table 3-4. Control signal timing ($V_{cc} = 5.0V$, $f = 16MHz$)	3-17
Table 3-5. Timing condition of On-chip supporting modules ($V_{cc} = 5.0V$, $f = 16MHz$)	3-18
Table 3-6. Bus timing ($V_{cc} = 3.0V$, $f = 10MHz$)	3-20
Table 3-7. Control signal timing ($V_{cc} = 3.0V$, $f = 10MHz$)	3-23
Table 3-8. Timing condition of On-chip supporting modules ($V_{cc} = 3.0V$, $f = 10MHz$)	3-24
Table 4-1. Clock Speeds	4-6



Introduction to the H8/3003 Emulator

Introduction

The topics in this chapter include:

- Purpose of the H8/3003 Emulator
- Features of the H8/3003 Emulator

Purpose of the H8/3003 Emulator

The H8/3003 Emulator is designed to replace the H8/3003 microprocessor in your target system so you can control operation of the microprocessor in your application hardware (usually refer to as the *target system*). The H8/3003 emulator performs just like the H8/3003 microprocessor, but is a device that allows you to control the H8/3003 microprocessor directly. These features allow you to easily debug software before any hardware is available, and ease the task of integrating hardware and software.

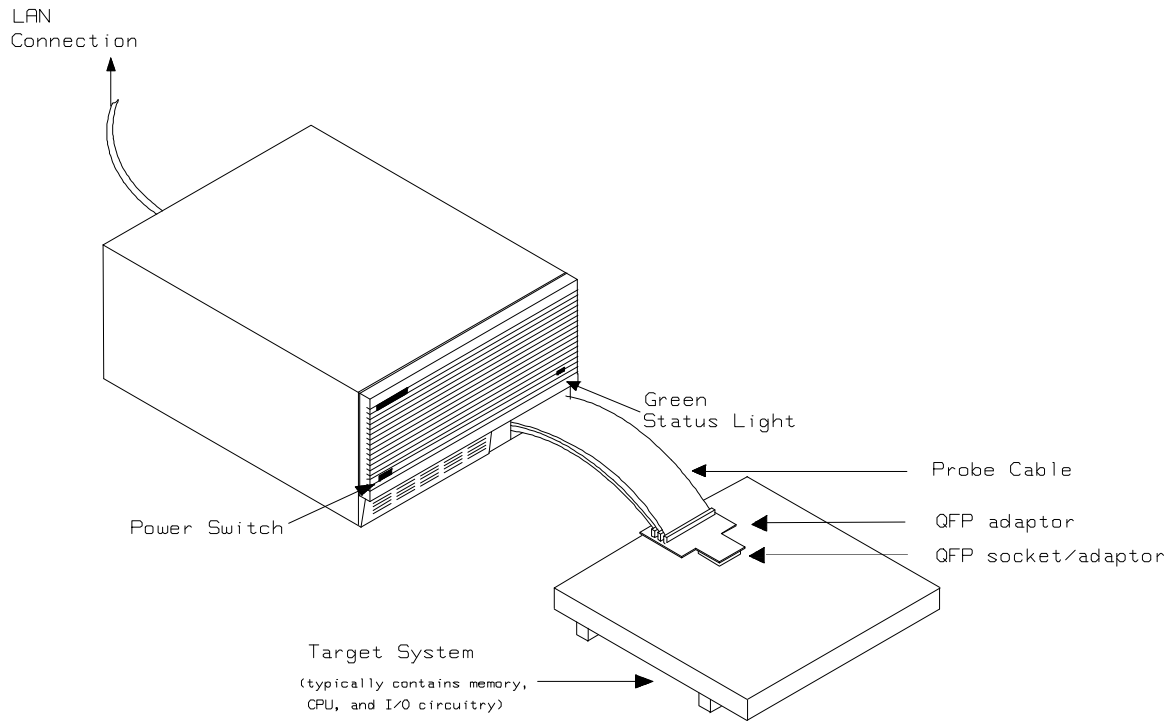


Figure 1-1. HP 64784 Emulator for the H8/3003

1-2 Introduction to the H8/3003 Emulator

Features of the H8/3003 Emulator

Supported Microprocessors

The HP 64784A H8/3003 emulator supports the microprocessors listed in Table 1-1.

Table 1-1. Supported Microprocessors

Supported Microprocessor					QFP Adaptor Board	PGA Adaptor Board/ QFP probe
Type	Package	System Clock Divider	On-chip ROM	Supply Voltage		
H8/3005	80pinQFP	-	-	4.75 to 5.25V	-	HP64784E/ HP64784K*1
				2.7 to 5.25V		
H8/3004	80pinQFP	-	-	4.75 to 5.25V	-	HP64784E/ HP64784K*1
				2.7 to 5.25V		
H8/3003	112 pin QFP	1:2	-	4.75 to 5.25V	HP64784C	HP64784E/ HP64784F
				2.7 to 5.25V		
		-	-	4.75 to 5.25V		
				2.7 to 5.25V		
H8/3002	100 pin QFP	-	-	4.75 to 5.25V	HP64784D	HP64784E/ HP64784G
				2.7 to 5.25V		
H8/3001	80 pin QFP	-	-	4.75 to 5.25V	-	HP64784E/ HP64784J *1

Table 1-1. Supported Microprocessors (Cont'd)

Supported Microprocessor					QFP Adaptor Board	PGA Adaptor Board/ QFP probe
Type	Package	System Clock Divider	On-chip ROM	Supply Voltage		
				2.7 to 5.25V		
H8/3032	80 pin QFP	-	PROM	4.75 to 5.25V	-	HP64784E/ HP64784H
				2.7 to 5.25V		
			Masked ROM	4.75 to 5.25 V		
				2.7 to 5.25V		
H8/3031	80 pin QFP	-	-	4.75 to 5.25 V	-	HP64784E/ HP64784H
				2.7 to 5.25 V		
H8/3030	80 pin QFP	-	-	4.75 to 5.25 V	-	HP64784E/ HP64784H
				2.7 to 5.25 V		
H8/3042	100 pin QFP	-	PROM	4.75 to 5.25V	HP64784D	HP64784E/ HP64784G
				2.7 to 5.25V		
			Masked ROM	4.75 to 5.25V		
				2.7 to 5.25V		
H8/3041	100 pin QFP	-	Masked ROM	4.75 to 5.25V	HP64784D	HP64784E/ HP64784G
				2.7 to 5.5V		
H8/3040	100 pin QFP	-	Masked ROM	4.752 to 5.25V	HP64784D	HP64784E/ HP64784G
				2.7 to 5.25V		

*1 When you do in-circuit emulation for H8/3001 with mode 3/4 or H8/3004/5 with mode 3, you must use HP 64784-66509 shipped with HP 64784J/K. Refer to the "In-Circuit Emulation" Chapter in this manual for more details.

1-4 Introduction to the H8/3003 Emulator

The H8/3003 emulator is provided without any QFP adaptors and PGA adaptor(HP 64784E) with QFP probe. To emulate each processor with your target system, you need to purchase appropriate QFP adaptor or PGA adaptor with QFP probe listed in Table 1-1. To purchase them, contact your local HP sales representative.



You can buy HP 64797B low voltage adaptor to emulate each processor running with supply voltage from 2.7 up to 5.25V input in your target system. To buy HP 64797B, contact your local HP sales representative.

The list of supported microprocessors in Table 1-1 is not necessarily complete. To determine if your microprocessor is supported or not, contact Hewlett-Packard.

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. When you select a clock input conforming to the specification of Table 1-2.

Refer to the "Configuration the Emulator" Chapter in this manual for more details.

Table 1-2. Clock Speeds

Clock source	Chip	Without 64797B	With 64797B
Internal	H8/3001 H8/3002 H8/3003T H8/3004 H8/3005 H8/3030 H8/3031 H8/3032 H8/3040 H8/3041 H8/3042	16MHz (System clock)	8MHz (System clock)
	H8/3003 with system clock divider	8MHz (System clock)	8MHz (System clock)
External	H8/3001 H8/3002 H8/3003T H8/3004 H8/3005 H8/3030 H8/3031 H8/3032 H8/3040 H8/3041 H8/3042	From 0.5 up to 16MHz (System clock)	From 0.5 up to 10MHz (System clock)
	H8/3003 with system clock divider	From 1 up to 24MHz (System clock is from 0.5 up to 12MHz)	From 1 up to 20MHz (System clock is from 0.5 up to 10MHz)

Emulation memory

The H8/3003 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
- HP 64729A 2M byte Emulation Memory Card

You can define up to 16 memory ranges (at 512 byte boundaries and least 512 byte in length.) The emulator occupies 6K byte, which is used for monitor program and internal RAM of microprocessor mapped as emulation RAM, leaving 122K, 506K, 1018K, 2042K byte of emulation memory which you may use. You can characterize memory range as emulation RAM (eram), emulation ROM (erom), target system RAM (tram), target system ROM (trom), or guarded memory (grd). The emulator generates an error message when accesses are made to guarded memory locations. You can also configure the emulator so that writes to memory defined as ROM cause emulator execution to break out of target program execution.

Analysis

The H8/3003 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 64794A/C/D Deep Emulation Bus Analyzer

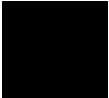
The Emulation Bus Analyzer monitors the emulation processor using an internal analysis bus. The HP 64703A 64-channel Emulation Bus Analyzer and 16-channel State/Timing Analyzer allows you to probe up to 16 different lines in your target system.

Registers

You can display or modify the H8/3003 internal register contents. This includes the ability to modify the program counter (PC) value so you can control where the emulator starts a program run.

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. This feature is realized by inserting a special instruction into user



program. One of undefined opcodes (5770 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 modification of target system memory, load/dump of target memory, display or modification of registers.

Limitations, Restrictions



Foreground Monitor	Foreground monitor is not supported for the H8/3003 emulator.
DMA Support	Direct memory access to the emulation by external DMAC is not allowed.
Internal RAM of H8/3005	When you emulate H8/3005 processor, you can't use address 0fef10h - 0ff00fh (mode 1) and 0ffef10h - 0fff00fh (mode 3) as internal RAM. These area are worked as external 8bit 3state area.
Watch Dog Timer in Background	Watch dog timer is suspended count up while the emulator is running in background monitor.
Monitor Break at Sleep/Standby Mode	When the emulator breaks into the background monitor, sleep or software standby mode is released. Then, PC indicates next address of "SLEEP" instruction.
Hardware Standby Mode	Hardware standby mode is not supported for the H8/3003 emulator. Hardware standby request from target system will give the emulator reset signal.
Interrupts in Background Cycles	The H8/3003 emulator does not accept any interrupts while in background monitor. Such interrupts are suspended while running the background monitor, and will occur when context is changed to foreground.
Reset Output Enable Bit	The RSTOE (Reset output enable bit) is used to determine whether the H8/3003 processor outputs reset signal when the processor is reset by the watch dog timer. However, the H8/3003 emulator ignores the configuration of the RSTOE, and works as it is configured with the modify configuration command.



Evaluation chip

Hewlett-Packard makes no warranty of the problem caused by the H8/3003 Evaluation chip in the emulator.

Getting Started

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 H8/3003 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

Msg_A, Msg_B and Msg_I are the messages used by the program to respond to various command inputs.

Initialization

The locations of stack and input area(Cmd_Input) are moved into address registers for use by the program. Next, the CLEAR routine clears the command byte(the first location pointed to by Cmd_Input - 0ff800 hex). Cmd_Input contains 00 hex for late use.

Scan

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

Exe_Cmd

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.

Cmd_A, Cmd_B, Cmd_I

These routines set up the proper parameters for writing the output message: the number of bytes in the message is moved to the R3L register and the base address of the message in the data area is moved to address register ER4.

Write_Msg

First the base address of the output area is copied to ER5. 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 Write_Loop routine. When done, Write_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.


```

002000          1          .SECTION          Table,DATA,LOCATE=H'2000
002000          2      Msgs
002000 5448495320495320 3      Msg_A          .SDATA          "THIS IS MESSAGE A"
002008 4D45535341474520
002010 41
002011 5448495320495320 4      Msg_B          .SDATA          "THIS IS MESSAGE B"
002019 4D45535341474520
002021 42
002022 494E56414C494420 5      Msg_I          .SDATA          "INVALID COMMAND"
00202A 434F4D4D414E44
002031          6      End_Msgs
          7
001000          8          .SECTION          Prog,CODE,LOCATE=H'1000
          9      ;*****
10      ;* Set up the Pointers.
11      ;*****
001000 7A07000FF904 12      Init          MOV.L          #Stack,ER7
001006 7A01000FF800 13      MOV.L          #Cmd_Input,ER1
          14      ;*****
          15      ;* Clear previous command.
          16      ;*****
00100C F800          17      Clear          MOV.B          #H'00,R0L
00100E 6AA8000FF800 18      MOV.B          R0L,@Cmd_Input
          19      ;*****
          20      ;* Read command input byte. If no command has been
          21      ;* entered, continue to scan for it.
          22      ;*****
001014 6A2A000FF800 23      Scan          MOV.B          @Cmd_Input,R2L
00101A AA00          24      CMP.B          #H'00,R2L
00101C 47F6          25      BEQ          Scan
          26      ;*****
          27      ;* A command has been entered. Check if it is
          28      ;* command A, command B, or invalid command.
          29      ;*****
00101E AA41          30      Exe_Cmd          CMP.B          #H'41,R2L
001020 5870000A          31      BEQ          Cmd_A
001024 AA42          32      CMP.B          #H'42,R2L
001026 58700010          33      BEQ          Cmd_B
00102A 58000018          34      BRA          Cmd_I
          35      ;*****
          36      ;* Command A is entered. R3L = the number of bytes
          37      ;* in message A. R4 = location of the message.
          38      ;* Jump to the routine which writes the message.
          39      ;*****
00102E FB11          40      Cmd_A          MOV.B          #Msg_B-Msg_A,R3L
001030 7A0400002000 41      MOV.L          #Msg_A,ER4
001036 58000014          42      BRA          Write_Msg
          43      ;*****
          44      ;* Command B is entered.
          45      ;*****
00103A FB11          46      Cmd_B          MOV.B          #Msg_I-Msg_B,R3L
00103C 7A0400002011 47      MOV.L          #Msg_B,ER4
001042 58000008          48      BRA          Write_Msg

```

Figure 2-1. Sample Program Listing

```

49 ;*****
50 ;* An invalid command is entered.
51 ;*****
001046 FB0F 52 Cmd_I      MOV.B      #End_Msgs-Msg_I,R3L
001048 7A0400002022 53      MOV.L      #Msg_I,ER4
54 ;*****
55 ;* The destination area is cleared.
56 ;*****
00104E 7A05000FF804 57 Write_Msg  MOV.L      #Msg_Dest,ER5
001054 FE20 58 Clear_Old  MOV.B      #H'20,R6L
001056 68D8 59 Clear_Loop MOV.B      R0L,@ER5
001058 0B05 60      ADDS.L     #1,ER5
00105A 1A0E 61      DEC.B      R6L
00105C 46F8 62      BNE       Clear_Loop
63 ;*****
64 ;* Message is written to the destination.
65 ;*****
00105E 7A05000FF804 66      MOV.L      #Msg_Dest,ER5
001064 6C4E 67 Write_Loop  MOV.B      @ER4+,R6L
001066 68DE 68      MOV.B      R6L,@ER5
001068 0B05 69      ADDS.L     #1,ER5
00106A 1A0B 70      DEC.B      R3L
00106C 46F6 71      BNE       Write_Loop
72 ;*****
73 ;* Go back and scan for next command.
74 ;*****
00106E 409C 75      BRA       Clear
76
0FF800 77      .SECTION   Data,DATA,LOCATE=H'FF800
78 ;*****
79 ;* Command input area.
80 ;*****
0FF800 00000004 81 Cmd_Input  .RES.L     1
82 ;*****
83 ;* Destination of the command messages.
84 ;*****
0FF804 00000100 85 Msg_Dest   .RES.W     H'80
0FF904 86 Stack     .RES.W
00001000 87      .END      Init

```

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

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
hl        - highlevel commands (hp internal use only)
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
```

```
emul - emulation commands
-----
b.....break to monitor   cp....copy memory       mo....modes
bc....break condition     dump...dump memory      r.....run user code
bp....breakpoints         es....emulation status  reg...registers
cf....configuration       io....input/output      rst...reset
cim....copy target image  load...load memory      rx....run at CMB execute
cmb....CMB interaction    m.....memory           s.....step
cov....coverage           map....memory mapper    ser....search memory
```

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

```
R> help load
```

load - download absolute file into processor memory space

```
load -i      - download intel hex format
load -m      - download motorola S-record format
load -t      - download extended tek hex format
load -S      - download symbol file
load -h      - download hp format (requires transfer protocol)
load -a      - reserved for internal hp use
load -e      - write only to emulation memory
load -u      - write only to target memory
load -o      - data received from the non-command source port
load -s <str> - send a character string out the other port
load -b      - data sent in binary (valid with -h option)
load -x      - data sent in hex ascii (valid with -h option)
load -q      - quiet mode
load -p      - record ACK/NAK protocol (valid with -imt options)
load -c <file> - data is received from the 64000. file name format is:
               <filename>:<userid>:absolute
```

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
```

Set Up the Proper Emulation Configuration

Set Up Emulation Conditions

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=3042
cf clk=int
cf dbc=en
cf drst=dis
cf mode=7
cf nmi=en
cf rrt=dis
cf rsp=9
cf tdma=en
cf trfsh=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=3042
cf clk=ext
cf dbc=en
cf drst=dis
cf mode=7
cf nmi=en
cf rrt=en
cf rsp=9
cf tdma=en
cf trfsh=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=0ff904
```

4. Let's go ahead and set up the proper break conditions.

Type:

```
R> bc
```

You will see:

```
bc -d bp #disable
bc -e 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:

2-10 Getting Started

```
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)

Mapping Memory

Depending on the memory board, emulation memory consists of 128K, 512K, 1M or 2M bytes, mappable in 512 byte blocks. The monitor occupies 2K bytes and the emulator maps 4K bytes for internal RAM as emulation RAM automatically, leaving 122K, 506K, 1018K or 2042K bytes of emulation memory which you may use.

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

Type:

```
R> map 0..0ffff erom
```

To verify that memory blocks are mapped properly, type:

```
R> map
```

You will see:

```
# remaining number of terms      : 15
# remaining emulation memory    : 6e800h bytes
map 0000000..000ffff          erom   # term 1
map other tram
```

Note



You must map internal ROM as emulation memory.

Note



You don't have to map internal RAM, since the emulator maps internal RAM as emulation RAM. And the emulator memory system dose not introduce it in memory mapping display.

Refer to "Memory Mapping" section of "Configuring the Emulator" chapter in this manual for more details.

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 002000..00200f=54,48,49,53,20,49,53,20,4d,45,53,53,41,47,45,20
R> m 002010..00201f=41,54,48,49,53,20,49,53,20,4d,45,53,53,41,47,45
```



```
R> m 002020..00202f=20,42,49,4e,56,41,4c,49,44,20,43,4f,4d,4d,41,4e
R> m 002030=44
```

You could also type the following line instead:

```
R> m 002000="THIS IS MESSAGE ATTHIS IS MESSAGE BINVALID COMMAND"
```

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

```
R> m 002000..002030
```

You should see:

```
002000..00200f 54 48 49 53 20 49 53 20 4d 45 53 53 41 47 45 20
002010..00201f 41 54 48 49 53 20 49 53 20 4d 45 53 53 41 47 45
002020..00202f 20 42 49 4e 56 41 4c 49 44 20 43 4f 4d 4d 41 4e
002030..002030 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 002000..00200f shows these values:

```
002000..00200f 54 48 49 53 20 20 49 53 20 4d 45 53 53 41 47 45
```

you can correct this row of memory by typing:

```
R> m 002000..00200f=54,48,49,53,20,49,53,20,4d,45,53,53,41,47,45,20
```

Or, you might need to modify only one location, as in the instance where address 00200f equals 22 hex rather than 20 hex. Type:

```
R> m 00200f=22
```

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

(Note the hex letters must be preceded by a digit.)

```
R> m 001000..00100f=7a,07,00,0f,0f9,04,7a,01,00,0f, 0f8,00,0f8,00,6a,0a8
R> m 001010..00101f=00,0f,0f8,00,6a,2a,00,0f,0f8,00,0aa,00,47,0f6,0aa,41
R> m 001020..00102f=58,70,00,0a,0aa,42,58,70,00,10,58,00,00,18,0fb,11
R> m 001030..00103f=7a,04,00,00,20,00,58,00,00,14,0fb,11,7a,04,00,00
R> m 001040..00104f=20,11,58,00,00,08,0fb,0f,7a,04,00,00,20,22,7a,05
R> m 001050..00105f=00,0f,0f8,04,0fe,20,68,0d8,0b,05,1a,0e,46,0f8,7a,05
R> m 001060..00106f=00,0f,0f8,04,6c,4e,68,0de,0b,05,1a,0b,46,0f6,40,9c
```

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

R> **m 001000..00106f**

You should see:

```
001000..00100f 7a 07 00 0f f9 04 7a 01 00 0f f8 00 f8 00 6a a8
001010..00101f 00 0f f8 00 6a 2a 00 0f f8 00 aa 00 47 f6 aa 41
001020..00102f 58 70 00 0a aa 42 58 70 00 10 58 00 00 18 fb 11
001030..00103f 7a 04 00 00 20 00 58 00 00 14 fb 11 7a 04 00 00
001040..00104f 20 11 58 00 00 08 fb 0f 7a 04 00 00 20 22 7a 05
001050..00105f 00 0f f8 04 fe 20 68 d8 0b 05 1a 0e 46 f8 7a 05
001060..00106f 00 0f f8 04 6c 4e 68 de 0b 05 1a 0b 46 f6 40 9c
```

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.

Transferring Code From A Host, HP 64700 In Transparent Configuration

The method provided in this example assumes that you are running an 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 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:

```
login:  
login:  
login:
```

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 and link your code.
4. Convert your absolute file generated above into HP format with the following command. This is needed to load the file into the emulator.

```
$ h83cnvhp -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
```

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 2000..2030
```

You should see:

```
002000..00200F 54 48 49 53 20 49 53 20 4d 45 53 53 41 47 45 20
002010..00201F 41 54 48 49 53 20 49 53 20 4d 45 53 53 41 47 45
002020..00202F 20 42 49 4e 56 41 4c 49 44 20 43 4f 4d 4d 41 4e
002030..002030 44
```

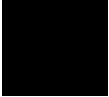
If your system does not match, verify 1) that you entered the source code correctly; 2) that you entered the linker parameters correctly.

Looking at Your Code

Now that you have loaded your code into emulation memory, you can display it in mnemonic format. Type:

```
R> m -dm 1000..106f
```

You will see:



```
0001000 - MOV.L #000ff904,ER7
0001006 - MOV.L #000ff800,ER1
000100c - MOV.B #00,R0L
000100e - MOV.B R0L,@0ff800
0001014 - MOV.B @0ff800,R2L
000101a - CMP.B #00,R2L
000101c - BEQ 001014
000101e - CMP.B #41,R2L
0001020 - BEQ 00102e
0001024 - CMP.B #42,R2L
0001026 - BEQ 00103a
000102a - BRA 001046
000102e - MOV.B #11,R3L
0001030 - MOV.L #00002000,ER4
0001036 - BRA 00104e
000103a - MOV.B #11,R3L
000103c - MOV.L #00002011,ER4
0001042 - BRA 00104e
0001046 - MOV.B #0f,R3L
0001048 - MOV.L #00002022,ER4
000104e - MOV.L #000ff804,ER5
0001054 - MOV.B #20,R6L
0001056 - MOV.B R0L,@ER5
0001058 - ADDS #1,ER5
000105a - DEC.B R6L
000105c - BNE 001056
000105e - MOV.L #000ff804,ER5
0001064 - MOV.B @ER4+,R6L
0001066 - MOV.B R6L,@ER5
0001068 - ADDS #1,ER5
000106a - DEC.B R3L
000106c - BNE 001064
000106e - BRA 00100c
```

Familiarize Yourself with the System Prompts

Note



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:

```
*> rst
```

You will see:

```
R>
```

The **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> r 1000
```

You will see:

```
U>
```

The **r** command runs the processor from address 1000 hex.

3. Type:

```
U> b
```

You will see:

```
M>
```

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.

Note



If DMA transfer is in progress with BURST transfer mode, **b** command is suspended and occurs after DMA transfer is completed.

Running the Sample Program

4. Type:

M> **r 1000**

The emulator changes state from background to foreground and begins running the sample program from location 1000 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=001014 ccr=84 er0=00000000 er1=000ff800 er2=00000000 er3=00000000
reg er4=00000000 er5=00000000 er6=00000000 er7=000ff904 sp=000ff904 mdcr=c7
```

Notice that ER1 contains 0ff800 hex.

6. Verify that the input area command byte was cleared during initialization.

Type:

```
U> m -db 0ff800
```

You will see:

```
00ff800..00ff800    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 0ff800=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 0ff804..0ff823
```

You will see:

```
00ff804..00ff813    54 48 49 53 20 49 53 20 4d 45 53 53 41 47 45 20
00ff814..00ff823    41 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

These are the ASCII values for `Msg_A`.

Repeat the last two commands twice. The first time, use 42 instead of 41 at location ff800h and note that `Msg_B` overwrites `Msg_A`. Then try these again, using any number except 00, 41, or 42 and note that the `Msg_I` 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 1000;reg
```

This command steps 1 instruction from address 1000 hex, and displays registers. You will see:

```
0001000 -      MOV.L #000ff904,ER7
PC =0001006
reg pc=001006 ccr=80 er0=00000000 er1=000ff800 er2=00000000 er3=00000000
reg er4=00000000 er5=00000000 er6=00000000 er7=000ff904 sp=000ff904 mdcrc=c7
```

Notice that PC contains 1006 hex.

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

```
M> s;reg
```

You will see:

```
0001006 -      MOV.L #000ff800,ER1
PC =000100c
reg pc=00100c ccr=80 er0=00000000 er1=000ff800 er2=00000000 er3=00000000
reg er4=00000000 er5=00000000 er6=00000000 er7=000ff904 sp=000ff904 mdcrc=c7
```

Tracing Program Execution

Predefined Trace Labels

Three trace labels are predefined in the H8/3003 emulator. You can view these labels by entering the tlb (trace label) command with no options.

```
M> tlb
```

```
#### Emulation trace labels
tlb addr 16..39
tlb data 0..15
tlb stat 40..57
```

Predefined Status Equates

Common values for the H8/3003 status trace signals have been predefined. You can view these predefined equates by entering the equ command with no options.

M> equ

```
### Equates ###
equ bg=0xxx0xxxxxxxxxxxxxxxxxy
equ byte=0xxxxxx1xxxx1xxxx1xy
equ cpu=0xxxxxx1xxxx11xxxxxy
equ data=0xxxxxx1xxxx1x1xxxxxy
equ dma=0xxxxxx1xxxx10xxxxxy
equ fetch=0xxxxxx1x1xx110xx01y
equ fg=0xxx1xxxxxxxxxxxxxxxxxy
equ grd=0xxx011xxxx1xx1xxxxy
equ intack=0xxxxxx0xxxxxxxxxxxxxy
equ io=0xxxxxx1xxxx1xx0xxxxy
equ mem=0xxxxxx1xxxx1xx1xxxxy
equ read=0xxxxxx1xxxx1xxxxx1y
equ refresh=0xxxxxx1xxxx01xxxxxy
equ word=0xxxxxx1xxxx1xxxx0xy
equ write=0xxxxxx1xxxx1xxxxx0y
equ wrrom=0xxxx101xxxx1xx1xx0y
```

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

Specifying a Trigger

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 ff804 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=0ff804
```

To store only the accesses to the address range ff804 through ff815 hex, type:

```
M> tsto addr=0ff804..0ff815
```

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 1000
```

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 ff804 hex). Type:

```
U> m 0ff800=41
```

To display the trace list, type:

```
U> t1 0..34
```

You will see:

Line	addr,H	data,A	count,R	seq
0	0ff804	..	---	+
1	0ff805	..	0.760 uS	.
2	0ff806	..	0.720 uS	.
3	0ff807	..	0.760 uS	.
4	0ff808	..	0.760 uS	.
5	0ff809	..	0.760 uS	.
6	0ff80a	..	0.720 uS	.
7	0ff80b	..	0.760 uS	.
8	0ff80c	..	0.760 uS	.
9	0ff80d	..	0.760 uS	.
10	0ff80e	..	0.720 uS	.
11	0ff80f	..	0.760 uS	.
12	0ff810	..	0.760 uS	.
13	0ff811	..	0.760 uS	.
14	0ff812	..	0.720 uS	.
15	0ff813	..	0.760 uS	.
16	0ff814	..	0.760 uS	.
17	0ff815	..	0.760 uS	.
18	0ff804	TT	12.00 uS	.
19	0ff805	HH	1.120 uS	.
20	0ff806	II	1.120 uS	.
21	0ff807	SS	1.120 uS	.
22	0ff808	..	1.120 uS	.
23	0ff809	II	1.120 uS	.
24	0ff80a	SS	1.120 uS	.
25	0ff80b	..	1.120 uS	.
26	0ff80c	MM	1.160 uS	.
27	0ff80d	EE	1.120 uS	.
28	0ff80e	SS	1.120 uS	.
29	0ff80f	SS	1.120 uS	.
30	0ff810	AA	1.120 uS	.
31	0ff811	GG	1.120 uS	.
32	0ff812	EE	1.120 uS	.
33	0ff813	..	1.120 uS	.
34				



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
```

Now display the trace list:

```
U> t1 0..34
```

You will see:

Line	addr,H	data,A	count,R	seq
0	0ff804	..	---	+
1	0ff805	..	0.760 uS	.
2	0ff806	..	0.720 uS	.
3	0ff807	..	0.760 uS	.
4	0ff808	..	0.760 uS	.
5	0ff809	..	0.760 uS	.
6	0ff80a	..	0.720 uS	.
7	0ff80b	..	0.760 uS	.
8	0ff80c	..	0.760 uS	.
9	0ff80d	..	0.760 uS	.
10	0ff80e	..	0.720 uS	.
11	0ff80f	..	0.760 uS	.
12	0ff810	..	0.760 uS	.
13	0ff811	..	0.760 uS	.
14	0ff812	..	0.720 uS	.
15	0ff813	..	0.760 uS	.
16	0ff814	..	0.760 uS	.
17	0ff815	..	0.760 uS	.
18	0ff804	TT	12.00 uS	.
19	0ff805	HH	1.120 uS	.
20	0ff806	II	1.120 uS	.
21	0ff807	SS	1.120 uS	.
22	0ff808	..	1.120 uS	.
23	0ff809	II	1.120 uS	.
24	0ff80a	SS	1.120 uS	.
25	0ff80b	..	1.120 uS	.
26	0ff80c	MM	1.160 uS	.
27	0ff80d	EE	1.120 uS	.
28	0ff80e	SS	1.120 uS	.
29	0ff80f	SS	1.120 uS	.
30	0ff810	AA	1.120 uS	.
31	0ff811	GG	1.120 uS	.
32	0ff812	EE	1.120 uS	.
33	0ff813	..	1.120 uS	.
34	0ff814	AA	1.160 uS	.

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

Using Software Breakpoints

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 (5770 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. (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 Write_Msg label.

```
M> bp 104e
```

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

```
M> r 1000
```

```
U> m 0ff800=41
```

```
!ASYNC_STAT 615! Software break point: 000104e
```

```
M> reg
```

```
reg pc=00104e ccr=80 er0=00000000 er1=000ff800 er2=00000041 er3=00000011
reg er4=00002000 er5=00000000 er6=00000000 er7=000ff904 sp=000ff904 mdcrc=c7
```

Notice that PC contains 104e.

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

```
M> bp
```

```
###BREAKPOINT FEATURE IS ENABLED###
bp 000104e #disabled
```

```
M> bp -e 104e
```

```
M> bp
```

```
###BREAKPOINT FEATURE IS ENABLED###
bp 000104e #enabled
```

```
M> r 1000
```

```
U> m 0ff800=41
```

```
!ASYNC_STAT 615! Software breakpoint: 000104e
```

```
M> bp
```

```
###BREAKPOINT FEATURE IS ENABLED###
bp 000104e #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> ser 2000..2fff="THIS IS MESSAGE A"
```

```
pattern match at address: 0002000
```

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

```
M> ser -db 10000..106f=0aa
```

```
pattern match at address: 000101a  
pattern match at address: 000101e  
pattern match at address: 0001024
```

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> cov -r
```

Note



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 1000
M> m 0ff800=41
M> m 0ff800=42
M> m 0ff800=43
```

Make the coverage measurement:

```
U> cov 1000..106f
```

```
percentage of memory accessed: % 100.0
```

Trace Analysis Considerations

There are some points you need to attend to in using the emulation analyzer. The following section describes such points.

How to Specify the Trigger Condition

Suppose that you would like to start the trace when the program begins executing Exe_Cmd routine.

To initialize the emulation analyzer, type:

```
U> tinit
```

To set the trigger condition, type:

```
U> tg addr=101e
```

Start the trace and modify memory so that the program will jump to the Exe_Cmd routine:

```
U> t
U> m 0ff800=41
```

To display the trace list, type:

U> **t1 0..20**

Line	addr,H	H8/3042 mnemonic,H	count,R	seq
0	00101e	aa41 fetch mem	---	+
1	001014	MOV.B @0ff800,R2L	0.120 uS	.
2	001016	000f fetch mem	0.120 uS	.
3	001018	f800 fetch mem	0.120 uS	.
4	00101a	CMP.B #00,R2L	0.120 uS	.
5	0ff800	00xx read mem byte	0.120 uS	.
6	00101c	BEQ 001014	0.120 uS	.
7	00101e	aa41 fetch mem	0.160 uS	.
8	001014	MOV.B @0ff800,R2L	0.120 uS	.
9	001016	000f fetch mem	0.120 uS	.
10	001018	f800 fetch mem	0.120 uS	.
11	00101a	CMP.B #00,R2L	0.120 uS	.
12	0ff800	00xx read mem byte	0.120 uS	.
13	00101c	BEQ 001014	0.120 uS	.
14	00101e	aa41 fetch mem	0.120 uS	.
15	001014	MOV.B @0ff800,R2L	0.160 uS	.
16	001016	000f fetch mem	0.120 uS	.
17	001018	f800 fetch mem	0.120 uS	.
18	00101a	CMP.B #00,R2L	0.120 uS	.
19	0ff800	00xx read mem byte	0.120 uS	.
20	00101c	BEQ 001014	0.120 uS	.

This is not what we were expecting to see. (We expected to see the program executed Exe_Cmd routine which starts from 101e hex.) As you can see at the first line of the trace list, address 101e hex appears on the address bus during the program executing Scan loop. This triggered the emulation analyzer before EXE_Cmd routine was executed. To avoid mis-trigger by this cause, set the trigger condition to the second instruction of the routine you want to trace. Type:

U> **tg addr=1020**

To change the trigger position so that 10 states appear before the trigger in the trace list, type:

U> **tp -b 10**

Start the trace again and modify memory:

U> **t**

U> **m 0ff800=41**

Now display the trace list:

U> **t1 -10..10**

As you can see, the analyzer captured the execution of Exe_Cmd routine which starts from line -2 of the trace list.

Line	addr,H	H8/3042 mnemonic,H	count,R	seq
-10	00101c	BEQ 001014	0.120 uS	.
-9	00101e	aa41 fetch mem	0.120 uS	.
-8	001014	MOV.B @0ff800,R2L	0.120 uS	.
-7	001016	000f fetch mem	0.120 uS	.
-6	001018	f800 fetch mem	0.120 uS	.
-5	00101a	CMP.B #00,R2L	0.160 uS	.
-4	0ff800	41xx read mem byte	0.120 uS	.
-3	00101c	BEQ 001014	0.120 uS	.
-2	00101e	CMP.B #41,R2L	0.120 uS	.
-1	001014	6a2a unused fetch mem	0.120 uS	.
0	001020	BEQ 00102e	0.120 uS	+
1	001022	000a fetch mem	0.120 uS	.
2	00102e	MOV.B #11,R3L	0.280 uS	.
3	001030	MOV.L #00002000,ER4	0.120 uS	.
4	001032	0000 fetch mem	0.120 uS	.
5	001034	2000 fetch mem	0.120 uS	.
6	001036	BRA 00104e	0.120 uS	.
7	001038	0014 fetch mem	0.120 uS	.
8	00104e	MOV.L #000ff804,ER5	0.240 uS	.
9	001050	000f fetch mem	0.160 uS	.
10	001052	f804 fetch mem	0.120 uS	.

Store Condition and Disassembling

When you specify store condition with `tsto` command, disassembling of program execution may not be accurate.

Type:

U> `tinit`

U> `t`

U> `t1 0..20`

Line	addr,H	H8/3042 mnemonic,H	count,R	seq
0	001016	000f fetch mem	---	+
1	001018	f800 fetch mem	0.120 uS	.
2	00101a	CMP.B #00,R2L	0.160 uS	.
3	0ff800	00xx read mem byte	0.120 uS	.
4	00101c	BEQ 001014	0.120 uS	.
5	00101e	aa41 fetch mem	0.120 uS	.
6	001014	MOV.B @0ff800,R2L	0.120 uS	.
7	001016	000f fetch mem	0.120 uS	.
8	001018	f800 fetch mem	0.120 uS	.
9	00101a	CMP.B #00,R2L	0.120 uS	.
10	0ff800	00xx read mem byte	0.160 uS	.
11	00101c	BEQ 001014	0.120 uS	.
12	00101e	aa41 fetch mem	0.120 uS	.
13	001014	MOV.B @0ff800,R2L	0.120 uS	.
14	001016	000f fetch mem	0.120 uS	.
15	001018	f800 fetch mem	0.120 uS	.
16	00101a	CMP.B #00,R2L	0.120 uS	.
17	0ff800	00xx read mem byte	0.120 uS	.
18	00101c	BEQ 001014	0.160 uS	.
19	00101e	aa41 fetch mem	0.120 uS	.
20	001014	MOV.B @0ff800,R2L	0.120 uS	.

The program is executing Scan loop.

Now, specify the store condition so that only accesses to the address range 1000 hex through 10ff hex will be stored:

U> **tsto addr=1000..10ff**

Start the trace and display the trace list:

U> **t**

U> **t1 0..20**

Line	addr,H	H8/3042 mnemonic,H	count,R	seq
0	00101c	BEQ 001014	---	+
1	00101e	aa41 fetch mem	0.120 uS	.
2	001014	MOV.B @0ff800,R2L	0.120 uS	.
3	001016	000f fetch mem	0.120 uS	.
4	001018	f800 fetch mem	0.120 uS	.
5	00101a	aa00 fetch mem	0.120 uS	.
6	00101c	BEQ 001014	0.280 uS	.
7	00101e	aa41 fetch mem	0.120 uS	.
8	001014	MOV.B @0ff800,R2L	0.120 uS	.
9	001016	000f fetch mem	0.120 uS	.
10	001018	f800 fetch mem	0.120 uS	.
11	00101a	aa00 fetch mem	0.120 uS	.
12	00101c	BEQ 001014	0.240 uS	.
13	00101e	aa41 fetch mem	0.120 uS	.
14	001014	MOV.B @0ff800,R2L	0.160 uS	.
15	001016	000f fetch mem	0.120 uS	.
16	001018	f800 fetch mem	0.120 uS	.
17	00101a	aa00 fetch mem	0.120 uS	.
18	00101c	BEQ 001014	0.240 uS	.
19	00101e	aa41 fetch mem	0.120 uS	.
20	001014	MOV.B @0ff800,R2L	0.160 uS	.

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

Triggering the Analyzer by Data

You may want to trigger the emulation analyzer when specific data appears on the data bus. You can accomplish this with the following command.

```
U> tg data=<data>
```

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

Location of data	Access size	Address value	Available <data> Specification
8 bit data bus area	byte/word	even	ddxx *1
		odd	xxdd *1
16 bit data bus area	byte	even	ddxx *1
		odd	xxdd *1
	word	even	hhll *2

*1 dd means 8 bits data

*2 hhll means 16 bits data

For example, to trigger the analyzer when the processor performs word access to data 1234 hex in 16 bit bus area, you can specify the following:

```
U> tg data=1234
```

To trigger the analyzer when the processor accesses data 12 hex to the even address located in 8 bit data bus area:

```
U> tg data=12xx
```

On the other hand, to trigger 12 hex to the odd address located 8 bit data bus.

```
U> tg data=0xx12
```

Notice that you always need to specify "xx" value to capture byte access to 8 bit data bus area. Be careful to trigger the analyzer by data.

You're now finished with the "Getting Started" example. You can proceed on with using the emulator and use this manual and the *Terminal Interface Reference* manual as needed to answer your questions.



Notes

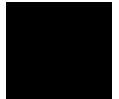


In-Circuit Emulation

When you are ready to use the H8/3003 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 H8/3003 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 H8/3003 emulator before attaching and detaching the QFP adaptor to the emulator or target system to avoid circuit damage resulting from voltage transients or mis-insertion of the QFP board.

Verify User Plug Orientation. Make certain that Pin 1 of the QFP socket/adaptor and Pin 1 of the QFP adaptor are properly aligned before inserting the QFP adaptor the QFP socket/adaptor. Failure to do so may result in damage to the emulator circuitry.

Protect Against Static Discharge. The H8/3003 emulator and the QFP adaptor contain devices which are susceptible to damage by static discharge. Therefore, operators should take precautionary measures before handling the user plug to avoid emulator damage.

Compatibility of VOLTAGE/CURRENCY. Please be sure to check that the voltage/currency of the emulator and target system being connected are compatible. If there is a discrepancy, damage may result.

Protect Target System CMOS Components. If your target system includes any CMOS components, turn on the target system first, then turn on the H8/3003 emulator; when powering down, turn off the emulator first, then turn off power to the target system.

The H8/3003 emulator is provided without any QFP adaptor or PGA adaptor with QFP probe. To emulate each processor with your target system, you need to purchase appropriate QFP adaptor or PGA adaptor with QFP probe.

QFP adaptor

To emulate each processor with your target system, you need to purchase appropriate adaptor listed in Table 1-1. The QFP adaptor allows you to connect the emulation probe to your target system using the QFP socket/adaptor provided with the QFP adaptor.

PGA adaptor

To emulate each processor with your target system, you need to use HP 64784E PGA adaptor as shown in Figure 3-2. The PGA adaptor allows you to connect the emulation probe to QFP socket/adaptor on your target system through the QFP probe listed in Table 1-1.

Caution



Do not apply strong force to QFP probe, as that might damage the QFP probe.



To emulate H8/3001 processor with mode 3/4 or H8/3004/05 processor with mode 3, you must use HP 64784-66509 board as shown in Figure 3-3 and 3-4. Connecting the emulator and your target system without this board causes serious damage, when you emulate in these cases.

Caution



Always detach HP 64784-66509 except you emulate H8/3001 processor with mode 3/4 or H8/3004/5 with mode 3.

QFP socket/adaptor

The QFP socket/adaptor is provided with the QFP adaptor and QFP probe, and designed for H8/3003 microprocessor. To do in-circuit emulation, you must attach the QFP socket/adaptor to your target system and connect with the QFP adaptor or PGA adaptor.

Note



You can order additional QFP socket/adaptor with part No. HP 64784-61611(for H8/3003), HP 64784-61612(for H8/3002/4x), HP 64784-61613(for H8/3004/05/3x) or HP 64784-61614(for H8/3001).

Installing the QFP Adaptor

1. Attach the QFP socket/adaptor to your target system.
2. Connect the QFP adaptor to the emulation probe.
3. Install the QFP adaptor to the QFP socket/adaptor on your target system as shown in Figure 3-1.

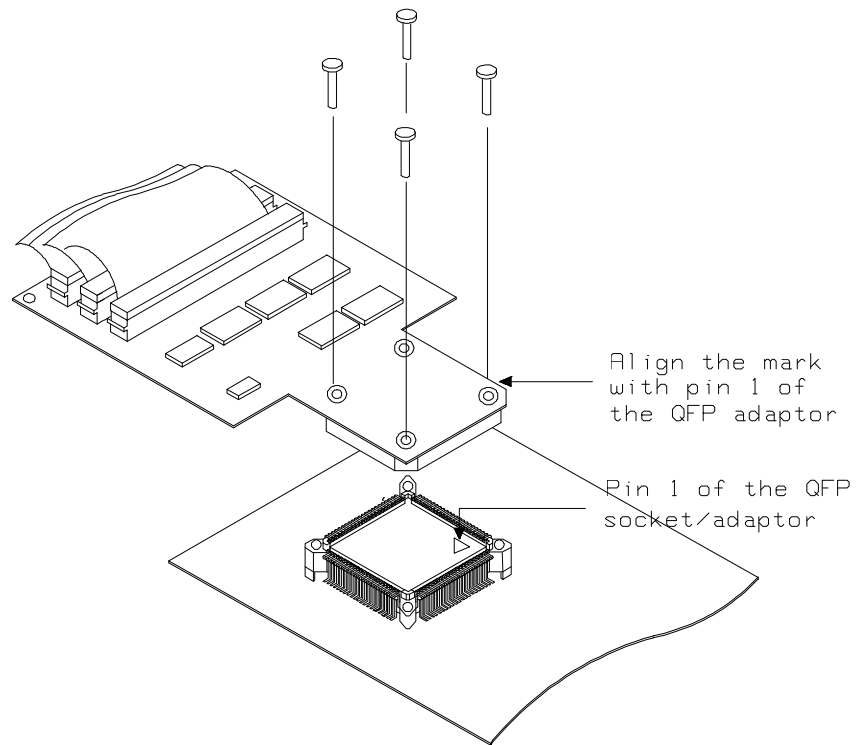


Figure 3-1. Installing the QFP adaptor

Installing the 64784E PGA adaptor

1. Attach the QFP socket/adaptor to your target system.
2. Connect the 64784E PGA adaptor to the emulation probe.
3. Install the 64784E PGA adaptor to the QFP socket/adaptor on your target system through QFP probe(or QFP probe and HP 64784-66509) as shown in Figure 3-2,3-3 and 3-4.

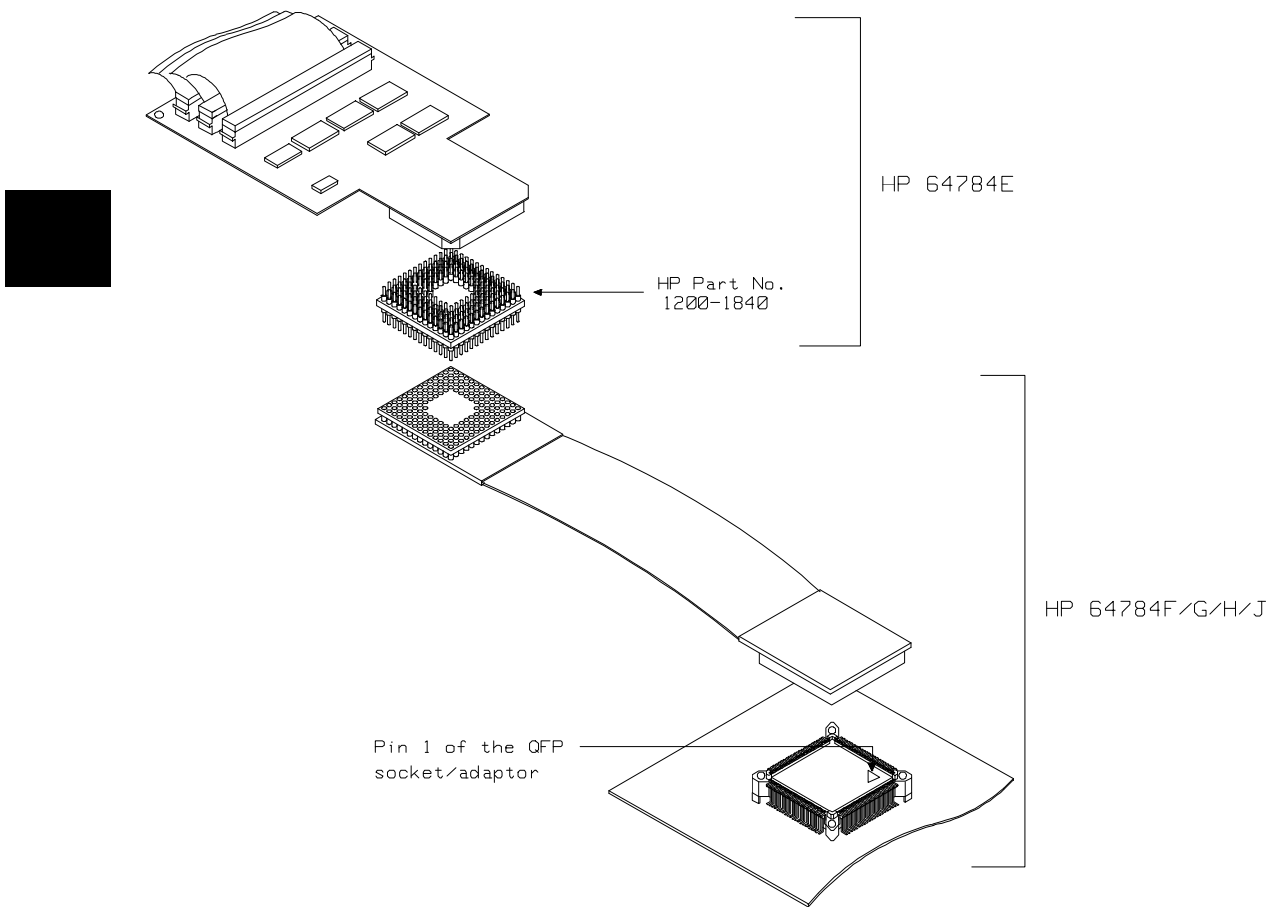


Figure 3-2 Installing the PGA adaptor (General)

3-6 In-Circuit Emulation

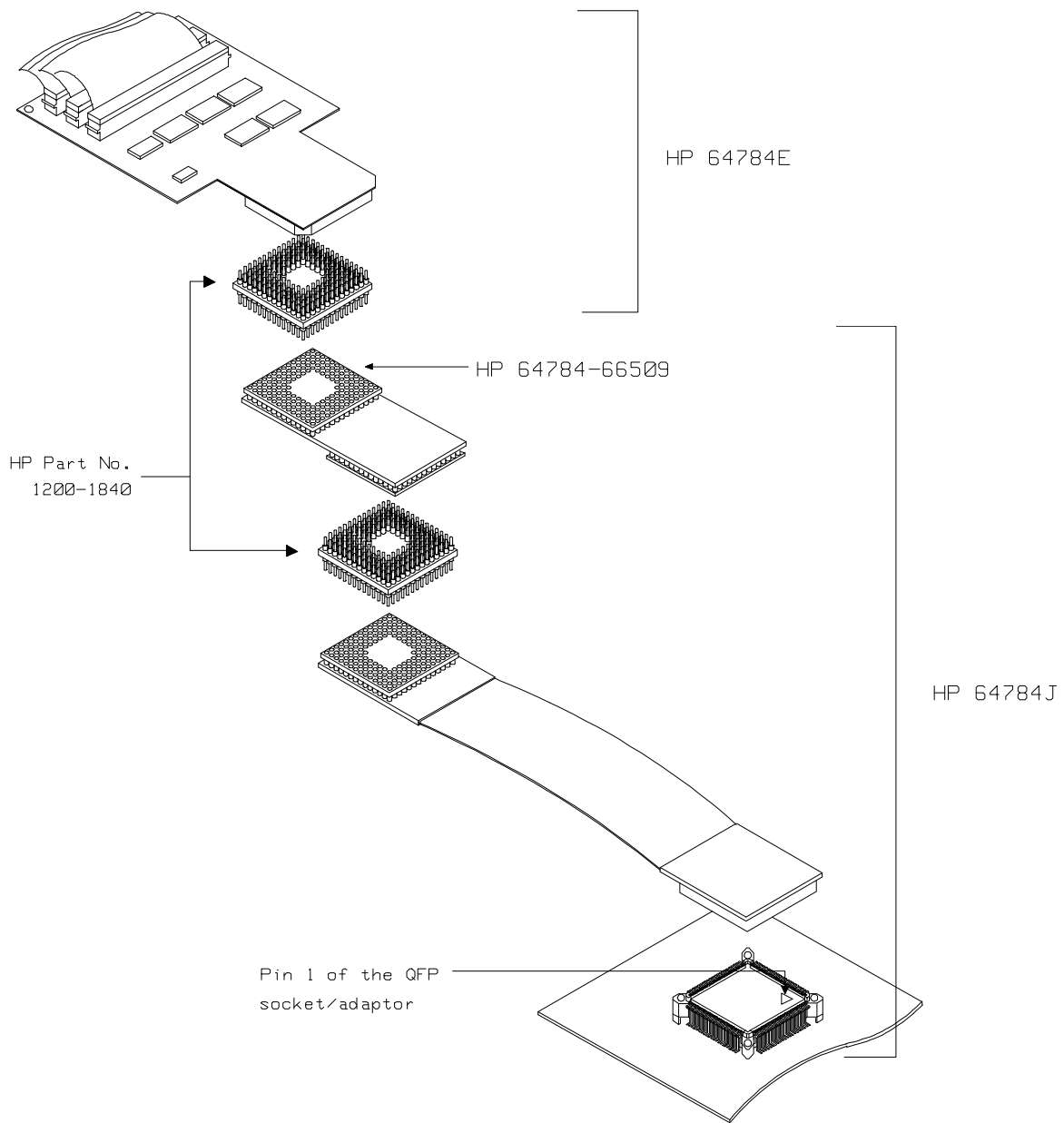


Figure 3-3 Installing the PGA adaptor (3001 mode 3/4)

3-7 In-Circuit Emulation

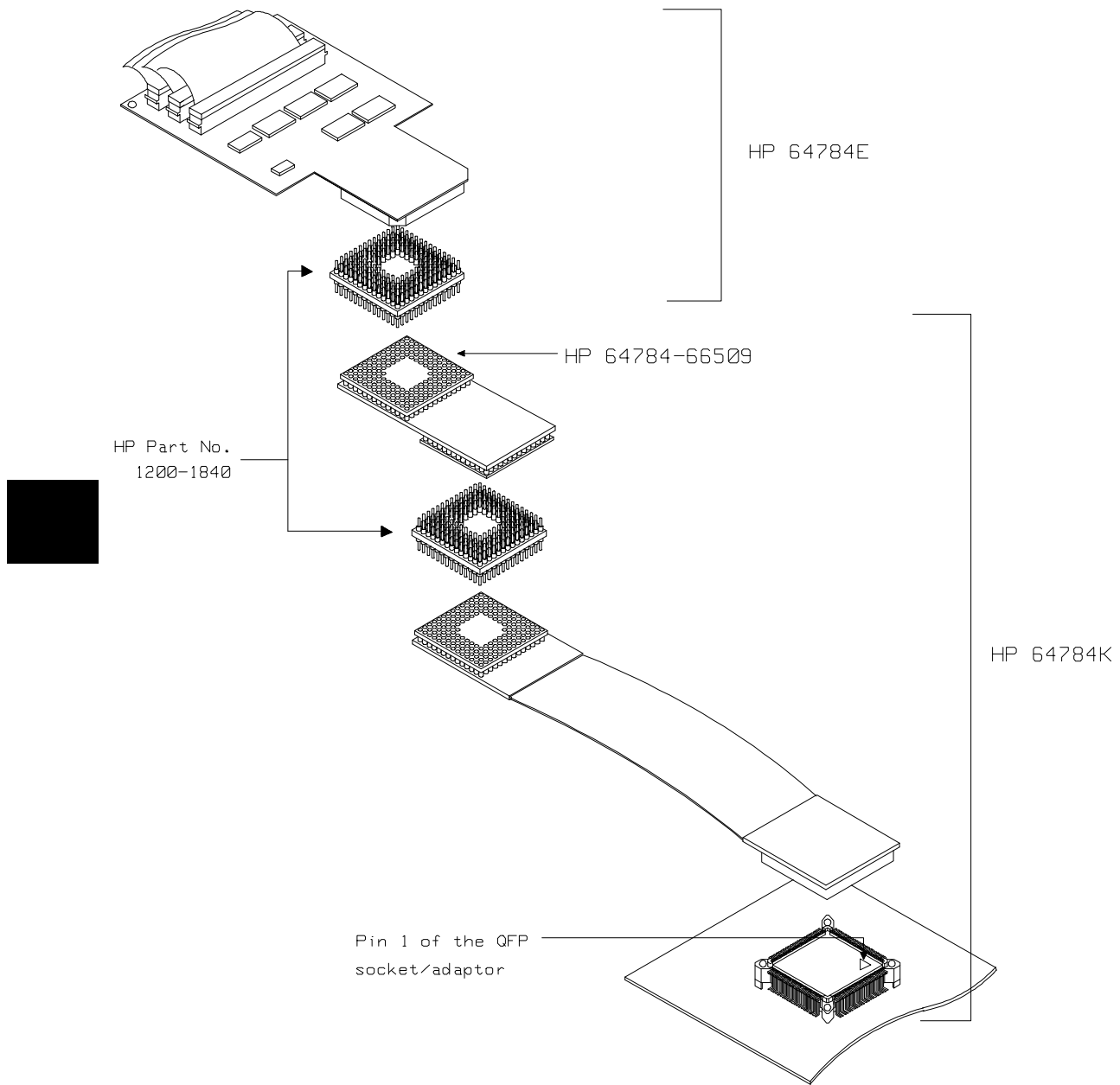


Figure 3-4 Installing the PGA adaptor (3004/5 mode 3)

3-8 In-Circuit Emulation

Installing the H8/3003 microprocessor

You can replace the QFP/PGA adaptor with H8/3003 microprocessor. Refer to the Figure 3-5.

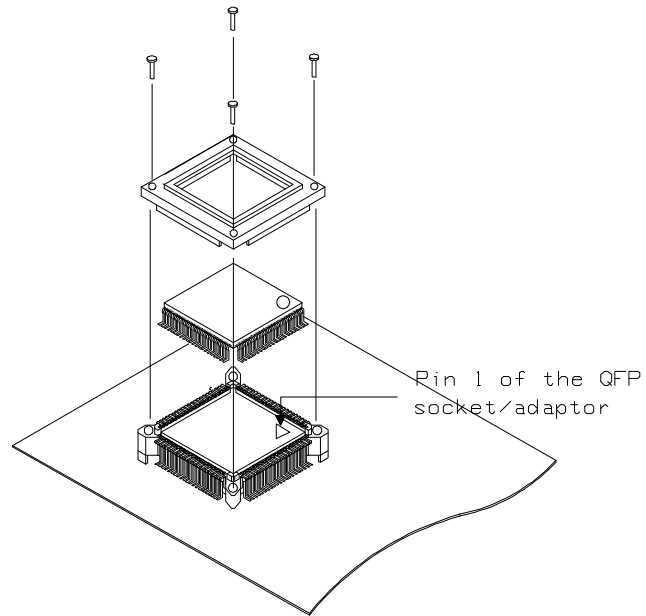


Figure 3-5 Installing the H8/3003 microprocessor

Using Low Voltage Adaptor

You can use optional low voltage adaptor with the H8/3003 emulator. The low voltage adaptor gives you a feature to emulate your target system running with supply voltage from 2.7V up to 5.25V.

Specification You must conform input high voltage(V_{ih}) to the specification of Table 3-1, when you use the low voltage adaptor with the H8/3003 emulator.

Table 3-1. DC Characteristics of input high voltage

Item	Minimum (V)
P1 - P5, D0 - D15	$V_{cc} \times 0.7$ or 2.4 *1
Others	$V_{cc} \times 0.7$ or 2.0 *1

*1 Higher of the two.

Note



This is different from the target processor's specification.

Note



You must also use a clock conforming to the specification of Table 4-1, when you use the low voltage adaptor and configure the emulator to use external clock.

Installing the 64797B PGA adaptor

1. Attach the QFP socket/adaptor to your target system.
2. Connect the 64797B PGA adaptor to the emulation probe.
3. Install the 64797B PGA adaptor to the QFP socket/adaptor on your target system through QFP probe(or QFP probe and HP 64784-66509) as shown in Figure 3-6.

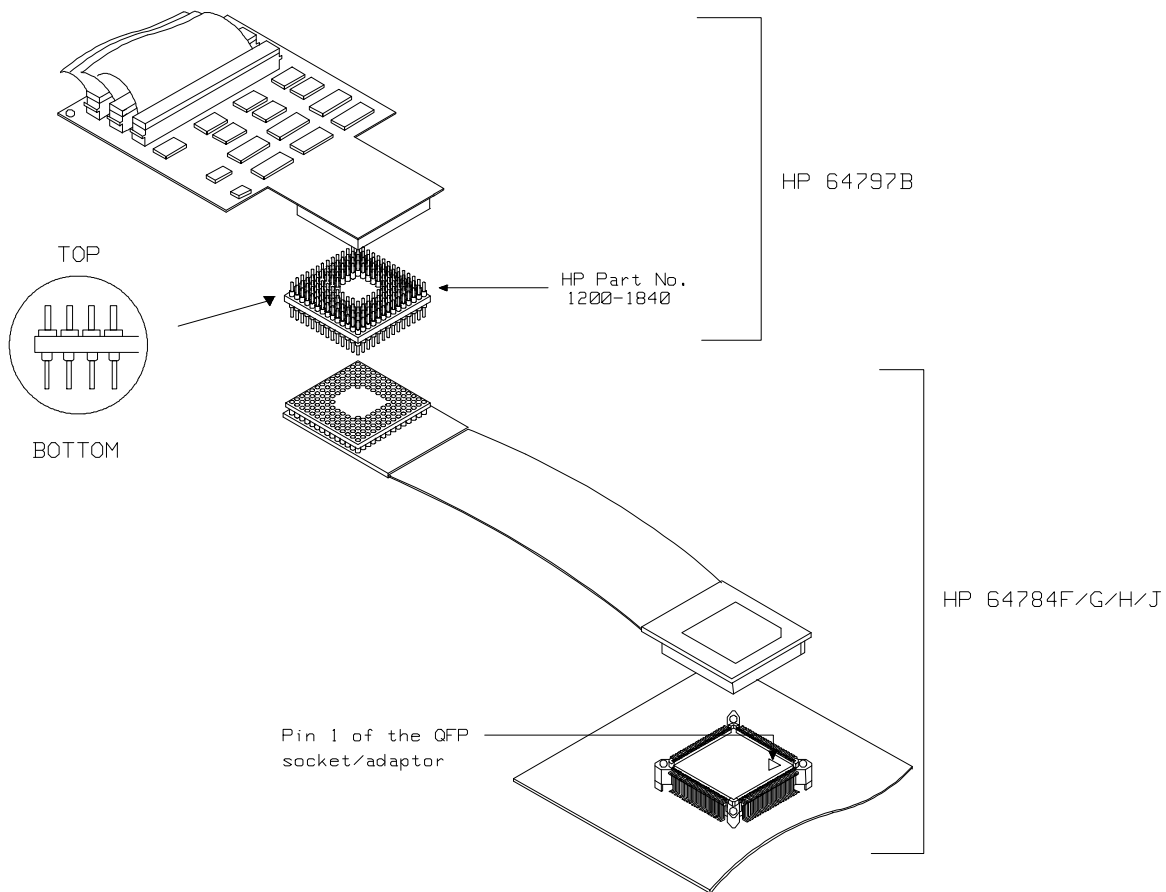


Figure 3-6 Installing the PGA adaptor (General)

Note



You have to use HP 64784-66509 when you emulate H8/3001 with mode 3/4 or H8/3004/05 with mode 3.

Run from Target System Reset

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

Note



In the "Awaiting target reset" status(**T>**), you can not break into the monitor. If you enter "**r rst**" in out-of-circuit or in the configuration that emulator does not accept target system reset (cf **trst=dis**), you must reset the emulator.

Electrical Characteristics

The AC characteristics of the HP 64784 H8/3003 emulator are listed in the following table.

Table 3-2. Bus timing (Vcc = 5.0V, f = 16MHz)

Characteristics	Symbol	H8/3003		Probe Type				Unit
		Vcc = 5V f = 16MHz		HP 64784E + HP 64784x		HP 64784D		
		min	max	typ *1	worst	typ *1	worst	
Clock cycle time	t _{cyc}	62.5	500	-	-	-	-	ns
Clock pulse low width	t _{CL}	20	-	23.2	12.9	23.8	13.3	ns
Clock pulse high width	t _{CH}	20	-	28.3	12.9	28.3	13.3	ns
Clock rise time	t _{CR}	-	10	5.6	17.1	5.0	16.7	ns
Clock fall time	t _{CF}	-	10	5.4	17.1	5.4	16.7	ns
Address delay time	t _{AD}	-	30	24.0	33.5	23.6	32.0	ns
Address hold time	t _{AH}	10	-	41.0	-4.8	41.2	-5.2	ns
Address strobe delay time	t _{ASD}	-	30	7.4	35.8	6.6	34.7	ns
Write strobe delay time	t _{WSD}	-	30	10.0	35.8	9.6	34.7	ns
Strobe delay time	t _{SD}	-	30	2.8	35.8	3.0	34.7	ns
Write data strobe pulse width 1	t _{WSW1}	35	-	53.0	28.6	53.8	29.1	ns
Write data strobe pulse width 2	t _{WSW2}	65	-	83.0	58.6	83.8	59.1	ns

Table 3-2. Bus timing (Vcc = 5.0V, f = 16MHz) (Cont'd)

Characteristics	Symbol	H8/3003		Probe Type				Unit
		Vcc = 5V f = 16MHz		HP 64784E + HP 64784x		HP 64784D		
		min	max	typ *1	worst	typ *1	worst	
Address setup time 1	tAS1	10	-	9.8	-24.1	10.6	-23.2	ns
Address setup time 2	tAS2	40	-	41.4	5.9	42.2	6.8	ns
Read data setup time	tRDS	20	-	30.4	49.7	24.0	47.3	ns
Read data hold time	tRDH	0	-	-27.0	29.7	-20.6	33.3	ns
Write data delay time	tWDD	-	60	29.0	68.7	29.8	69.9	ns
Write data setup time 1	tWDS1	35	-	31.0	-4.8	30.0	-6.6	ns
Write data setup time 2	tWDS2	5	-	3.2	-36.0	2.4	-37.8	ns
Write data hold time	tWDH	20	-	47.2	4.4	47.6	5.8	ns
Read data access time 1	tACC1	-	55	42.4	21.3	51.2	24.8	ns
Read data access time 2	tACC2	-	115	104.0	83.8	112.8	87.2	ns
Read data access time 3	tACC3	-	25	25.8	-12.3	33.4	-9.2	ns
Read data access time 4	tACC4	-	85	87.2	50.2	94.8	53.3	ns
Pre-charge time	tPCH	35	-	62.8	28.6	62.8	29.1	ns

3-14 In-Circuit Emulation

Table 3-2. Bus timing (Vcc = 5.0V, f = 16MHz) (Cont'd)

Characteristics	Symbol	H8/3003		Probe Type				Unit
		Vcc = 5V f = 16MHz		HP 64784E + HP 64784x		HP 64784D		
		min	max	typ *1	worst	typ *1	worst	
WAIT setup time	tWTS	25	-	26.2	60.9	22.6	54.7	ns
WAIT set hold time	tWTH	5	-	-23.6	-11.0	-20.0	-6.1	ns
BREQ setup time	tBRQS	40	-	-	75.9	-	69.7	ns
BACK delay time 1	tBACD1	-	30	10.4	35.8	10.0	34.7	ns
BACK delay time 2	tBACD2	-	30	-4.2	35.8	-4.0	34.7	ns
Bus floating time	tBZD	-	40	19.0	46.2	20.4	44.7	ns

*1 Typical outputs measured with 50pF load

Table 3-3. Refresh controller timing (Vcc = 5.0V, f = 16MHz)

Characteristics	Symbol	H8/3003		Probe Type				Unit
		Vcc = 5V f = 16MHz		HP 64784E + HP 64784x		HP 64784D		
		min	max	typ *1	worst	typ *1	worst	
$\overline{\text{RAS}}$ delay time 1	tRAD1	-	30	23.6	41.6	17.8	39.0	ns
$\overline{\text{RAS}}$ delay time 2	tRAD2	-	30	22.2	41.6	16.2	39.0	ns
$\overline{\text{RAS}}$ delay time 3	tRAD3	-	30	8.0	41.6	6.6	39.0	ns
Row address hold time	tRAH	15	-	22.0	-10.5	27.0	-9.5	ns
$\overline{\text{RAS}}$ pre-charge time	tRP	35	-	60.8	25.8	61.4	26.7	ns
$\overline{\text{CAS}}$ to $\overline{\text{RAS}}$ pre-charge time	tCRP	35	-	61.4	28.6	60.2	29.1	ns
$\overline{\text{CAS}}$ pulse width	tCAS	40	-	52.8	33.6	53.2	34.1	ns
$\overline{\text{RAS}}$ access time	tRAC	-	85	74.4	44.4	82.8	49.0	ns
Address access time	tAA	-	55	42.4	21.3	51.2	24.8	ns
$\overline{\text{CAS}}$ access time	tCAC	-	25	23.0	-12.3	30.6	-9.2	ns
Write data setup time 3	twDS3	40	-	33.8	-4.8	33.4	-6.6	ns
$\overline{\text{CAS}}$ setup time	tCSR	15	-	24.2	11.6	24.2	11.0	ns
Read strobe delay time	tRSD	-	30	9.0	35.8	9.6	34.7	ns

*1 Typical outputs measured with 50pF load

3-16 In-Circuit Emulation

Table 3-4. Control signal timing (Vcc = 5.0V, f = 16MHz)

Characteristics	Symbol	H8/3003		Probe Type				Unit
		Vcc = 5V f = 16MHz		HP 64784E + HP 64784x		HP 64784D		
		min	max	typ *1	worst	typ *1	worst	
$\overline{\text{RES}}$ setup time	tRESS	200	-	-	281.9	-	275.7	ns
$\overline{\text{RES}}$ pulse width	tRESW	10	-	-	-	-	-	tcyc
$\overline{\text{RESO}}$ output delay time	tRESO	-	100	-	109.6	-	108.4	ns
$\overline{\text{RESO}}$ output pulse width	tRESOW	132	-	-	-	-	-	tcyc
NMI setup time	tNMIS	150	-	-	231.9	-	225.7	ns
NMI hold time	tNMIH	10	-	-	-9.0	-	-4.1	ns
Interrupt pulse width	tNMIW	200	-	-	209.2	-	208.3	ns
Crystal oscillator setting time(reset)	tOSC1	20	-	-	-	-	-	ms
Crystal oscillator setting time (software standby)	tOSC2	8	-	-	-	-	-	ms

*1 Typical outputs measured with 50pF load

**Table 3-5. Timing condition of On-chip supporting modules
(Vcc = 5.0V, f = 16MHz)**

Characteristics	Symbol	H8/3003		Probe Type				Unit	
		Vcc = 5V f = 16MHz		HP 64784E + HP 64784x		HP 64784D			
		min	max	typ *1	worst	typ *1	worst		
DMA	$\overline{\text{DREQ}}$ setup time	tDRQS	30	-	-	65.9	-	59.7	ns
	$\overline{\text{DREQ}}$ hold time	tDRQH	10	-	-	-6.0	-	-1.1	ns
	$\overline{\text{TEND}}$ delay time 1	tTED1	-	50	-	61.6	-	59.0	ns
	$\overline{\text{TEND}}$ delay time 2	tTED2	-	50	-	61.6	-	59.0	ns
ITU	Timer outputdelay time	tTOCD	-	100	-	111.6	-	109.0	ns
	Timer input setup time	tTICS	50	-	-	85.9	-	79.7	ns
	Timer clock input setup time	tTCKS	50	-	-	85.9	-	79.7	ns
	Timer clock pulse width (single edge)	tTCKWH	1.5	-	-	-	-	-	tcyc
	Timer clock pulse width (both edge)	tTCKWL	2.5	-	-	-	-	-	tcyc

Table 3-5. Timing condition of On-chip supporting modules (Cont'd)
(Vcc = 5.0V, f = 16MHz)

Characteristics	Symbol	H8/3003		Probe Type				Unit	
		Vcc = 5V f = 16MHz		HP 64784E + HP 64784x		HP 64784D			
		min	max	typ *1	worst	typ *1	worst		
SCI	Input clock cycle(Async)	tSCYC	4	-	-	-	-	-	t cyc
	Input clock cycle(Sync)	tSCYC	6	-	-	-	-	-	t cyc
	Input clock rise time	tSCKr	-	1.5	-	-	-	-	t scyc
	Input clock fall time	tSCKf	-	1.5	-	-	-	-	t scyc
	Input clock pulse width	tSCKw	0.4	0.6	-	-	-	-	t scyc
	Transmit data delay time	tTXD	-	100	-	105.8	-	104.7	ns
	Received data setup time	tRXS	100	-	-	136.8	-	128.4	ns
	Received data hold time (Clock input)	tRXH	100	-	-	109.2	-	108.3	ns
PORT TPC	Output data delay time	tPWD	-	100	-	111.6	-	109.0	ns
	Input data setup time	tPRS	50	-	-	85.9	-	79.7	ns
	Input data hold time	tPRH	50	-	-	37.0	-	40.8	ns

*1 Typical outputs measured with 50pF Load

Table 3-6. Bus timing (Vcc = 3.0V, f = 10MHz)

Characteristics	Symbol	H8/3003		Probe Type		Unit
		Vcc = 3V f = 10MHz		HP 64797B + HP 64784x		
		min	max	typ *1	worst	
Clock cycle time	t _{cy}	100	500	-	-	ns
Clock pulse low width	t _{CL}	30	-	42.8	31.5	ns
Clock pulse high width	t _{CH}	30	-	46.0	31.5	ns
Clock rise time	t _{CR}	-	15	4.8	17.3	ns
Clock fall time	t _{CF}	-	15	6.4	17.3	ns
Address delay time	t _{AD}	-	50	23.0	33.9	ns
Address hold time	t _{AH}	20	-	42.2	12.6	ns
Address strobe delay time	t _{ASD}	-	40	7.8	36.6	ns
Write strobe delay time	t _{WSD}	-	50	10.4	36.6	ns
Strobe delay time	t _{SD}	-	50	3.4	36.6	ns
Write data strobe pulse width 1	t _{WSW1}	60	-	88.7	64.7	ns
Write data strobe pulse width 2	t _{WSW2}	110	-	137.5	113.4	ns

3-20 In-Circuit Emulation

Table 3-6. Bus timing (Vcc = 3.0V, f = 10MHz) (Cont'd)

Characteristics	Symbol	H8/3003		Probe Type		Unit
		Vcc = 3V f = 10MHz		HP 64797B + HP 64784x		
		min	max	typ *1	worst	
Address setup time 1	tAS1	15	-	29.4	-6.3	ns
Address setup time 2	tAS2	65	-	79.7	42.4	ns
Read data setup time	tRDS	35	-	38.4	52.6	ns
Read data hold time	tRDH	0	-	-35.0	28.7	ns
Write data delay time	tWDD	-	75	27.8	69.4	ns
Write data setup time 1	tWDS1	65	-	68.1	31.4	ns
Write data setup time 2	tWDS2	10	-	23.1	-18.6	ns
Write data hold time	tWDH	20	-	66.3	22.0	ns
Read data access time 1	tACC1	-	100	107.8	74.4	ns
Read data access time 2	tACC2	-	200	206.9	174.4	ns
Read data access time 3	tACC3	-	50	72.2	21.7	ns
Read data access time 4	tACC4	-	150	171.1	121.7	ns
Pre-charge time	tPCH	60	-	100.3	64.7	ns

Table 3-6. Bus timing (Vcc = 3.0V, f = 10MHz) (Cont'd)

Characteristics	Symbol	H8/3003		Probe Type		Unit
		Vcc = 3V f = 10MHz		HP 64797B + HP 64784x		
		min	max	typ *1	worst	
WAIT setup time	twTS	40	-	30.2	63.1	ns
WAIT set hold time	twTH	10	-	-27.6	-13.0	ns
BREQ setup time	tBRQS	40	-	-	78.1	ns
BACK delay time 1	tBACD1	-	50	10.8	36.6	ns
BACK delay time 2	tBACD2	-	50	-4.0	36.6	ns
Bus floating time	tBZD	-	70	17.8	46.6	ns

*1 Typical outputs measured with 50pF load

3-22 In-Circuit Emulation

Table 3-7. Control signal timing (Vcc = 3.0V, f = 10MHz)

Characteristics	Symbol	H8/3003		Probe Type		Unit
		Vcc = 3V f = 10MHz		HP 64797B + HP 64784x		
		min	max	typ *1	worst	
$\overline{\text{RES}}$ setup time	tRESS	200	-	-	284.1	ns
$\overline{\text{RES}}$ pulse width	tRESW	10	-	-	-	tcyc
$\overline{\text{RESO}}$ output delay time	tRESO	-	100	-	110.3	ns
$\overline{\text{RESO}}$ output pulse width	tRESOW	132	-	-	-	tcyc
NMI setup time	tNMIS	150	-	-	234.1	ns
NMI hold time	tNMIH	10	-	-	-11.0	ns
Interrupt pulse width	tNMIW	200	-	-	209.2	ns
Crystal oscillator setting time(reset)	tOSC1	20	-	-	-	ns
Crystal oscillator setting time (software standby)	tOSC2	8	-	-	-	ns

*1 Typical outputs measured with 50pF load

**Table 3-8. Timing condition of On-chip supporting modules
(V_{cc} = 3.0V, f = 10MHz)**

Characteristics	Symbol	H8/3003		Probe Type		Unit	
		V _{cc} = 3V f = 10MHz		HP 64797B + HP 64784x			
		min	max	typ *1	worst		
ITU	Timer output delay time	t _{TOCD}	-	100	-	111.6	ns
	Timer input setup time	t _{TICS}	50	-	-	88.1	ns
	Timer clock input setup time	t _{TCKS}	50	-	-	88.1	ns
	Timer clock pulse width (single edge)	t _{TCKWH}	1.5	-	-	-	t _{cyc}
	Timer clock pulse width (both edge)	t _{TCKWL}	2.5	-	-	-	t _{cyc}
SCI	Input clock cycle(Async)	t _{SCYC}	4	-	-	-	t _{cyc}
	Input clock cycle(Sync)	t _{SCYC}	6	-	-	-	t _{cyc}
	Input clock rise time	t _{SCKr}	-	1.5	-	-	t _{csyc}
	Input clock fall time	t _{SCKf}	-	1.5	-	-	t _{csyc}
	Input clock pulse width	t _{SCKw}	0.4	0.6	-	-	t _{csyc}
	Transmit data delay time	t _{TXD}	-	100	-	106.6	ns
	Received data setup time	t _{RXS}	100	-	-	138.8	ns
	Received data hold time (Clock input)	t _{RXH}	100	-	-	109.2	ns

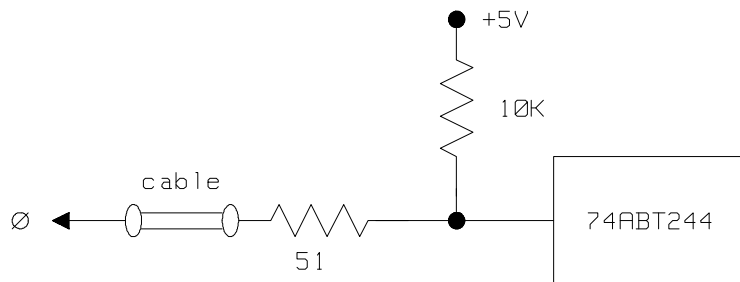
Table 3-8. Timing condition of On-chip supporting modules (Cont'd)
(V_{cc} = 3.0V, f = 10MHz)

Characteristics	Symbol	H8/3003		Probe Type		Unit	
		V _{cc} = 3V f = 10MHz		HP 64797B + HP 64784x			
		min	max	typ *1	worst		
PORT TPC	Output data delay time	t _{PWD}	-	100	-	111.6	ns
	Input data setup time	t _{PRS}	50	-	-	88.1	ns
	Input data hold time	t _{PRH}	50	-	-	35.6	ns

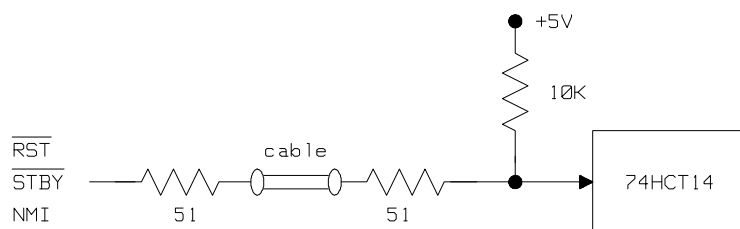
*1 Typical outputs measured with 50pF load

Target System Interface

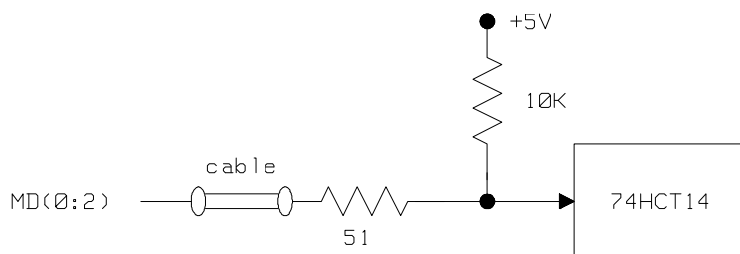
Ø



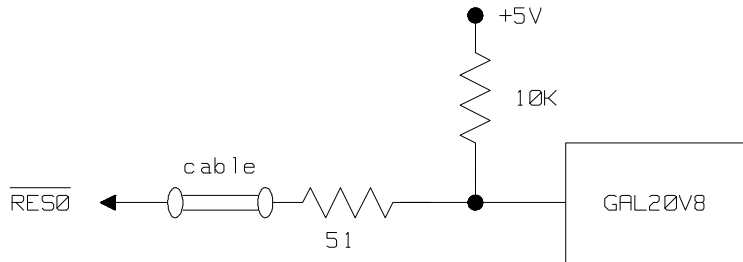
/RES, /STBY, NMI



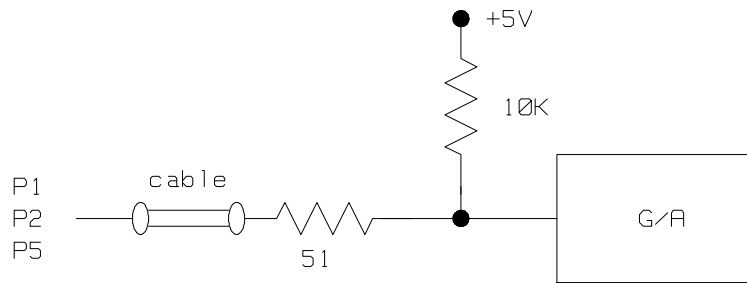
MD0-2



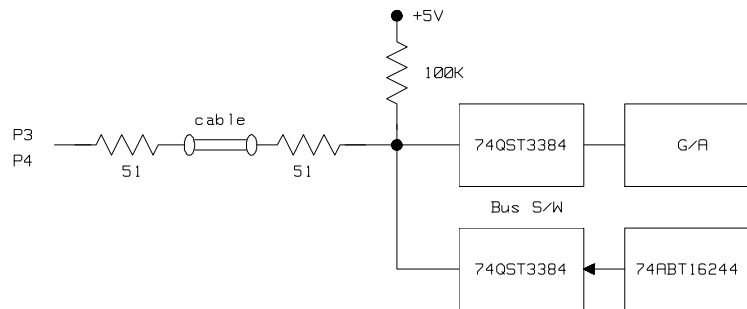
/RES0



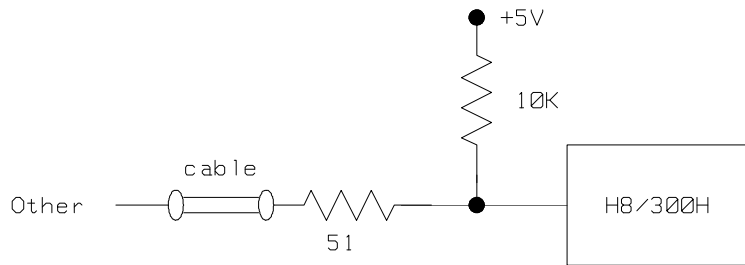
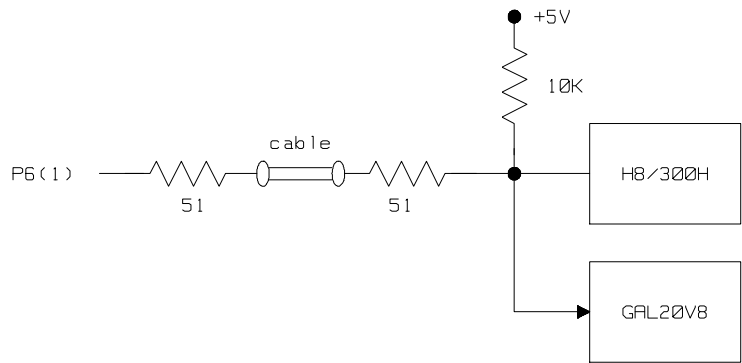
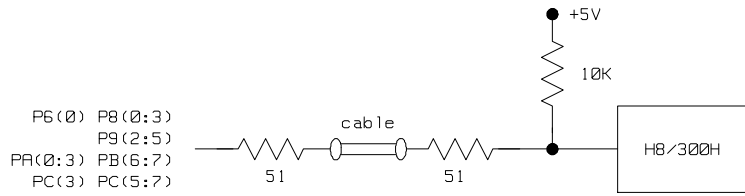
P1, P2, P5 (A0-23)



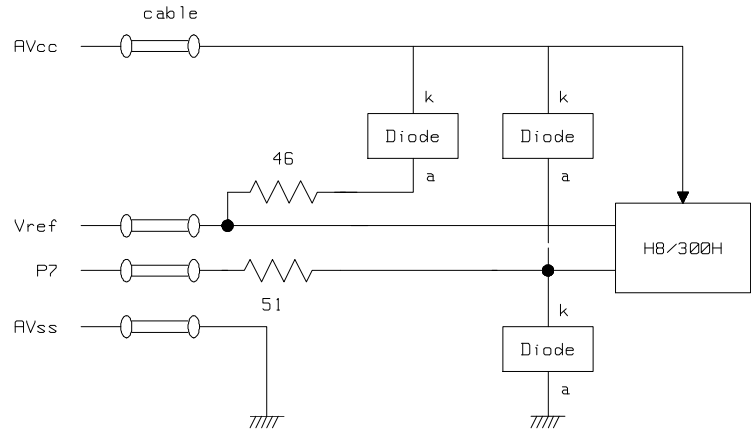
P3, P4 (D0-15)



P6, P8, P9, PA, PB, PC



P7, AVcc, Vref, AVss



Configuring the H8/3003 Emulator

In this chapter, we will discuss:

- how to configure the HP 64700 emulator for H8/3003 microprocessor to fit your particular measurement needs.
- some restrictions of HP 64700 emulator for H8/3003 microprocessor.

Types of Emulator Configuration

Emulation Processor to Emulator/Target System

These are the commands which are generally thought of as "configuration" items in the context of other HP 64700 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 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 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 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*.

Emulation Processor to Emulator/Target System

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.

cf The **cf** command defines how the emulation processor will respond to certain target system signals.

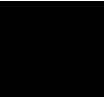
To see the default configuration settings defined by the **cf** command, type:

```
M> cf
```

You will see:

```
cf ba=en
cf chip=3042
cf clk=int
cf dbc=en
cf drst=dis
cf mode=7
cf nmi=en
cf rrt=dis
cf rsp=9
cf tdma=en
cf trfsh=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.

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.

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.



Note



When bus arbitration is enabled, the emulator responds to **/BREQ** signal during both foreground and background operation.

Note



The commands which cause the emulator to break to monitor are ignored during the processor releases bus cycles.

Note



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

cf chip The chip configuration item defines the microprocessor you emulate.

M> **cf chip=<chip_name>**

Valid <chip_name> are the following:

<chip_name>	Description
3001	Emulate H8/3001 microprocessor.
3002	Emulate H8/3002 microprocessor.
3003t	Emulate H8/3003 microprocessor.
3003	Emulate H8/3003 microprocessor with system clock divider.
3004	Emulate H8/3004 microprocessor.
3005	Emulate H8/3005 microprocessor.
3030	Emulate H8/3030 microprocessor.
3031	Emulate H8/3031 microprocessor.
3032	Emulate H8/3032 microprocessor.
3040	Emulate H8/3040 microprocessor.
3041	Emulate H8/3041 microprocessor.
3042	Emulate H8/3042 microprocessor.

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 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. You must use a clock input conforming to the specifications of Table 4-1.

Table 4-1. Clock Speeds

Clock source	Chip	Without 64797B	With 64797B
Internal	H8/3001 H8/3002 H8/3003T H8/3004 H8/3005 H8/3030 H8/3031 H8/3032 H8/3040 H8/3041 H8/3042	16MHz (System clock)	8MHz (System clock)
	H8/3003 with system clock divider	8MHz (System clock)	8MHz (System clock)

Table 4-1. Clock Speeds(Cont'd)

Clock source	Chip	Without 64797B	With 64797B
External	H8/3001 H8/3002 H8/3003T H8/3004 H8/3005 H8/3030 H8/3031 H83032 H8/3040 H8/3041 H8/3042	From 0.5 up to 16MHz (System clock)	From 0.5 up to 10MHz (System clock)
	H8/3003 with system clock divider	From 1 up to 24MHz (System clock is from 0.5 up to 12MHz)	From 1 up to 20MHz (System clock is from 0.5 up to 10MHz)

Note



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

cf dbc

The **dbc** (drive background cycles) option allows you to select whether or not the emulator will drive the target system bus on background 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 /LWR and /HWR) are driven during background cycles.

Background write cycles won't appear to the target system. (/LWR and /HWR signals are 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.

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



Refresh cycles, internal DMA cycles and target memory access are always driven to the target system regardless of this configuration.

Note



When **dbc** is disabled, the emulator can't respond to /WAIT signal.

Note



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

cf drst

The **drst** (drive reset) configuration item allows you to specify whether or not the emulator drives the /RESO signal to the target system by the Watchdog Timer.

```
M> cf drst=dis
```

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

```
M> cf drst=en
```

The emulator will drive the reset signal to the target system by the Watchdog Timer.

Note



The RSTOE (Reset output enable bit) is used to determine whether the H8/3003 processor outputs reset signal when the processor is reset by the watchdog timer. However, the H8/3003 emulator ignores the configuration of the RSTOE, and works as it is configured with **cf drst** command.

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

```
M> cf mode=ext
```

The emulator will work using the mode setting by the target system. The target system must supply appropriate inputs to MD0, MD1 and MD2.

```
M> cf mode=<mode_num>
```

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

Valid <mode_num> are following:

<mode_num>	Description
1	The emulator will operate in mode 1. (expanded 1M bytes mode without internal ROM: 8 bit data bus)
2	The emulator will operate in mode 2. (expanded 1M bytes mode without internal ROM:16 bit data bus)
3	The emulator will operate in mode 3. (expanded 16M bytes mode without internal ROM: 8 bit data bus)
4	The emulator will operate in mode 4. (expanded 16M bytes mode without internal ROM:16 bit data bus)
5	The emulator will operate in mode 5. (expanded 1M bytes mode with internal ROM: 8 bit data bus)

6 The emulator will operate in mode 6. (single chip normal mode)

7 The emulator will operate in mode 7. (single chip advanced mode)

Note



It is recommended to specify operation mode number in this configuration, since the emulator does not work fine when MD0,MD1 and MD2 are not steady.

Note



When you emulate H8/3004/05, **cf mode = ext** is not available. You have to configure processor mode using **cf mode=<mode_num>**

Note



When mismatch takes place between **cf chip** and **cf mode**, the emulator will operate in mode which you don't specify.

Note



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

cf nmi 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.

The emulator does not accept any interrupt while in background monitor. Such interrupts are suspended while running the background monitor, and will occur when context is changed to foreground.

Note



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

cf rrt 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.

Caution



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 will be set upon the transition from emulation reset into the emulation monitor.

```
R> cf rsp=XXXXXXXX
```

where **XXXXXXXX** is a 32-bit even address, will set the stack pointer 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

For example, to set the stack pointer to 0ff00 hex, type:

```
R> cf rsp=0ff00
```

Now, if you break the emulator to monitor using the **b** command, the stack pointer will be modified to the value 0ff00 hex.

Note



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 tdma

The **tdma** (trace internal DMA cycles) configuration item defines whether or not the emulator traces internal DMA cycles.

```
M> cf tdma=en
```

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

```
M> cf tdma=dis
```

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

Note



Internal DMA cycles may be traced regardless of this configuration in order to disassemble the trace list correctly.

cf trfsh

The **trfsh** (trace refresh cycles) configuration item defines whether or not the emulator traces refresh cycles.

```
M> cf trfsh=en
```

When you enable this item with the above command, refresh cycles are traced by the emulation analyzer.

```
M> cf trfsh=dis
```

When disabled, refresh cycles are not traced by the analyzer.

Note



Refresh cycles may be traced regardless of this configuration in order to disassemble the trace list correctly.

Note



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

cf trst The **trst** (target reset) configuration item allows you to specify whether or not the emulator responds to /RES and /STBY signals from the target system during foreground operation. When running the background monitor, the emulator ignores such signals.

M> **cf trst=en**

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

M> **cf trst=dis**

When disabled, the emulator won't respond to /RES and /STBY inputs from the target system.



Note



/RES and /STBY signals are always ignored during background operation regardless of this configuration.

Note



The H8/3003 does not support hardware standby mode, and /STBY input will be given the emulator /RES input.

Note



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

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 H8/3003 emulator memory mapper allows you to define up to 16 different map terms; each map term has a minimum size of 512 bytes. If you specify a value less than 512 bytes, 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 1000 through 3fff 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:

```
R> map 0f000..0f100 tram
R> map 1000..3fff eram
R> map
# remaining number of terms : 14
# remaining emulation memory : 7e800h bytes
map 001000..003fff eram # term 1
map 00f000..00f1ff tram # term 2
map other tram
```

As you can see, the mapper rounded up the second term to 512 bytes block, since those are minimum size blocks supported by the H8/3003 emulator.

Note



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

Note



You don't have to map internal RAM as emulation RAM, since the H8/3003 emulator automatically maps internal RAM as emulation RAM and this area is behaved like internal RAM. However emulation memory system does not introduce internal RAM area in memory mapping display.

Note



If you map internal RAM area as emulation memory, this area is behaved like external memory overlapped with internal RAM. However the H8/3003 emulator is always accessed internal RAM area mapped by the emulator. And if you map internal RAM as guarded memory, the emulator prohibits to access to this area by m commands.

Note



When you emulate H8/3005 processor, you can't use address 0fef10h - 0ff00fh (mode 1) and 0ffef10h - 0fff00fh (mode 3) as internal RAM. These area are worked as external 8bit 3state area, and you have to map these area as emulation RAM.

Note



You should map all memory ranges except internal RAM 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.

Note



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

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.

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.

Note



If emulator writes to the memory mapped as ROM or guarded area in internal DMA cycles, the emulator will not break to the monitor regardless of this configuration.

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 **trig1** signal from the analyzer, type:

```
M> bc -e trig1
```

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

Where to Find More Information

Due to the architecture of the HP 64700 emulators, there are a wide variety of items that affect how the emulator interacts with your system, controller, and other measuring instruments. If you need more configuration information, we suggest the following strategy:

If you need tutorial information --

- Emulator: look at this manual.
- Analyzer: look at the *Analyzer User's Guide* and this manual.
- CMB: look at the *CMB User's Guide*.

If you need reference information --

- Look at the *Terminal Interface User's Reference* manual (also contains some examples).

Notes

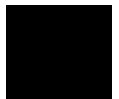


H8/3003 Emulator Specific Command Syntax

The following pages contain descriptions of command syntax specific to the H8/3003 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.
- **<ACCESS_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.
- **<REG_NAME>**. May be specified in the **reg** (register) command.

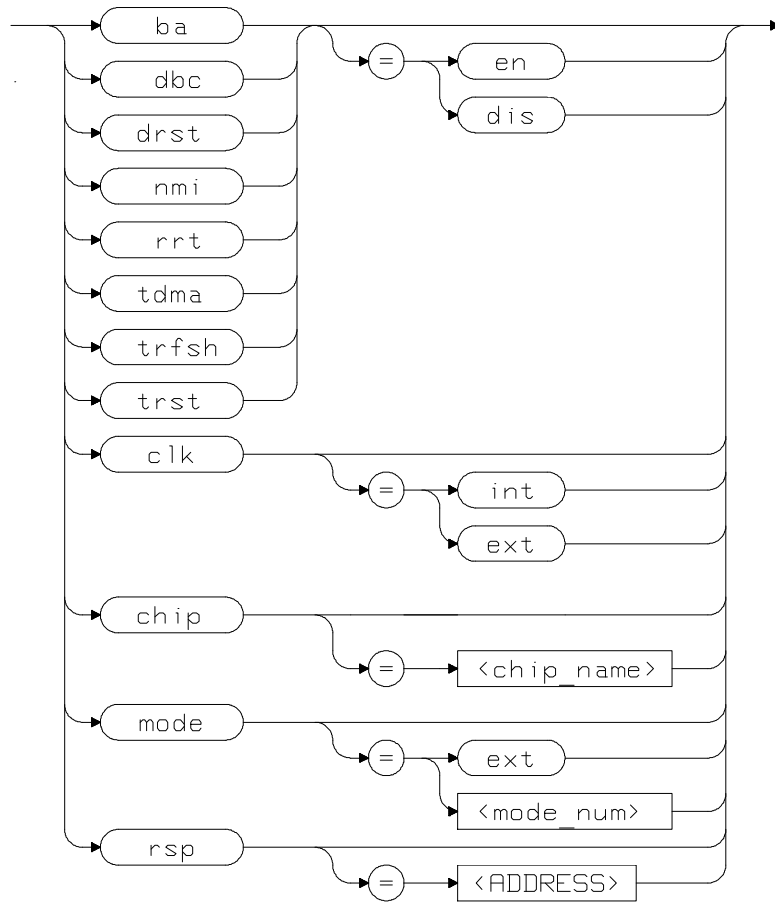
Command and error messages which are specific to the H8/3003 emulator are also described in this chapter.



CONFIG_ITEMS

Summary H8/3003 emulator configuration items.

Syntax

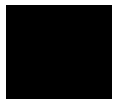


Description

The H8/3003 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 reset to target system.
mode	Determine emulator processor operation mode.
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.
tdma	Enable/disable tracing internal DMA cycles.
trfsh	Enable/disable tracing refresh 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> 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> cf nmi=dis rrt=dis clk
```

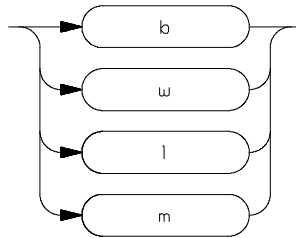
```
cf clk=ext
```

Related information Refer to the cf syntax pages in the *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 accessed in a byte format, and when memory locations are modified, bytes are changed.

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

Note



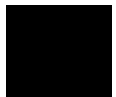
When the **<ACCESS_MODE>** is **w**, modifying target memory will fail if you try to modify memory from an odd address or with data which byte count is odd. Also, you can't load file which byte count is odd. Therefore, it is recommended to use the emulator with the default **<ACCESS_MODE>** (**b**).

Defaults

The **<ACCESS_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 24 bits address information.

Examples `m 1000`
`m 200000..2000ff`

REGISTER CLASS and NAME

Summary H8/3003 register designators. All available register class names and register names are listed below.

<REG_CLASS>

<REG_NAME> Description

* (All basic registers)

pc	Program counter
ccr	Condition code register
er0	Register ER0
er1	Register ER1
er2	Register ER2
er3	Register ER3
er4	Register ER4
er5	Register ER5
er6	Register ER6
er7	Register ER7
sp	Stack pointer
m dcr	Mode control register(Read Only)

sys (System control)

mdcr	Mode control register(Read Only)
syscr	System control register

intc (Interrupt controller)

iscr	IRQ sense control register
ier	IRQ enable register
isr	IRQ status register
ipra	Interrupt priority register A
iprb	Interrupt priority register B

busc (Bus controller)

abwcr	Byte/Word area control register *
aster	2/3 state area control register
wcr	Wait control register
wcer	Wait controller enable register
brcr	Bus release control register *

* Except 3030, 3031,3032

rfshcr (Refresh controller)

The following registers does not exist in 3030, 3031, and 3032

rfshcr	Refresh control register
rtmcsr	Refresh timer control/status register
rtcnt	Refresh timer counter
rtcor	Refresh time constant register

dmac0 (DMA controller 0)

The following registers does not exist in 3030, 3031, and 3032

mar0a	Memory address register 0A
etcr0a	Transfer count register 0A
ioar0a	I/O address register 0A
dtr0a	Data transfer control register 0A
mar0b	Memory address register 0B
etcr0b	Transfer count register 0B
ioar0b	I/O address register 0B
dtr 0b	Data transfer control register 0B

dmac1 (DMA controller 1)

The following registers does not exist in 3030, 3031, and 3032

mar1a	Memory address register 1A
etcr1a	Transfer count register 1A
ioar1a	I/O address register 1A
dtr1a	Data transfer control register 1A
mar1b	Memory address register 1B
etcr1b	Transfer count register 1B
ioar1b	I/O address register 1B
dtr 1b	Data transfer control register 1B

dmac2 (DMA controller 2)

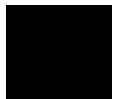
The following registers does not exist in 3002, 3030, 3031, 3032 and 3042

mar2a	Memory address register 2A
etcr2a	Transfer count register 2A
ioar2a	I/O address register 2A
dtr2a	Data transfer control register 2A
mar2b	Memory address register 2B
etcr2b	Transfer count register 2B
ioar2b	I/O address register 2B
dtr 2b	Data transfer control register 2B

dmac3 (DMA controller 3)

The following registers does not exist in 3002, 3030, 3031, 3032 and 3042

mar3a	Memory address register 3A
etcr3a	Transfer count register 3A
ioar3a	I/O address register 3A
dtr3a	Data transfer control register 3A
mar3b	Memory address register 3B
etcr3b	Transfer count register 3B
ioar3b	I/O address register 3B
dtr 3b	Data transfer control register 3B



port (I/O port)

p1ddr	Port 1 data direction register(Write Only)*1,2
p2ddr	Port 2 data direction register(Write Only)*1,2
p3ddr	Port 3 data direction register(Write Only)*1,2
p4ddr	Port 4 data direction register(Write Only)*5
p5ddr	Port 5 data direction register(Write Only)*1
p6ddr	Port 6 data direction register(Write Only)*1
p8ddr	Port 8 data direction register(Write Only)
p9ddr	Port 9 data direction register(Write Only)
paddr	Port A data direction register(Write Only)
pbddr	Port B data direction register(Write Only)
pcddr	Port C data direction register(Write Only)*1,3,5
p1dr	Port 1 data register*1,2
p2dr	Port 2 data register*1,2
p3dr	Port 3 data register*1,2
p4dr	Port 4 data register*5
p5dr	Port 5 data register*1
p6dr	Port 6 data register
p7dr	Port 7 data register(Write Only)
p8dr	Port 8 data register
p9dr	Port 9 data register
padr	Port A data register
pbdr	Port B data register
pcdr	Port C data register*1,3,5
p2pcr	Port 2 input pull up MOS control register *1,2,4
p4pcr	Port 4 input pull up MOS control register *4,5
p5pcr	Port 5 input pull up MOS control register *1,4

*1 Except 3002

*2 Except 3003

*3 Except 3042

*4 NOT effective

*5 Except 3030, 3031, 3032

itug (16 bit integrated timer pulse unit general)

tstr	Timer start register
tsnc	Timer synchro register
tmdr	Timer mode register
tfcn	Timer function control register
toer	Timer output master control register
toer	Timer output control register

itu0 (16 bit integrated timer pulse unit 0)

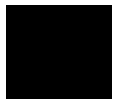
tcr0	Timer control register 0
tior0	Timer I/O control register 0
tier0	Timer interrupt enable register 0
tsr0	Timer status register 0
tcnt0	Timer counter 0
gra0	General register A0
grb0	General register B0

itu1 (16 bit integrated timer pulse unit 1)

tcr1	Timer control register 1
tior1	Timer I/O control register 1
tier1	Timer interrupt enable register 1
tsr1	Timer status register 1
tcnt1	Timer counter 1
gra1	General register A1
grb1	General register B1

itu2 (16 bit integrated timer pulse unit 2)

tcr2	Timer control register 2
tior2	Timer I/O control register 2
tier2	Timer interrupt enable register 2
tsr2	Timer status register 2
tcnt2	Timer counter 2
gra2	General register A2
grb2	General register B2



itu3 (16 bit integrated timer pulse unit 3)

tcr3	Timer control register 3
tior3	Timer I/O control register 3
tier3	Timer interrupt enable register 3
tsr3	Timer status register 3
tcnt3	Timer counter 3
gra3	General register A3
grb3	General register B3
bra3	Buffer register A3
brb3	Buffer register B3

itu4 (16 bit integrated timer pulse unit 4)

tcr4	Timer control register 4
tior4	Timer I/O control register 4
tier4	Timer interrupt enable register 4
tsr4	Timer status register 4
tcnt4	Timer counter 4
gra4	General register A4
grb4	General register B4
bra4	Buffer register A4
brb4	Buffer register B4

tpc (Programable timing pattern controller)

tpmr	TPC output mode register
tpcr	TPC output control register
ndera	Next data enable register A
ndra	Next data register A (address: 0xfffa5h)
ndra0	Next data register A (address: 0xfffa7h)
nderb	Next date enable register B
ndrb	Next data register B (address: 0xfffa4h)
ndrb2	Next data register B (address: 0xfffa6h)

wdt (Watch dog timer)

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

sci0 (Serial communication interface 0)

smr0	Serial mode register 0
brr0	Bit rate register 0
scr0	Serial control register 0
tdr0	Transmit data register 0
ssr0	Serial status register 0
rdr0	Receive data register 0 (Read Only)

sci1 (Serial communication interface 1)

The following registers does not exist in 3030, 3031, and 3032

smr1	Serial mode register 1
brr1	Bit rate register 1
scr1	Serial control register 1
tdr1	Transmit data register 1
ssr1	Serial status register 1
rdr1	Receive data register 1 (Read Only)

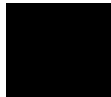
adc (A/D converter)

addra	A/D data register A (Read Only)
addrb	A/D data register B (Read Only)
addrc	A/D data register C (Read Only)
addrd	A/D data register D (Read Only)
adcsr	A/D control/status register
adcr	A/D control register

dac (D/A converter)

The following registers does not exist in 3002, 3003, 3030, 3031, and 3032

dadr0	D/A data register 0
dadr1	D/A data register 1
dacr	D/A control register



NOCLASS

The following register names are not included in any register class.

r0	Register R0
r1	Register R1
r2	Register R2
r3	Register R3
r4	Register R4
r5	Register R5
r6	Register R6
r7	Register R7
e0	Register E0
e1	Register E1
e2	Register E2
e3	Register E3
e4	Register E4
e5	Register E5
e6	Register E6
e7	Register E7
r0h	Register R0H
r0l	Register R0L
r1h	Register R1H
r1l	Register R1L
r2h	Register R2H
r2l	Register R2L
r3h	Register R3H
r3l	Register R3L
r4 h	Register R4H
r4l	Register R4L
r5h	Register R5H
r5l	Register R5L
r6h	Register R6H
r6l	Register R6L
r7h	Register R7H
r7l	Register R7L

Emulator Specific Error Messages

The following is the error messages which are specific to the H8/3003 emulator. The cause of the errors is described, as well as the action you must take to remedy the situation.

Message 140 : Stack is in I/O registers

Cause

This error occurs when you attempt to execute user program (with r or s command) with the stack pointer set at internal I/O register area.

Action

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

Message 141 : Invalid address for run or step in current mode

Cause

This error occurs when you attempt to execute user program (with r or s command) from address over area of current mode.

Message 170 : Emulation memory card not found in card cage

Cause

This error occurs when you don't insert memory board in card cage, or connect memory board which is not supported.

Action

Insert correct memory board.

Notes

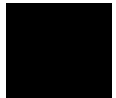


Index

- A** ADDRESS syntax, **A-6**
 - Analyzer
 - configuration, **2-23**
 - configuration commands, **4-2**
 - halting, **2-25**
 - matters to be attended to, **2-30**
 - pipeline, **2-25**
 - storage specification, **2-23 - 2-24**
 - trace, **2-22**
 - trace list display, **2-24**
 - trace list format, **2-24**
 - trigger, **2-23**
 - trigger specification, **2-23**
 - triggering by data, **2-34**
 - analyzer status
 - predefined equates, **2-23**
 - Analyzer trace
 - starting, **2-24**

- B** b Command, **2-19**
 - bc Command, **2-10, 2-27, 4-18**
 - Before using the emulator, **2-2**
 - bp Command, **2-27, 4-18**
 - Break
 - write to ROM, **4-18**
 - Break condition, **2-27**
 - Breaks, **4-18**
 - Bus arbitration
 - configure emulator's response, **4-4**
 - using configuration to isolate target problem, **4-4**

- C** cf ba Command, **4-4**
 - cf chip Command, **4-5**
 - cf clk Command, **4-6**
 - cf Command, **2-9, 4-3**
 - cf dbc Command, **4-7**



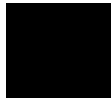
cf drst Command, **4-8**
cf mode Command, **4-9**
cf nmi Command, **4-11**
cf rrt Command, **4-11**
cf rsp Command, **4-12**
cf tdma Command, **4-13**
cf trfsh Command, **4-13**
cf trst Command, **4-14**
cim Command, **2-27**
Clock selection for microprocessor, **4-6**
Command help, **2-7**
Command prompts, **2-19**
Command syntax, specific to H8/3003 emulator, **A-1**
Commands
 analyzer configuration, **4-2**
 b, **2-19**
 bc, **2-10, 2-27, 4-18**
 bp, **2-27, 4-18**
 cf, **2-9, 4-3**
 cf ba, **4-4**
 cf chip, **4-5**
 cf clk, **4-6**
 cf dbc, **4-7**
 cf drst, **4-8**
 cf mode, **4-9**
 cf nmi, **4-11**
 cf rrt, **4-11**
 cf rsp, **4-12**
 cf tdma, **4-13**
 cf trfsh, **4-13**
 cf trst, **4-14**
 cim, **2-27**
 configuration, **4-1**
 coordinated measurement, **4-2**
 cov, **2-29**
 help, **2-7**
 init, **2-8**
 load, **2-16**
 m, **2-13, 2-21**
 map, **2-11, 4-15**
 measurement, **4-2**

- r, **2-19 - 2-20**
- reg, **2-20**
- rst, **2-19**
- s, **2-22**
- ser, **2-29**
- system, **4-2**
- t, **2-24**
- tf, **2-24**
- tg, **2-24**
- th, **2-25**
- tinit, **2-30**
- tl, **2-24**
- tp, **2-31**
- tsto, **2-24, 2-32**
- xp, **2-15**

CONFIG_ITEMS syntax, A-2

Configuration

- analyzer, **4-2**
- breaks, **4-18**
- bus arbitration, **4-4**
- clock selection, **4-6**
- displaying, **4-3**
- drive background cycles to target, **4-7**
- drive emulation reset to target, **4-8**
- enable/disable target interrupts, **4-11**
- enable/disable target system reset, **4-14**
- enable/disable to trace DMA cycles, **4-13**
- enable/disable to trace refresh cycles, **4-13**
- for getting started, **2-9, 2-11**
- hardware standby, **4-14**
- internal RAM, **4-17**
- measurement commands, **4-2**
- memory mapping, **4-15**
- microprocessor operation mode, **4-9**
- microprocessor selection, **4-5**
- processor to emulator/target system, **4-1, 4-3**
- restrict to real-time runs, **4-11**
- stack pointer, **4-12**
- system, **4-2**
- to access the internal ROM, **4-16**
- types of, **4-1**



Coordinated measurement commands, **4-2**
cov Command, **2-29**
Coverage measurement, **2-29**

- D** Displaying
 - configuration, **4-3**
 - memory, **2-21**
 - registers, **2-20**
 - trace list, **2-24**
- DMA cycles
 - enable/disable tracing DMA cycles, **4-13**
- E** electrical characteristics, **3-13**
 - Emulator
 - configuration, **2-9, 2-11**
 - initialization, **2-8**
 - purpose, **1-1**
 - Emulator features, **1-3**
 - analyzer, **1-7**
 - breakpoints, **1-7**
 - clock speeds, **1-5**
 - emulation memory, **1-7**
 - processor reset control, **1-8**
 - register display/modify, **1-7**
 - restrict to real-time runs, **1-8**
 - supported microprocessors, **1-3**
 - Emulator limitations, **1-9**
 - Emulator specific command syntax, **A-1**
 - equates predefined for analyzer status, **2-23**
 - Evaluation chip, **1-10**
- F** Function codes
 - memory mapping, **4-15**
- H** H8/3003 microprocessor
 - installation procedure, **3-9**
- Halting the analyzer, **2-25**
- Help, **2-7**
- help Command, **2-7**
- I** In-circuit emulation
 - H8/3001 with mode 3/4, **3-3**
 - H8/3004/5 with mode 3, **3-3**

- installing the PGA adaptor, **3-6, 3-11**
- installing the QFP adaptor, **3-5**
- PGA adaptor, **3-3**
- QFP adaptor, **3-3**
- QFP probe, **3-4**
- QFP socket/adaptor, **3-4**
- Information help, **2-7**
- init Command, **2-8**
- Initializing the Emulator, **2-8**
- installing H8/3003 microprocessor, **3-9**
- Installing target system probe
 - target system probe, **3-2**
- internal RAM
 - mapping, **4-17**
- Internal ROM access, **4-16**
- Interrupts
 - enable/disable from target system, **4-11**

L

- labels (trace), predefined, **2-22**
- limitations
 - DMA support, **1-9**
 - Hardware standby mode, **1-9, 4-14**
 - internal RAM of H8/3005, **1-9**
 - Interrupts in background, **1-9**
 - Reset output enable bit, **1-9**
 - Sleep/standby mode, **1-9**
 - store condition and trace, **2-32**
 - Watch dog timer in background, **1-9**
- load Command, **2-16**
- Loading programs, **2-12**
 - for Standalone Configuration, **2-12**
 - for Transparent Configuration, **2-15**
 - load command, **2-16**
 - transfer utility, **2-15**
- low voltage adaptor
 - installation, **3-10**
 - specification, **3-10**

M

- m Command, **2-13, 2-21**
- map Command, **2-11, 4-15**
- mapping of internal RAM, **4-17**
- Measurement commands, **4-2**

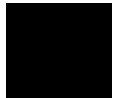


- Memory Display, **2-21**
 - mnemonic format, **2-18**
- Memory mapping, **4-15**
 - defining memory type to emulator, **4-15**
 - function codes, **4-15**
 - sequence of map/load commands, **4-17**
- Memory search, **2-29**
- Mnemonic display format, **2-18**
- N** notes
 - /STBY input will give the emulator /RES input, **4-14**
 - mapping of internal RAM, **4-17**
- P** PGA adaptor, **3-3**
 - installation procedure, **3-6, 3-11**
- predefined equates, **2-23**
- predefined trace labels, **2-22**
- Predefining stack pointer, **4-12**
- Prerequisites for using the emulator, **2-2**
- Processor clock selection, **4-6**
- Program loads, **2-12**
- Program tracing, **2-22**
- Prompts
 - emulator command, **2-19**
- Purpose of the Emulator, **1-1**
- Q** QFP adaptor, **3-3**
 - installation procedure, **3-5**
- QFP probe, **3-4**
- QFP socket/adaptor, **3-4**
- R** r Command, **2-19 - 2-20**
- Real-time runs
 - restricting emulator to, **4-11**
- Refresh cycles
 - enable/disable tracing refresh cycles, **4-13**
- reg Command, **2-20**
- REGISTER CLASS, **A-7**
- Register Display, **2-20**
- REGISTER NAME, **A-7**
- Restrict to real time runs, **4-11**
 - permissible commands, **4-11**
 - target system dependency, **4-12**

rst Command, **2-19**
RSTOE, **1-9**
run from reset, **3-12**

S s Command, **2-22**
Sample programs
 for getting started, **2-3**
ser Command, **2-29**
Single step, **2-22**
Software breakpoints, **2-27, 4-18**
 defining in target ROM, **2-27**
specification
 low voltage adaptor, **3-10, 3-12**
Stack pointer
 predefining, **4-12**
Starting a trace, **2-24**
stat (emulation analyzer status) trace label, **2-23**
Storage qualifier, **2-24**
Syntax (command), specific to H8/3003 emulator, **A-1**
System commands, **4-2**

T t Command, **2-24**
target system
 H8/3001 with mode 3/4, **3-3**
 H8/3004/5 with mode 3, **3-3**
 interface, **3-26**
 PGA adaptor, **3-3**
 QFP adaptor, **3-3 - 3-4**
Target system dependency on executing code, **4-12**
Target system interrupts
 enable/disable, **4-11**
Target system probe
 installation, **3-2**
target system reset, **4-14**
 run from reset, **3-12**
tf Command, **2-24**
tg Command, **2-24**
th Command, **2-25**
tinit Command, **2-30**
tl Command, **2-24**
tlb (display/modify trace labels) command, **2-22**
tp Command, **2-31**



trace labels, predefined, **2-22**
Trace list display, **2-24**
Trace list format, **2-24**
Tracing program execution, **2-22**
Transfer utility, **2-15**
Transparent mode, **2-15**
Trigger signals
 break upon, **4-19**
tsto Command, **2-24**
 effect on the analyzer, **2-32**
Types of configuration, **4-1**

X xp Command, **2-15**