HP 64784

H8/3003 Emulator Softkey Interface

User's Guide

HP Part No. 64784-97011
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.

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

Printing History

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

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

Edition 1	64784-97001, July 1993
Edition 2	64784-97003, March 1994
Edition 3	64784-97005, August 1994
Edition 4	64784-97007, April 1995
Edition 5	64784-97009, May 1995
Edition 6	**64784-97011, Aug 1995**

Using This Manual

This manual will show you how to use the HP 64784 H8/3003 Emulator with the Softkey Interface. This manual will also help define how these emulators differ from other HP 64700 Emulators.

This manual will:

- Show you how to use emulation commands by executing them on a sample program and describing their results.
- Show you how to configure the emulator for your development needs. Topics include: restricting the emulator to real-time execution, and selecting a target system clock source.
- Show you how to use the emulator in-circuit (connected to a target system).

This manual will not:

- Show you how to use every Softkey Interface command and option; the Softkey Interface is described in the *Softkey Interface Reference*.

Organization

Chapter 1 **Introduction.** This chapter lists the H8/3003 emulator features and describes how they can help you in developing new hardware and software.

Chapter 2 **Getting Started.** This chapter shows you how to use emulation commands by executing them on a sample program. This chapter describes the sample program and how to: load programs into the emulator, map memory, display and modify memory, display registers, step through programs, run programs, set software breakpoints, search memory for data, and use the analyzer.

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

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

Chapter 5 **Using the Emulator.** This chapter describes emulation topics which are not covered in the "Getting Started" chapter.

Conventions

Example commands throughout the manual use the following conventions:

bold	Commands, options, and parts of command syntax.
bold italic	Commands, options, and parts of command syntax which may be entered by pressing softkeys.
normal	User specified parts of a command.
$	Represents the HP-UX prompt. Commands which follow the "$" are entered at the HP-UX prompt.
<RETURN>	The carriage return key.

Notes

Contents

Illustrations

Tables

1

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.

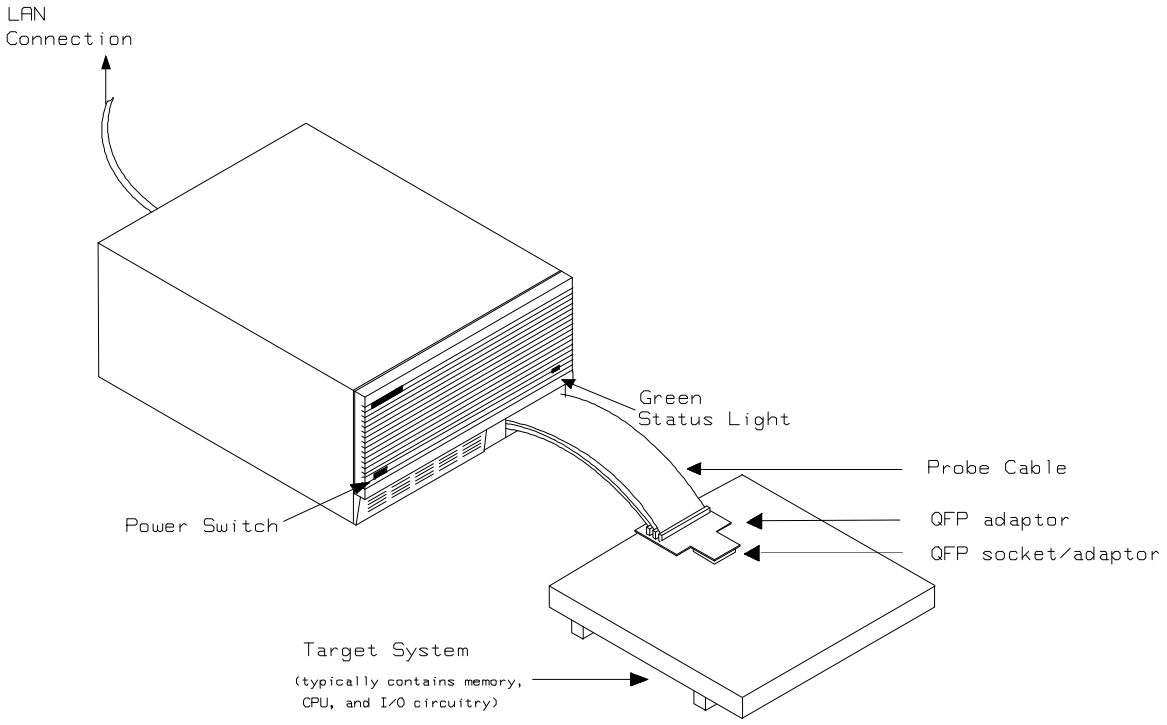


Figure 1-1. HP 64784 Emulator for the H8/3003

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.

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 problenm caused by the H8/3003 Evaluation chip in the emulator.

2

Getting Started

Introduction

This chapter will lead you through a basic, step by step tutorial designed to familiarize you with the use of the H8/3003 emulator with the Softkey Interface.

This chapter will:

- Tell you what must be done before you can use the emulator as shown in the tutorial examples.
- Describe the sample program used for this chapter's example.

This chapter will show you how to:

- Start up the Softkey Interface.
- Load programs into emulation and target system memory.
- Enter emulation commands to view execution of the sample program.

Before You Begin

Prerequisites

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

1. Connected the emulator to your computer. The *HP 64700 Series Installation/Service* manual show you how to do this.

2. Installed the Softkey Interface software on your computer. Refer to the *HP 64700 Series Installation/Service* manual for instructions on installing software.

3. In addition, you should read and understand the concepts of emulation presented in the *Concepts of Emulation and Analysis* manual. The *Installation/Service* manual also covers HP64700 system architecture. A brief understanding of these concepts may help avoid questions later.

   You should read the *Softkey Interface Reference* manual to learn how to use the Softkey Interface in general. For the most part, this manual contains information specific to the H8/3003 emulator.

A Look at the Sample Program

The sample program used in this chapter is listed in figure 2-1. The program emulates a primitive command interpreter. The sample program is shipped with the Softkey Interface and may be copied from the following location.

/usr/hp64000/demo/emul/hp64784/cmd_rds.src

Data Declarations

The "Table" section defines the messages used by the program to respond to various command inputs. These messages are labeled **Msg_A,Msg_B,** and **Msg_I**.

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

                .SECTION        Table,DATA
Msgs
Msg_A           .SDATA          "THIS IS MESSAGE A"
Msg_B           .SDATA          "THIS IS MESSAGE B"
Msg_I           .SDATA          "INVALID COMMAND"
End_Msgs

                .SECTION        Prog,CODE
;*************************************************
;* Set up the Stack Pointer.
;*************************************************
Init            MOV.L           #Stack,ER7
;*************************************************
;* Clear previous command.
;*************************************************
Clear           MOV.B           #H'00,R0L
                MOV.B           R0L,@Cmd_Input
;*************************************************
;* Read command input byte.  If no command has been
;* entered, continue to scan for it.
;*************************************************
Scan            MOV.B           @Cmd_Input,R2L
                CMP.B           #H'00,R2L
                BEQ             Scan
;*************************************************
;* A command has been entered.  Check if it is
;* command A, command B, or invalid command.
;*************************************************
Exe_Cmd         CMP.B           #H'41,R2L
                BEQ             Cmd_A
                CMP.B           #H'42,R2L
                BEQ             Cmd_B
                BRA             Cmd_I
;*************************************************
;* Command A is entered.  R3L = the number of bytes
;* in message A.  R4 = location of the message.
;* Jump to the routine which writes the message.
;*************************************************
Cmd_A           MOV.B           #Msg_B-Msg_A,R3L
                MOV.L           #Msg_A,ER4
                BRA             Write_Msg
;*************************************************
;* Command B is entered.
;*************************************************
Cmd_B           MOV.B           #Msg_I-Msg_B,R3L
                MOV.L           #Msg_B,ER4
                BRA             Write_Msg
;*************************************************
;* An invalid command is entered.
;*************************************************
Cmd_I           MOV.B           #End_Msgs-Msg_I,R3L
                MOV.L           #Msg_I,ER4
;*************************************************
```

Figure 2-1. Sample Program Listing

```
;* The destination area is cleared.
;*************************************************
;
Write_Msg       MOV.L           #Msg_Dest,ER5
Clear_Old       MOV.B           #H'20,R6L
Clear_Loop      MOV.B           R0L,@ER5
                ADDS.L          #1,ER5
                DEC.B           R6L
                BNE             Clear_Loop
;*************************************************
;* Message is written to the destination.
;*************************************************
;
                MOV.L           #Msg_Dest,ER5
Write_Loop      MOV.B           @ER4+,R6L
                MOV.B           R6L,@ER5
                ADDS.L          #1,ER5
                DEC.B           R3L
                BNE             Write_Loop
;*************************************************
;* Go back and scan for next command.
;*************************************************
;
                BRA             Clear

                .SECTION        Data,DATA
;*************************************************
;* Command input byte.
;*************************************************
;
Cmd_Input       .RES.B          1
                .RES.B          1
;*************************************************
;* Destination of the command messages.
;*************************************************
;
Msg_Dest        .RES.W          H'80
Stack
                .END            Init
```

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

Initialization

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

Reading Input

The instruction at the **Clear** label clears any random data or previous commands from the **Cmd_Input** byte. The **Scan** loop continually reads the **Cmd_Input** byte to see if a command is entered (a value other than 0 hex).

Processing Commands

When a command is entered, the instructions from **Exe_Cmd** to **Cmd_A** determine whether the command was "A", "B", or an invalid command.

If the command input byte is "A" (ASCII 41 hex), execution is transferred to the instructions at **Cmd_A**.

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

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

The instructions at **Cmd_A, Cmd_B**, and **Cmd_I** each load register R3L with the length of the message to be displayed and register ER4 with the starting location of the appropriate message. Then, execution transfers to **Write_Msg** which writes the appropriate message to the destination location, **Msg_Dest**.

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

The Destination Area

The "Data" section declares memory storage for the command input byte, the destination area, and the stack area.

Sample Program Assembly

The sample program is written for and assembled with the Hitachi Cross System. The sample program was assembled with the following command.

```
$ asm38 cmd_rds.src -debug -cpu=300ha
<RETURN>
```

Linking the Sample Program

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

```
$ lnk -subcommand=cmd_rds.k <RETURN>
```

```
debug
input cmd_rds
start Prog(1000),Table(2000),Data(0FF800)
output cmd_rds
print cmd_rds
exit
```

Figure 2-2. Linkage Editor Subcommand File

Generate HP Absolute file

To generate HP Absolute file for the Softkey Interface, you need to use "**h83cnvhp**" absolute file format converter program. The h83cnvhp converter is provided with HP 64784 Softkey Interface. To generate HP Absolute file, enter following command:

```
$ h83cnvhp cmd_rds <RETURN>
```

You will see that cmd_rds.X, cmd_rds.L, and cmd_rds.A are generated. These are sufficient throughout this chapter.

Note 

You need to specify "debug" command line option to compiler, assembler and linker command to generate local symbol information. Otherwise, you will see the warning message when file format converter **h83cnvhp** is executed. And no local symbol file will be generated. The "debug" option for the compiler, assembler and linker direct to include local symbol information to the object file.

Entering the Softkey Interface

If you have installed your emulator and Softkey Interface software as directed in the *HP 64700 Series Emulators Softkey Interface Installation Notice*, you are ready to enter the interface. The Softkey Interface can be entered from the HP-UX shell.

From the HP-UX Shell

If **/usr/hp64000/bin** is specified in your PATH environment variable, you can also enter the Softkey Interface with the following command.

```
$ emul700 <emul_name> <RETURN>
```

The "emul_name" in the command above is the logical emulator name given in the HP 64700 emulator device table (/usr/hp64000/etc/64700tab or /usr/hp64000/etc/64700tab.net).

If this command is successful, you will see a display similar to figure 2-3. The status message shows that the default configuration file has been loaded. If the command is not successful, you will be given an error message and returned to the HP-UX prompt. Error messages are described in the *Softkey Interface Reference* manual.

```
#--------+------------+-----------+-----------------------------------------
# Channel|  Logical   | Processor | Remainder of Information for the Channel
#  Type  |   Name     |   Type    | (IP address for LAN connections)
#--------+------------+-----------+-----------------------------------------
  lan:       h8300       h6413003    21.17.9.143
```

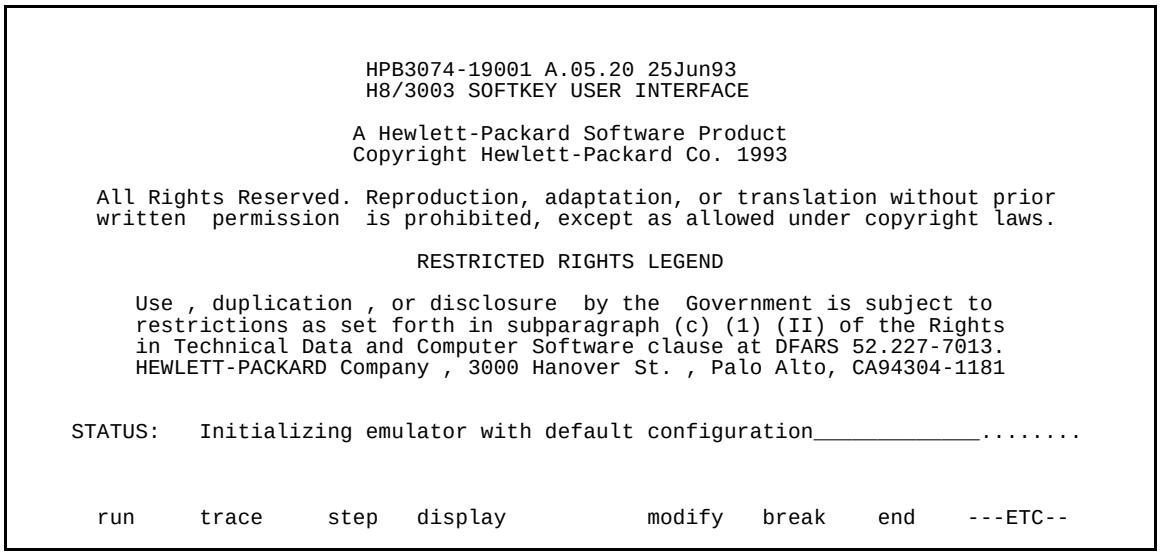


Figure 2-3. Softkey Interface Display

Using the Default Configuration

The default emulator configuration is used with the following examples. In this case, the H8/3003 emulator is configured to emulate H8/3042 chip and the address range 0 hex through ffff hex is mapped as emulation ROM.

Note

When you use internal ROM area, you **must** map that area as emulation memory. If you don't map internal ROM properly, you cannot access that area.

Note

Since the H8/3003 emulator automatically maps internal RAM as emulation RAM, you don't need to map this area.

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

On-Line Help

There are two ways to access on-line help in the Softkey Interface. The first is by using the Softkey Interface help facility. The second method allows you to access the firmware resident Terminal Interface on-line help information.

Softkey Driven Help

To access the Softkey Interface on-line help information, type either "help" or "?" on the command line; you will notice a new set of softkeys. By pressing one of these softkeys and <RETURN>, you can cause information on that topic to be displayed on your screen. For example, you can enter the following command to access "system command" help information.

? system_commands <RETURN>

```
---SYSTEM COMMANDS & COMMAND FILES---

?                          displays the possible help files
help                       displays the possible help files

!                          fork a shell (specified by  shell variable SH)
!<shell command>           fork a shell and execute a shell command

pwd                        print the working directory
cd <directory>             change the working directory

pws                        print the default symbol scope
cws <SYMB>                 change the working symbol - the working symbol also
                           gets updated when displaying local symbols and
                           displaying memory mnemonic


forward <UI> "command"     send the command in the quoted string from this user
                           interface to another one.  Replace <UI> with the name
                           the other user interface as shown on the softkeys:


--More--(15%)
```

The help information is scrolled on to the screen. If there is more than a screenful of information, you will have to press the space bar to see the next screenful, or the <RETURN> key to see the next line, just as you do with the HP-UX **more** command. After all the information on the particular topic has been displayed (or after you press "q" to quit scrolling through information), you are prompted to press <RETURN> to return to the Softkey Interface.

Pod Command Help

To access the emulator's firmware resident Terminal Interface help information, you can use the following commands.

display pod_command <RETURN>
pod_command 'help m' <RETURN>

The command enclosed in string delimiters (", ', or ^) is any Terminal Interface command, and the output of that command is seen in the pod_command display. The Terminal Interface help (or ?) command may be used to provide information on any Terminal Interface command or any of the emulator configuration options (as the example command above shows).

```
Pod Commands
  Time                Command

    m <addr>                - display memory at address
    m -d<dtype> <addr>      - display memory at address with display option
    m <addr>..<addr>        - display memory in specified address range
    m -dm <addr>..<addr>    - display memory mnemonics in specified range
    m <addr>..              - display 128 byte block starting at address A
    m <addr>=<value>        - modify memory at address to <value>
    m -d<dtype> <addr>=<value>       - modify memory with display option
    m <addr>=<value>,<value>         - modify memory to data sequence
    m <addr>..<addr>=<value>,<value> - fill range with repeating sequence

  --- VALID <dtype> MODE OPTIONS ---
    b - display size is 1 byte(s)
    w - display size is 2 byte(s)
    l - display size is 4 byte(s)
    m - display processor mnemonics

STATUS:   H8/3042--Running in monitor_____________________________________...R....
 pod_command 'help m'


pod_cmd    set    perfinit perfrun          perfend                    ---ETC--
```

Loading Absolute Files

The "load" command allows you to load absolute files into emulation or target system memory. If you wish to load only that portion of the absolute file that resides in memory mapped as emulation RAM or ROM, use the "load emul_mem" syntax. If you wish to load only the portion of the absolute file that resides in memory mapped as target RAM, use the "load user_mem" syntax. If you want both emulation and target memory to be loaded, do not specify "emul_mem" nor "user_mem". For example:

```
load cmd_rds <RETURN>
```

Normally, you will configure the emulator and map memory before you load the absolute file; however, the default configuration is sufficient for the sample program.

Displaying Symbols

When you load an absolute file into memory (unless you use the "nosymbols" option), symbol information is loaded. Both global symbols and symbols that are local to a source file can be displayed.

Global

To display global symbols, enter the following command.

```
display global_symbols <RETURN>
```

Listed are: address ranges associated with a symbol.

```
Global symbols in cmd_rds
Static symbols
Symbol name _____________________ Address range ___ Segment _____________ Offset
Cmd_Input                        0FF800                                    0000
Init                             001000                                    0000
Msg_Dest                         0FF802                                    0002
Msgs                             002000                                    0000

Filename symbols
Filename ____________________________________________________________________
cmd_rds.src









STATUS:  H8/3042-Running in monitor_____________________________________...R....
 display global_symbols

  run     trace     step   display           modify   break    end    ---ETC--
```

Local When displaying local symbols, you must include the name of the source file in which the symbols are defined. For example,

display local_symbols_in cmd_rds.src:
<RETURN>

```
Symbols in cmd_rds.src:
Static symbols
Symbol name _____________________ Address range ___ Segment _____________ Offset
Clear                            001006                                    0006
Clear_Loop                       001050                                    0050
Clear_Old                        00104E                                    004E
Cmd_A                            001028                                    0028
Cmd_B                            001034                                    0034
Cmd_I                            001040                                    0040
Cmd_Input                        0FF800                                    0000
Data                             0FF800                                    0000
END_Msgs                        00002031
Exe_Cmd                          001018                                    0018
Init                             001000                                    0000
Msg_A                            002000                                    0000
Msg_B                            002011                                    0011
Msg_Dest                         0FF802                                    0002
Msg_I                            002022                                    0022

STATUS:  cws: cmd_rds.src:_____________________________________________...R....
 display local_symbols_in cmd_rds.src:

  run     trace     step   display           modify   break    end    ---ETC--
```

Displaying Memory in Mnemonic Format

You can display, in mnemonic format, the absolute code in memory. For example, to display the memory of the "cmd_rds" program,

display memory Init ***mnemonic*** <RETURN>

Notice that you can use symbols when specifying expressions. The global symbol Init is used in the command above to specify the starting address of the memory to be displayed.

```
Memory  :mnemonic  :file = cmd_rds.src:
  address   data
   001000  7A07000FF9  MOV.L #000FF902,ER7
   001006  F800        MOV.B #00,R0L
   001008  6AA8000FF8  MOV.B R0L,@0FF800
   00100E  6A2A000FF8  MOV.B @0FF800,R2L
   001014  AA00        CMP.B #00,R2L
   001016  47F6        BEQ 00100E
   001018  AA41        CMP.B #41,R2L
   00101A  5870000A    BEQ 001028
   00101E  AA42        CMP.B #42,R2L
   001020  58700010    BEQ 001034
   001024  58000018    BRA 001040
   001028  FB11        MOV.B #11,R3L
   00102A  7A04000020  MOV.L #00002000,ER4
   001030  58000014    BRA 001048
   001034  FB11        MOV.B #11,R3L
   001036  7A04000020  MOV.L #00002011,ER4

STATUS:   H8/3042--Running in monitor_____________________________________...R....
 display  memory Init mnemonic


  run     trace     step   display          modify   break    end    ---ETC--
```

Displaying Memory with Symbols

You can include symbol information in memory display.

set symbols on <RETURN>

```
 Memory  :mnemonic  :file = cmd_rds.src:
   address  label          data
    001000     :Init      7A07000FF9  MOV.L #000FF902,ER7
    001006  cmd_rd:Clear  F800        MOV.B #00,R0L
    001008                6AA8000FF8  MOV.B R0L,@:Cmd_Input
    00100E  cmd_rds:Scan  6A2A000FF8  MOV.B @:Cmd_Input,R2L
    001014                AA00        CMP.B #00,R2L
    001016                47F6        BEQ cmd_rds.src:Scan
    001018  cmd_:Exe_Cmd  AA41        CMP.B #41,R2L
    00101A                5870000A    BEQ cmd_rds.sr:Cmd_A
    00101E                AA42        CMP.B #42,R2L
    001020                58700010    BEQ cmd_rds.sr:Cmd_B
    001024                58000018    BRA cmd_rds.sr:Cmd_I
    001028  cmd_rd:Cmd_A  FB11        MOV.B #11,R3L
    00102A                7A04000020  MOV.L #00002000,ER4
    001030                58000014    BRA cmd_rd:Write_Msg
    001034  cmd_rd:Cmd_B  FB11        MOV.B #11,R3L
    001036                7A04000020  MOV.L #00002011,ER4

STATUS:  H8/3042--Running in monitor_____________________________________...R....
 set symbols on


pod_cmd    set    perfinit perfrun          perfend                    ---ETC--
```

Note

The "**set**" command is effective only to the window which the command is invoked. When you access the emulator from multiple windows, you need to use the command at each window.

Running the Program

The "run" command lets the emulator execute a program in memory. Entering the "run" command by itself causes the emulator to begin executing at the current program counter address. The "run from" command allows you to specify an address at which execution is to start.

From Transfer Address

The "run from transfer_address" command specifies that the emulator start executing at a previously defined "start address". Transfer addresses are defined in assembly language source files with the .END assembler directive (i.e., pseudo instruction). For example, the sample program defines the address of the label Init as the transfer address. The following command will cause the emulator to execute from the address of the **Init** label.

```
run from transfer_address <RETURN>
```

From Reset

The "run from reset" command specifies that the emulator begin executing from target system reset (see "Running From Reset" section in the "In-Circuit Emulation" chapter).

Displaying Memory Repetitively

You can display memory locations repetitively so that the information on the screen is constantly updated. For example, to display the **Msg_Dest** locations of the sample program repetitively (in blocked byte format), enter the following command.

```
display memory Msg_Dest repetitively blocked
bytes <RETURN>
```

Modifying Memory

The sample program simulates a primitive command interpreter. Commands are sent to the sample program through a byte sized memory location labeled **Cmd_Input**. You can use the modify memory feature to send a command to the sample program. For example, to enter the command "A" (41 hex), use the following command.

```
modify memory Cmd_Input bytes to 41h <RETURN>
```

Or:

```
modify memory Cmd_Input string to 'A'
<RETURN>
```

```
 Memory  :bytes :access=bytes :blocked :repetitively
   address       data       :hex                                  :ascii
    0FF802-09    54   48    49   53   20   49   53   20      T H I S    I S
    0FF80A-11    4D   45    53   53   41   47   45   20      M E S S  A G E
    0FF812-19    41   00    00   00   00   00   00   00      A . . .  . . . .
    0FF81A-21    00   00    00   00   00   00   00   00      . . . .  . . . .
    0FF822-29    00   00    00   00   00   00   00   00      . . . .  . . . .
    0FF82A-31    00   00    00   00   00   00   00   00      . . . .  . . . .
    0FF832-39    00   00    00   00   00   00   00   00      . . . .  . . . .
    0FF83A-41    00   00    00   00   00   00   00   00      . . . .  . . . .
    0FF842-49    00   00    00   00   00   00   00   00      . . . .  . . . .
    0FF84A-51    00   00    00   00   00   00   00   00      . . . .  . . . .
    0FF852-59    00   00    00   00   00   00   00   00      . . . .  . . . .
    0FF85A-61    00   00    00   00   00   00   00   00      . . . .  . . . .
    0FF862-69    00   00    00   00   00   00   00   00      . . . .  . . . .
    0FF86A-71    00   00    00   00   00   00   00   00      . . . .  . . . .
    0FF872-79    00   00    00   00   00   00   00   00      . . . .  . . . .
    0FF87A-81    00   00    00   00   00   00   00   00      . . . .  . . . .

STATUS:   H8/3042--Running user program_____________________________...R....
 modify  memory Cmd_Input bytes to 41h


  run     trace     step   display           modify   break     end    ---ETC--
```

After the memory location is modified, the repetitive memory display shows that the "THIS IS MESSAGE A" message is written to the destination locations.

Breaking into the Monitor

The "break" command allows you to divert emulator execution from the user program to the monitor. You can continue user program execution with the "run" command. To break emulator execution from the sample program to the monitor, enter the following command.

```
break <RETURN>
```

Note

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

Using Software Breakpoints

Software breakpoints are provided with an H8/3003 special code; This special code (5770 hexadecimal) is H8/3003 undefined instruction.

When you define or enable a software breakpoint, the emulator will replace the opcode at the software breakpoint address with the special code.

Note

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

Note

Because software breakpoints are implemented by replacing opcodes with the special code, you cannot define software breakpoints in target ROM.

When software breakpoints are enabled and emulator detects a fetching the special code (5770 hexadecimal), it generates a break to background request which as with the "processor break" command. Since the system controller knows the locations of defined software breakpoints, it can determine whether the special code is software breakpoints or opcode in your target program.

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

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

When software breakpoints are disabled, the emulator replaces the special code with the original opcode.

Unlimited software breakpoints may be defined.

Enabling/Disabling Software Breakpoints

When you initially enter the Softkey Interface, software breakpoints are disabled. To enable the software breakpoints feature, enter the following command.

modify software_breakpoints enable <RETURN>

When software breakpoints are enabled and you set a software breakpoint, the H8/3003 special code (5770 hexadecimal) will be placed at the address specified. When the special code is executed, program execution will break into the monitor.

Setting a Software Breakpoint

To set a software breakpoint at the address of the Cmd_A label, enter the following command.

```
modify software_breakpoints set
cmd_rds.src:Cmd_A <RETURN>
```

Notice that when using local symbols in expressions, the source file in which the local symbol is defined must be included.

After the software breakpoint has been set, enter the following command to display memory and see if the software breakpoint was correctly inserted.

```
display memory Init mnemonic <RETURN>
```

```
 Memory  :mnemonic :file = cmd_rds.src:
   address  label          data
    001000     :Init      7A07000FF9  MOV.L #000FF902,ER7
    001006  cmd_rd:Clear  F800        MOV.B #00,R0L
    001008                6AA8000FF8  MOV.B R0L,@:Cmd_Input
    00100E  cmd_rds:Scan  6A2A000FF8  MOV.B @:Cmd_Input,R2L
    001014                AA00        CMP.B #00,R2L
    001016                47F6        BEQ cmd_rds.src:Scan
    001018  cmd_:Exe_Cmd  AA41        CMP.B #41,R2L
    00101A                5870000A    BEQ cmd_rds.sr:Cmd_A
    00101E                AA42        CMP.B #42,R2L
    001020                58700010    BEQ cmd_rds.sr:Cmd_B
    001024                58000018    BRA cmd_rds.sr:Cmd_I
*   001028  cmd_rd:Cmd_A  5770        Illegal Opcode
    00102A                7A04000020  MOV.L #00002000,ER4
    001030                58000014    BRA cmd_rd:Write_Msg
    001034  cmd_rd:Cmd_B  FB11        MOV.B #11,R3L
    001036                7A04000020  MOV.L #00002011,ER4

STATUS:   H8/3042--Running in monitor_____________________________...R....
 display  memory Init  mnemonic


  run     trace     step   display           modify   break     end    ---ETC--
```

As you can see, the software breakpoint is shown in the memory display with an asterisk.

Enter the following command to cause the emulator to continue executing the sample program.

```
run <RETURN>
```

Now, modify the command input byte to a valid command for the sample program.

```
modify memory Cmd_Input bytes to 41h <RETURN>
```

You will see the line of the software breakpoint is displayed in inverse-video. The inverse-video shows that the Program Counter is now at the address.

A message on the status line shows that the software breakpoint has been hit. The status line also shows that the emulator is now executing in the monitor.

Clearing a Software Breakpoint

To remove software breakpoint defined above, enter the following command.

```
modify software_breakpoints clear
cmd_rds.src:Cmd_A <RETURN>
```

The breakpoint is removed from the list, and the original opcode is restored if the breakpoint was pending.

To clear all software breakpoints, you can enter the following command.

```
modify software_breakpoints clear <RETURN>
```

Stepping Through the Program

The step command allows you to step through program execution an instruction or a number of instructions at a time. Also, you can step from the current program counter or from a specific address. To step through the example program from the address of the software breakpoint set earlier, enter the following command.

```
step <RETURN>, <RETURN>, ...
```

You will see the inverse-video moves according to the step execution. You can continue to step through the program just by pressing the <RETURN> key; when a command appears on the command line, it may be entered by pressing <RETURN>.

Displaying Registers

Enter the following command to display registers. You can display the basic registers class, or an individual register.

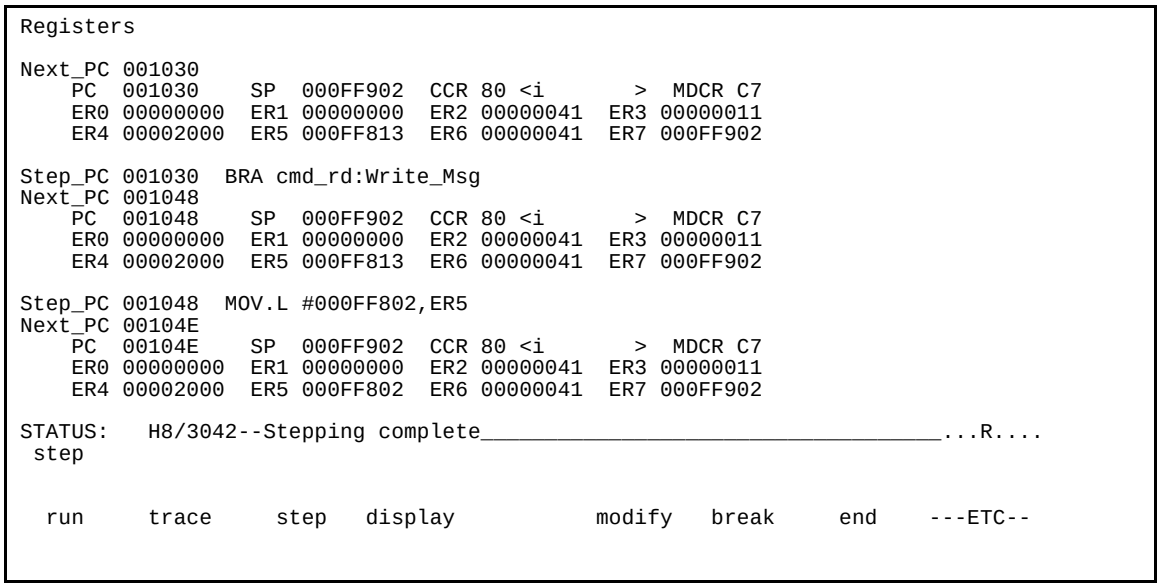
display registers <RETURN>

```
Registers

Next_PC 001030
    PC  001030    SP  000FF902  CCR 80 <i       >  MDCR C7
    ER0 00000000  ER1 00000000  ER2 00000041  ER3 00000011
    ER4 00002000  ER5 000FF813  ER6 00000041  ER7 000FF902









STATUS:   H8/3042--Stepping complete_____________________________________...R....
 display  registers

  run     trace     step   display           modify   break     end    ---ETC--
```

You can use "register class" and "register name" to display registers. Refer to the "Register Class and Name" section in Chapter 5.

When you enter the "**step**" command with registers displayed, the register display is updated every time you enter the command.

step <RETURN>, <RETURN>, <RETURN>



Enter the following command to cause sample program execution to continue from the current program counter.

run <RETURN>

Using the Analyzer

HP 64700 emulators contain an emulation analyzer. The emulation analyzer monitors the internal emulation lines (address, data, and status). Optionally, you may have an additional 16 trace signals which monitor external input lines. The analyzer collects data at each pulse of a clock signal, and saves the data (a trace state) if it meets a "storage qualification" condition.

Specifying a Simple Trigger

Suppose you want to trace program execution after the point at which the sample program reads the "B" (42 hex) command from the command input byte. To do this you would trace after the analyzer finds a state in which a value of 42xxh is read from the **Cmd_Input** byte. The following command makes this trace specification.

trace after Cmd_Input ***data*** 42xxh ***status read*** <RETURN>

The message "Emulation trace started" will appear on the status line. Now, modify the command input byte to "B" with the following command.

```
modify memory Cmd_Input bytes to 42h <RETURN>
```

The status line now shows "Emulation trace complete".

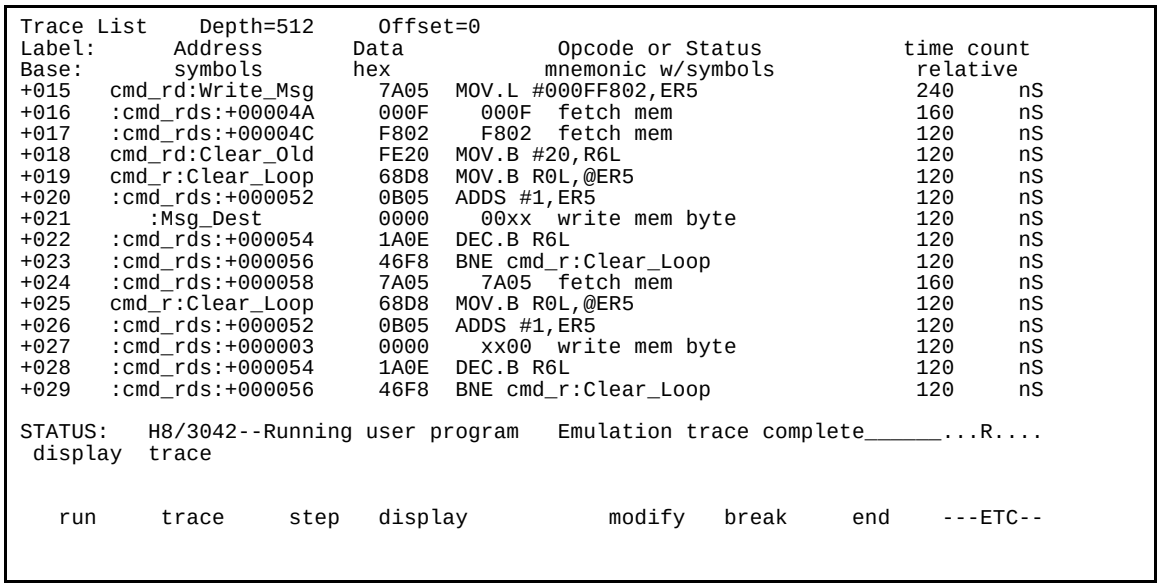
Displaying the Trace

The trace listings which follow are of program execution on the H8/3003 emulator. To display the trace, enter:

```
display trace <RETURN>
```

```
Trace List    Depth=512     Offset=0
Label:       Address       Data            Opcode or Status           time count
Base:        symbols       hex           mnemonic w/symbols           relative
after      :Cmd_Input        42FF    42xx  read  mem byte            ------------
+001    :cmd_rds:+000016     47F6  BEQ cmd_rds.src:Scan                 120     nS
+002    cmd_rds.:Exe_Cmd     AA41  CMP.B #41,R2L                       120     nS
+003    cmd_rds.src:Scan     6A2A    6A2A  unused fetch mem            160     nS
+004    :cmd_rds:+00001A     5870  BEQ cmd_rds.sr:Cmd_A                 120     nS
+005    :cmd_rds:+00001C     000A    000A  fetch mem                    120     nS
+006    :cmd_rds:+00001E     AA42  CMP.B #42,R2L                       240     nS
+007    :cmd_rds:+000020     5870  BEQ cmd_rds.sr:Cmd_B                 120     nS
+008    :cmd_rds:+000022     0010    0010  fetch mem                    120     nS
+009    cmd_rds.sr:Cmd_B     FB11  MOV.B #11,R3L                       280     nS
+010    :cmd_rds:+000036     7A04  MOV.L #00002011,ER4                 120     nS
+011    :cmd_rds:+000038     0000    0000  fetch mem                    120     nS
+012    :cmd_rds:+00003A     2011    2011  fetch mem                    120     nS
+013    :cmd_rds:+00003C     5800  BRA cmd_rd:Write_Msg                 120     nS
+014    :cmd_rds:+00003E     0008    0008  fetch mem                    120     nS

STATUS:   H8/3042--Running user program   Emulation trace complete______...R....
display trace


  run     trace     step   display           modify   break     end    ---ETC--
```

Line 0 (labeled "after") in the trace list above shows the state which triggered the analyzer. The trigger state is always on line 0. The other states show the exit from the **Scan** loop and the **Exe_Cmd** and **Cmd_B** instructions. To list the next lines of the trace, press the <PGDN> or <NEXT> key.



```
Trace List    Depth=512     Offset=0
Label:      Address       Data           Opcode or Status          time count
Base:       symbols       hex           mnemonic w/symbols          relative
+015   cmd_rd:Write_Msg     7A05  MOV.L #000FF802,ER5                 240     nS
+016   :cmd_rds:+00004A      000F    000F  fetch mem                   160     nS
+017   :cmd_rds:+00004C      F802    F802  fetch mem                   120     nS
+018   cmd_rd:Clear_Old      FE20  MOV.B #20,R6L                      120     nS
+019   cmd_r:Clear_Loop      68D8  MOV.B R0L,@ER5                     120     nS
+020   :cmd_rds:+000052      0B05  ADDS #1,ER5                       120     nS
+021      :Msg_Dest          0000    00xx  write mem byte            120     nS
+022   :cmd_rds:+000054      1A0E  DEC.B R6L                         120     nS
+023   :cmd_rds:+000056      46F8  BNE cmd_r:Clear_Loop              120     nS
+024   :cmd_rds:+000058      7A05    7A05  fetch mem                 160     nS
+025   cmd_r:Clear_Loop      68D8  MOV.B R0L,@ER5                     120     nS
+026   :cmd_rds:+000052      0B05  ADDS #1,ER5                       120     nS
+027   :cmd_rds:+000003      0000    xx00  write mem byte            120     nS
+028   :cmd_rds:+000054      1A0E  DEC.B R6L                         120     nS
+029   :cmd_rds:+000056      46F8  BNE cmd_r:Clear_Loop              120     nS

STATUS:   H8/3042--Running user program   Emulation trace complete_____...R....
 display  trace


   run     trace     step   display           modify   break     end    ---ETC--
```

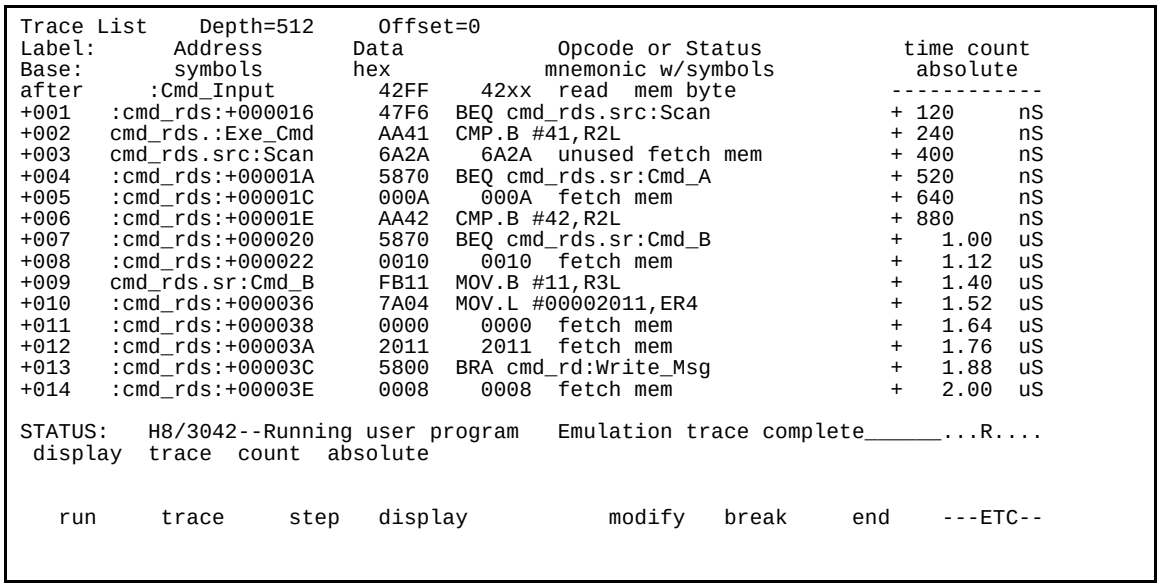
The resulting display shows **Cmd_B** instructions, the branch to **Write_Msg** and the beginning of the instructions which move the "THIS IS MESSAGE B" message to the destination locations.

To list the previous lines of the trace, press the <PGUP> or <PREV> key.

Displaying Trace with Time Count Absolute

Enter the following command to display count information absolute from the trigger state.

display trace count absolute <RETURN>



```
Trace List    Depth=512     Offset=0
Label:      Address       Data            Opcode or Status          time count
Base:       symbols       hex           mnemonic w/symbols          absolute
after     :Cmd_Input        42FF    42xx  read  mem byte           -----------
+001   :cmd_rds:+000016     47F6  BEQ cmd_rds.src:Scan             + 120     nS
+002   cmd_rds.:Exe_Cmd     AA41  CMP.B #41,R2L                    + 240     nS
+003   cmd_rds.src:Scan     6A2A    6A2A  unused fetch mem         + 400     nS
+004   :cmd_rds:+00001A     5870  BEQ cmd_rds.sr:Cmd_A             + 520     nS
+005   :cmd_rds:+00001C     000A    000A  fetch mem                + 640     nS
+006   :cmd_rds:+00001E     AA42  CMP.B #42,R2L                    + 880     nS
+007   :cmd_rds:+000020     5870  BEQ cmd_rds.sr:Cmd_B             +   1.00  uS
+008   :cmd_rds:+000022     0010    0010  fetch mem                +   1.12  uS
+009   cmd_rds.sr:Cmd_B     FB11  MOV.B #11,R3L                    +   1.40  uS
+010   :cmd_rds:+000036     7A04  MOV.L #00002011,ER4              +   1.52  uS
+011   :cmd_rds:+000038     0000    0000  fetch mem                +   1.64  uS
+012   :cmd_rds:+00003A     2011    2011  fetch mem                +   1.76  uS
+013   :cmd_rds:+00003C     5800  BRA cmd_rd:Write_Msg             +   1.88  uS
+014   :cmd_rds:+00003E     0008    0008  fetch mem                +   2.00  uS

STATUS:   H8/3042--Running user program   Emulation trace complete_____...R....
 display  trace  count  absolute


   run     trace     step   display           modify   break     end    ---ETC--
```

H8/3003 Analysis Status Qualifiers

The status qualifier "read" was used in the example trace command used above. The following analysis status qualifiers may also be used with the H8/3003 emulator.

Qualifier	Status Bits (40..57)	Description
backgrnd	xx x0xx xxxx xxxx xxxxB	Background cycle
byte	xx xxxx 1xxx x1xx xx1xB	Byte access
cpu	xx xxxx 1xxx x11x xxxxB	CPU access
data	xx xxxx 1xxx x1x1 xxxxB	Data access
dma	xx xxxx 1xxx x10x xxxxB	DMA memory access
fetch	xx xxxx 1x1x x110 xx01B	Fetch cycle
foregrnd	xx x1xx xxxx xxxx xxxxB	Foreground cycle
grd	xx xx01 1xxx x1xx 1xxxB	Guarded memory access
intack	xx xxxx x0xx xxxx xxxxB	Interrupt acknowledge cycle
io	xx xxxx 1xxx x1xx 0xxxB	Internal I/O access
memory	xx xxxx 1xxx x1xx 1xxxB	Memory access
nointack	xx xxxx x1xx xxxx xxxxB	No interrupt acknowledge cycle
read	xx xxxx 1xxx x1xx xxx1B	Read cycle
refresh	xx xxxx 1xxx x01x xxxxB	Refresh cycle
word	xx xxxx 1xxx x1xx xx0xB	Word access
write	xx xxxx 1xxx x1xx xxx0B	Write cycle
wrrom	xx xx10 1xxx x1xx 1xx0B	Write to ROM cycle

Trace Analysis Considerations

There are some points to be noticed when you use the emulation analyzer.

How to Specify Trigger Condition

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

```
trace after cmd_rds.src:Exe_Cmd <RETURN>
modify memory Cmd_Input bytes to 41h <RETURN>
```

(Actually trace will be completed before you enter "modify memory" command)

You will see:

```
Trace List    Depth=512     Offset=0
Label:      Address       Data            Opcode or Status          time count
Base:       symbols       hex           mnemonic w/symbols          absolute
after  cmd_rds.:Exe_Cmd     AA41    AA41  fetch mem                 ------------
+001   cmd_rds.src:Scan     6A2A  MOV.B @:Cmd_Input,R2L             + 120     nS
+002   :cmd_rds:+000010     000F    000F  fetch mem                 + 240     nS
+003   :cmd_rds:+000012     F800    F800  fetch mem                 + 400     nS
+004   :cmd_rds:+000014     AA00  CMP.B #00,R2L                     + 520     nS
+005      :Cmd_Input        00FF    00xx  read  mem byte            + 640     nS
+006   :cmd_rds:+000016     47F6  BEQ cmd_rds.src:Scan              + 760     nS
+007   cmd_rds.:Exe_Cmd     AA41    AA41  fetch mem                 + 880     nS
+008   cmd_rds.src:Scan     6A2A  MOV.B @:Cmd_Input,R2L             +   1.00  uS
+009   :cmd_rds:+000010     000F    000F  fetch mem                 +   1.12  uS
+010   :cmd_rds:+000012     F800    F800  fetch mem                 +   1.24  uS
+011   :cmd_rds:+000014     AA00  CMP.B #00,R2L                     +   1.40  uS
+012      :Cmd_Input        00FF    00xx  read  mem byte            +   1.52  uS
+013   :cmd_rds:+000016     47F6  BEQ cmd_rds.src:Scan              +   1.64  uS
+014   cmd_rds.:Exe_Cmd     AA41    AA41  fetch mem                 +   1.76  uS

STATUS:   H8/3042--Running user program   Emulation trace complete_____...R....
 trace after cmd_rds.src:Exe_Cmd


   run     trace     step   display           modify   break     end    ---ETC--
```

This is not what we were expecting to see. As you can see at the first line of the trace list, the address of **Exe_Cmd** routine appears on the address bus during the program executing **Scan** loop. This made the emulation analyzer start trace. To avoid mis-trigger by this cause, set the trigger condition to the second instruction of the routine you want to trace:

```
trace after cmd_rds.src:Exe_Cmd+2 <RETURN>
```

(Since the instruction at **Exe_Cmd** label is two bytes instruction, the next instruction starts from **Exe_Cmd+2**.)

modify memory Cmd_Input ***bytes to*** 41h <RETURN>

```
Trace List    Depth=512     Offset=0
Label:      Address       Data            Opcode or Status           time count
Base:       symbols       hex            mnemonic w/symbols           absolute
after  :cmd_rds:+00001A     5870  BEQ cmd_rds.sr:Cmd_A               ------------
+001   :cmd_rds:+00001C     000A    000A  fetch mem                  + 120     nS
+002   cmd_rds.sr:Cmd_A     FB11  MOV.B #11,R3L                      + 360     nS
+003   :cmd_rds:+00002A     7A04  MOV.L #00002000,ER4                + 480     nS
+004   :cmd_rds:+00002C     0000    0000  fetch mem                  + 600     nS
+005   :cmd_rds:+00002E     2000    2000  fetch mem                  + 720     nS
+006   :cmd_rds:+000030     5800  BRA cmd_rd:Write_Msg               + 880     nS
+007   :cmd_rds:+000032     0014    0014  fetch mem                  +   1.00  uS
+008   cmd_rd:Write_Msg     7A05  MOV.L #000FF802,ER5                +   1.24  uS
+009   :cmd_rds:+00004A     000F    000F  fetch mem                  +   1.36  uS
+010   :cmd_rds:+00004C     F802    F802  fetch mem                  +   1.48  uS
+011   cmd_rd:Clear_Old     FE20  MOV.B #20,R6L                      +   1.60  uS
+012   cmd_r:Clear_Loop     68D8  MOV.B R0L,@ER5                     +   1.72  uS
+013   :cmd_rds:+000052     0B05  ADDS #1,ER5                        +   1.84  uS
+014      :Msg_Dest         0000    00xx  write mem byte             +   2.00  uS

STATUS:   H8/3042--Running user program   Emulation trace complete_____...R....
 modify  memory Cmd_Input  bytes  to 41h

  run     trace     step   display           modify   break     end    ---ETC--
```

If you need to see the execution of the instruction at **Exe_Cmd** label, use **trace about** command instead of **trace after** command. When you use the **trace about** command, the state which triggered the analyzer will appear in the center of the trace list.

Store Condition and Trace

When you specify store condition with **trace only** command, disassembling of program execution is unreliable.

trace <RETURN>

```
Trace List    Depth=512     Offset=0
Label:      Address       Data            Opcode or Status          time count
Base:       symbols       hex           mnemonic w/symbols          absolute
after  :cmd_rds:+000016     47F6  BEQ cmd_rds.src:Scan             ------------
+001   cmd_rds.:Exe_Cmd     AA41    AA41  fetch mem                 + 120     nS
+002   cmd_rds.src:Scan     6A2A  MOV.B @:Cmd_Input,R2L             + 240     nS
+003   :cmd_rds:+000010     000F    000F  fetch mem                 + 400     nS
+004   :cmd_rds:+000012     F800    F800  fetch mem                 + 520     nS
+005   :cmd_rds:+000014     AA00  CMP.B #00,R2L                     + 640     nS
+006      :Cmd_Input        00FF    00xx  read  mem byte            + 760     nS
+007   :cmd_rds:+000016     47F6  BEQ cmd_rds.src:Scan              + 880     nS
+008   cmd_rds.:Exe_Cmd     AA41    AA41  fetch mem                 +   1.00  uS
+009   cmd_rds.src:Scan     6A2A  MOV.B @:Cmd_Input,R2L             +   1.12  uS
+010   :cmd_rds:+000010     000F    000F  fetch mem                 +   1.24  uS
+011   :cmd_rds:+000012     F800    F800  fetch mem                 +   1.40  uS
+012   :cmd_rds:+000014     AA00  CMP.B #00,R2L                     +   1.52  uS
+013      :Cmd_Input        00FF    00xx  read  mem byte            +   1.64  uS
+014   :cmd_rds:+000016     47F6  BEQ cmd_rds.src:Scan              +   1.76  uS

STATUS:   H8/3042--Running user program   Emulation trace complete_____...R....
 trace


   run     trace     step   display           modify   break     end    ---ETC--
```

The program is executing the **Scan** loop.

Now, trace only accesses to the address range **Init** through **Init+0ffh**.

trace only range Init ***thru*** Init+0ffh <RETURN>

```
Trace List    Depth=512     Offset=0
Label:      Address       Data            Opcode or Status          time count
Base:       symbols       hex           mnemonic w/symbols          absolute
after  cmd_rds.:Exe_Cmd     AA41    AA41  fetch mem                 ------------
+001   cmd_rds.src:Scan     6A2A  MOV.B @:Cmd_Input,R2L             + 160     nS
+002   :cmd_rds:+000010     000F    000F  fetch mem                 + 280     nS
+003   :cmd_rds:+000012     F800    F800  fetch mem                 + 400     nS
+004   :cmd_rds:+000014     AA00    AA00  fetch mem                 + 520     nS
+005   :cmd_rds:+000016     47F6  BEQ cmd_rds.src:Scan              + 760     nS
+006   cmd_rds.:Exe_Cmd     AA41    AA41  fetch mem                 + 880     nS
+007   cmd_rds.src:Scan     6A2A  MOV.B @:Cmd_Input,R2L             +   1.04  uS
+008   :cmd_rds:+000010     000F    000F  fetch mem                 +   1.16  uS
+009   :cmd_rds:+000012     F800    F800  fetch mem                 +   1.28  uS
+010   :cmd_rds:+000014     AA00    AA00  fetch mem                 +   1.40  uS
+011   :cmd_rds:+000016     47F6  BEQ cmd_rds.src:Scan              +   1.64  uS
+012   cmd_rds.:Exe_Cmd     AA41    AA41  fetch mem                 +   1.76  uS
+013   cmd_rds.src:Scan     6A2A  MOV.B @:Cmd_Input,R2L             +   1.88  uS
+014   :cmd_rds:+000010     000F    000F  fetch mem                 +   2.00  uS

STATUS:   H8/3042--Running user program   Emulation trace complete_____...R....
 trace  only  range Init  thru Init+0ffh


   run     trace     step   display           modify   break     end    ---ETC--
```

As you can see the execution of CMP.B instructions 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 **trace only** command.

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.

trace after data <data> <RETURN>

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:

trace after data 1234h <RETURN>

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

trace after data 12xxh <RETURN>

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

trace after data xx12h <RETURN>

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.

For a Complete Description

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

Exiting the Softkey Interface

There are several options available when exiting the Softkey Interface: exiting and releasing the emulation system, exiting with the intent of reentering (continuing), exiting locked from multiple emulation windows, and exiting (locked) and selecting the measurement system display or another module.

End Release System

To exit the Softkey Interface, releasing the emulator so that other users may use the emulator, enter the following command.

end release_system <RETURN>

Ending to Continue Later

You may also exit the Softkey Interface without specifying any options; this causes the emulator to be locked. When the emulator is locked, other users are prevented from using it and the emulator configuration is saved so that it can be restored the next time you enter (continue) the Softkey Interface.

end <RETURN>

Ending Locked from All Windows

When using the Softkey Interface from within window systems, the "end" command with no options causes an exit only in that window. To end locked from all windows, enter the following command.

end locked <RETURN>

This option only appears when you enter the Softkey Interface via the emul700 command. When you enter the Softkey Interface via MEAS_SYS, only one window is permitted.

Refer to the *Softkey Interface Reference* manual for more information on using the Softkey Interface with window systems.

Selecting the Measurement System Display or Another Module

When you enter the Softkey Interface via MEAS_SYS, you have the option to select the measurement system display or another module in the measurement system when exiting the Softkey Interface. This type of exit is also "locked"; that is, you can continue the emulation session later. For example, to exit and select the measurement system display, enter the following command.

```
end select measurement_system <RETURN>
```

This option is not available if you have entered the Softkey Interface via the emul700 command.

Notes

3

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/adapter 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.

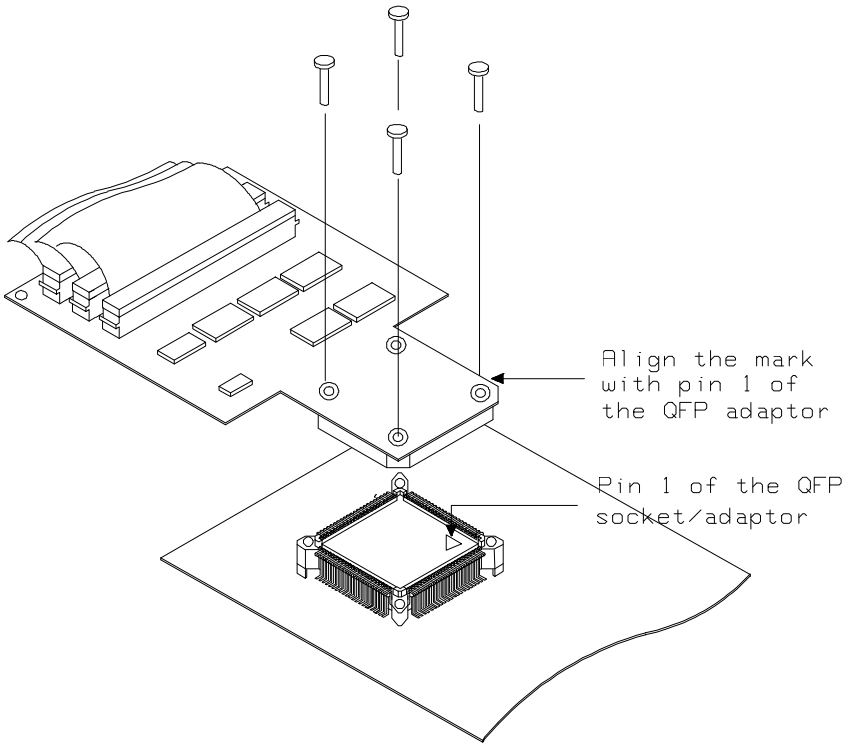


Figure 3-1. Installing the QFP adaptor

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.

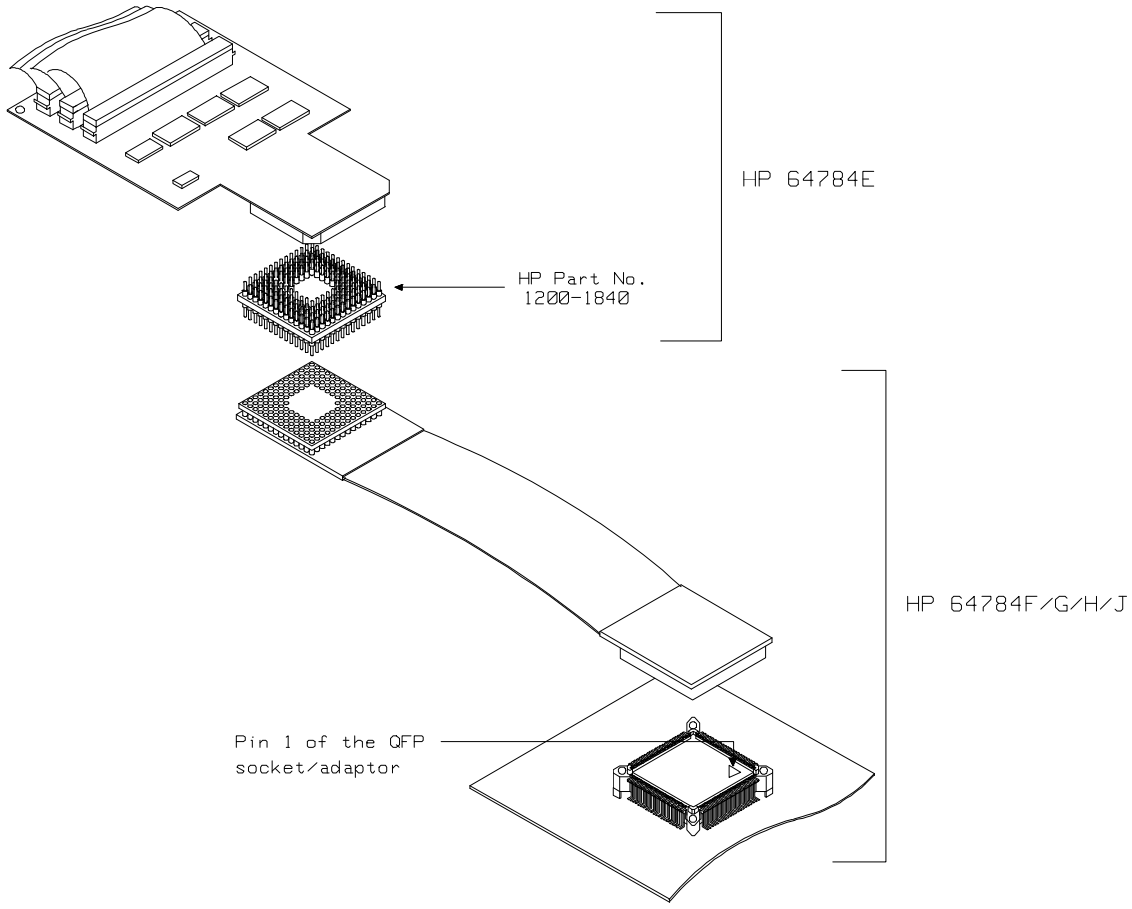


Figure 3-2 Installing the PGA adaptor (General)

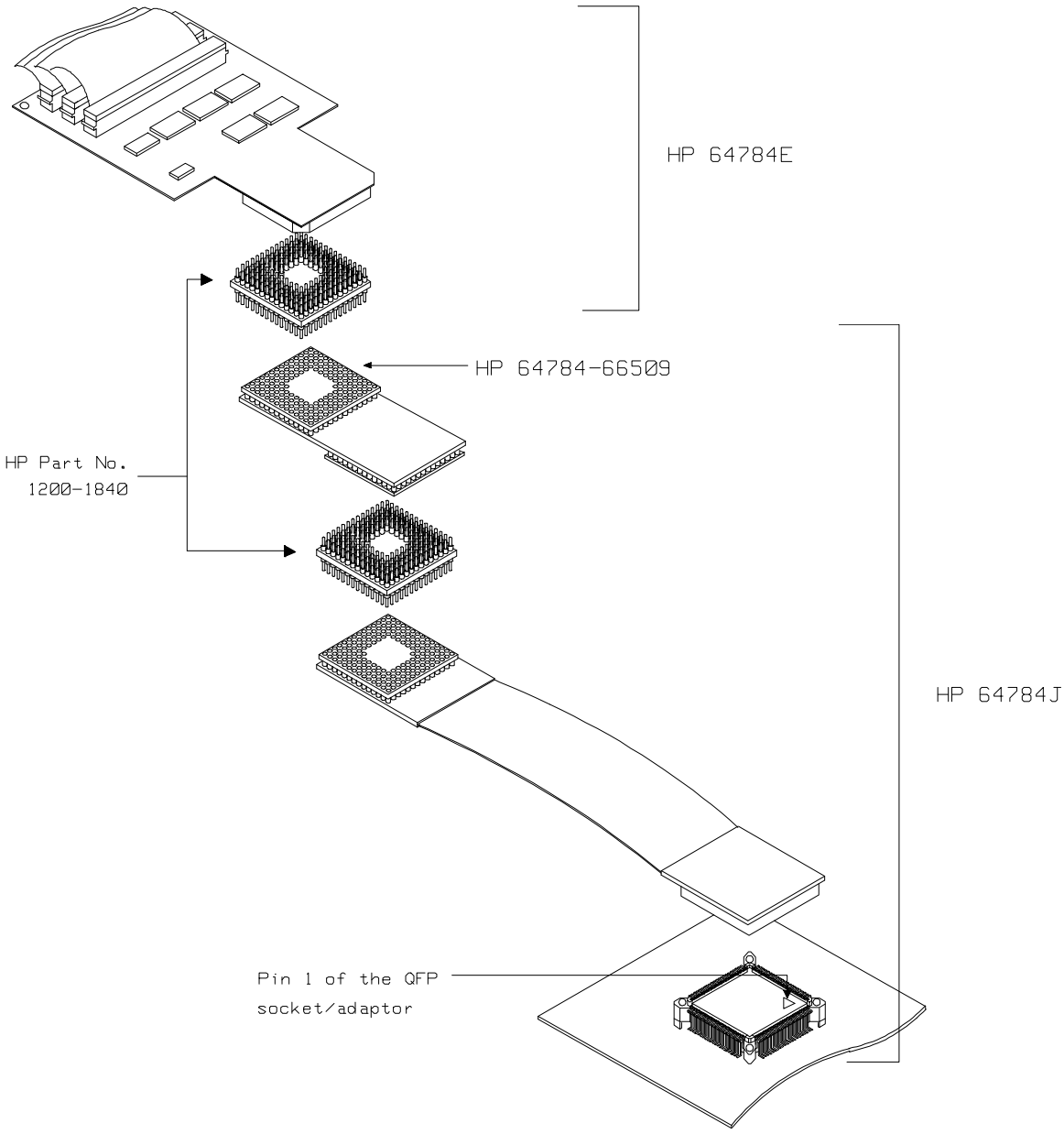


Figure 3-3 Installing the PGA adaptor (3001 mode 3/4)

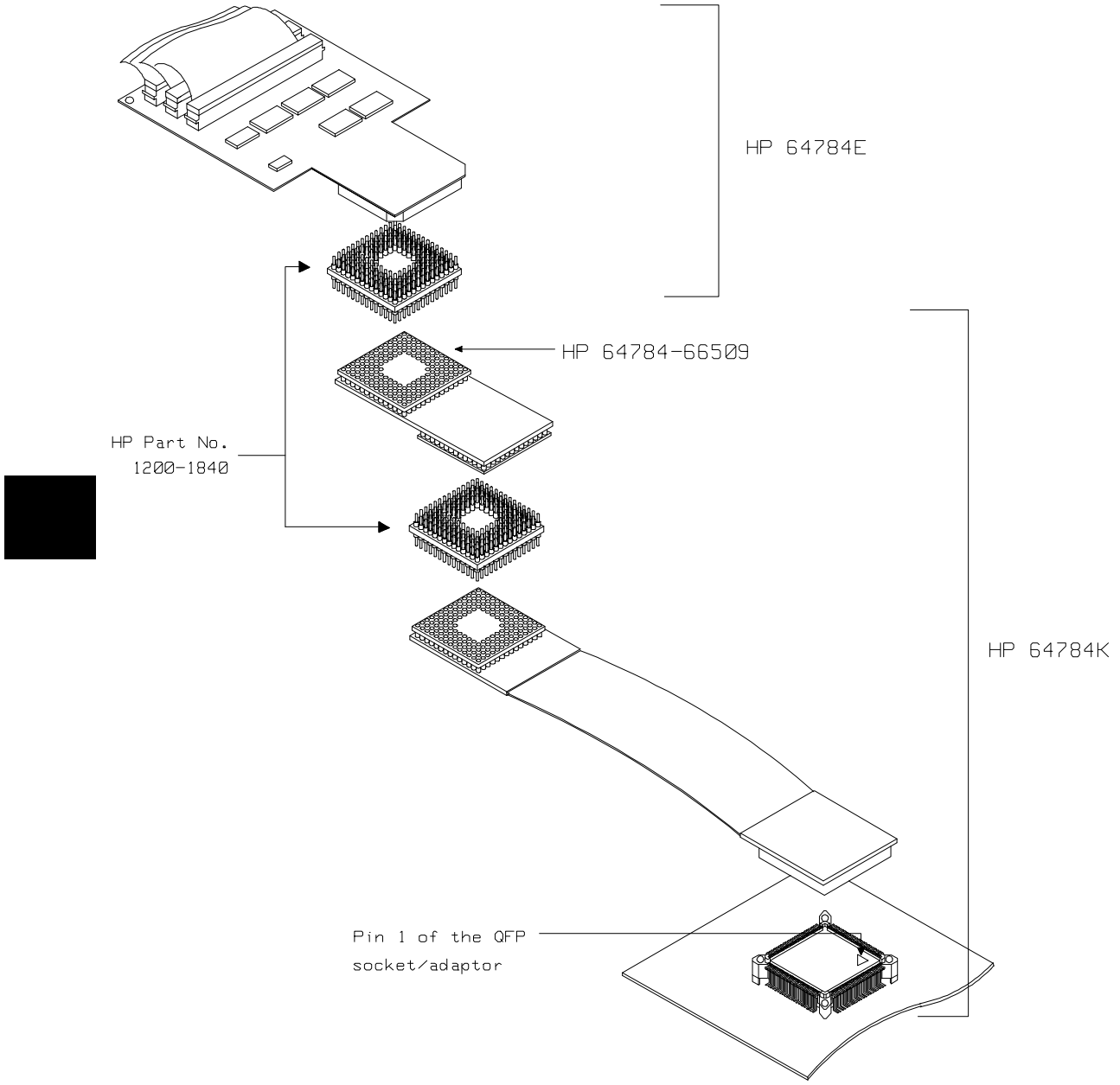


Figure 3-4 Installing the PGA adaptor (3004/05 mode 3)

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.

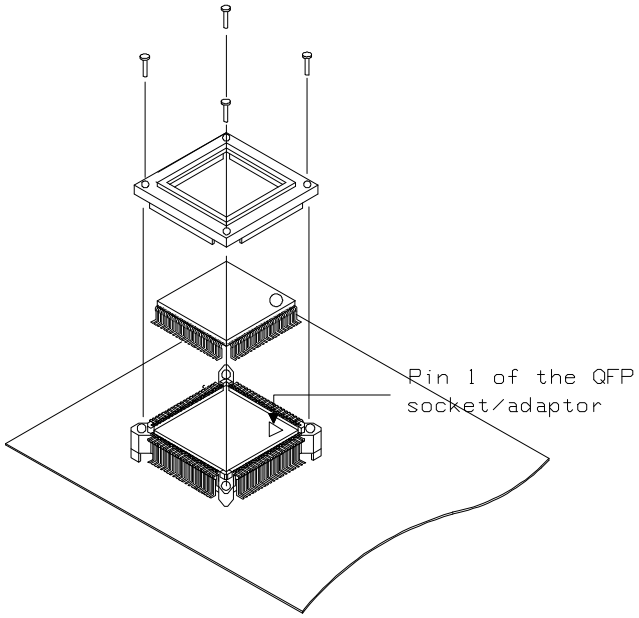


Figure 3-5 Installing the H8/3003 microprocessor

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(Vih) 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	Vcc x 0.7 or 2.4 *1
Others	Vcc x 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.

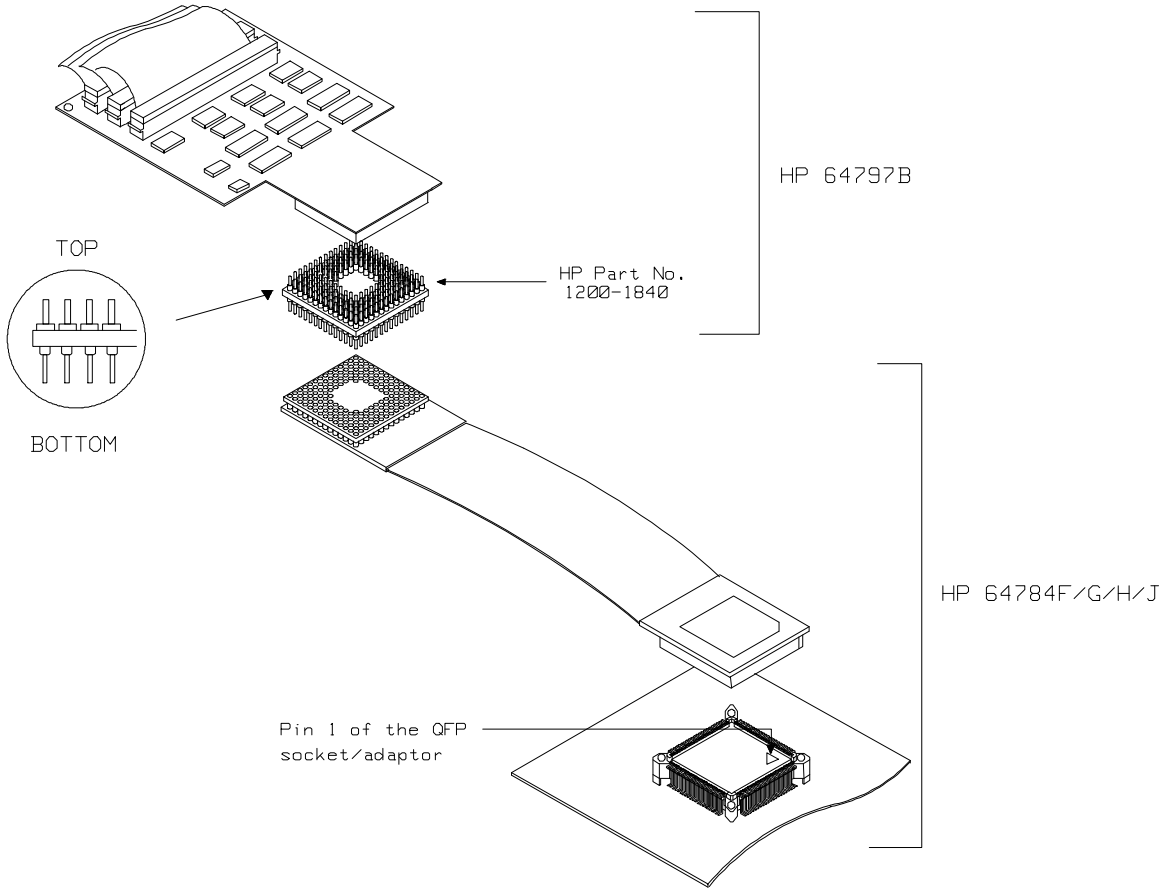


Figure 3-6 Installing the PGA adaptor (General)

Note

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

In-Circuit Configuration Options

The H8/3003 emulator provides configuration options for the following in-circuit emulation issues.

Refer to the Chapter 4 "Configuring the Emulator" for more information on these configuration options.

Using the Target System Clock Source

You can configure the emulator to use the external target system clock source.

Enabling Bus Arbitration

You can configure the emulator to enable/disable bus arbitration.

Enabling NMI from the Target

You can configure the emulator to accept/ignore NMI from the target system.

Enabling /RES signal from the Target

You can configure the emulator to accept/ignore /RES signal from the target system.

Enabling /RESO signal to the Target

You can configure the emulator to drive /RESO signal during watch dog timer reset.

Selecting Visible/Hidden Background Cycles

Emulation processor activity while executing in background can either be visible to target system (cycles are sent to the target system probe) or hidden (cycles are not sent to the target system probe).

Selecting Target Memory Access Size

You can specify the types of cycles that the emulation monitor uses when accessing target system memory.

Target System Interface and Timing Specification

Refer to the *H8/3003 Terminal Interface User's Guide* for information on the target system interface and timing specification of the H8/3003 emulator.

Running the Emulator from Target Reset

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

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

To specify a run from target system reset, select:

```
run from reset <RESET>
```

The status now shows that the emulator is "Awaiting target reset". After the target system is reset, the status line message will change to show the appropriate emulator status.

Notes

4

Configuring the Emulator

Introduction

Your H8/3003 emulator can be used in all stages of target system development. For instance, you can run the emulator out-of-circuit when developing your target system software, or you can use the emulator in-circuit when integrating software with target system hardware. You can use the emulator's internal clock or the target system clock. Emulation memory can be used in place of, or along with, target system memory. You can execute target programs in real-time or allow emulator execution to be diverted into the monitor when commands request access of target system resources (target system memory, register contents, etc).

The emulator is a flexible instrument and may be configured to suit your needs at any stage of the development process. This chapter describes the options available when configuring the H8/3003 emulator.

The configuration options are accessed with the following command.

modify configuration <RETURN>

After entering the command above, you will be asked questions regarding the emulator configuration. The configuration questions are listed below and grouped into the following classes.

General Emulator Configuration:

- Specifying the emulator clock source (internal/external).
- Selecting monitor entry after configuration.
- Restricting to real-time execution.
- Selecting the microprocessor to be emulated.
- Selecting the microprocessor operation mode.

Memory Configuration:

- Mapping memory.

Emulator Pod Configuration:

- Enabling emulator bus arbitration.
- Enabling NMI input from the target system.
- Enabling reset input from the target system.
- Allowing the emulator to drive reset output to the target system.
- Allowing the emulator to drive background cycles to the target systems.
- Setting up the reset value for the stack pointer.
- Selecting target memory access size.

Debug/Trace Configuration:

- Enabling breaks on writes to ROM.
- Specifying tracing of foreground/background cycles.
- Enabling tracing internal DMA cycles.
- Enabling tracing refresh cycles.

Simulated I/O Configuration: Simulated I/O is described in the *Simulated I/O reference* manual.

Interactive Measurement Configuration: See the chapter on coordinated measurements in the *Softkey Interface Reference* manual.

External Analyzer Configuration: See the *Analyzer Softkey Interface User's Guide.*

General Emulator Configuration

The configuration questions described in this section involve general emulator operation.

Micro-processor clock source?

This configuration question allows you to select whether the emulator will be clocked by the internal clock source or by a target system clock source.

internal Selects the internal clock oscillator as the emulator clock source.

external Selects the clock input to the emulation 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 H8/3030 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 H8/3030 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)

Note

Changing the clock source drives the emulator into the reset state. The emulator may later break into the monitor depending on how the following "Enter monitor after configuration?" question is answered.

Enter monitor after configuration?

This question allows you to select whether the emulator will be running in the monitor or held in the reset state upon completion of the emulator configuration.

How you answer this configuration question is important in some situations. For example, when the external clock has been selected and the target system is turned off, reset to monitor should not be selected; otherwise, configuration will fail.

When an external clock source is specified, this question becomes "Enter monitor after configuration (using external clock)?" and the default answer becomes "no".

yes — When reset to monitor is selected, the emulator will be running in the monitor after configuration is complete. If the reset to monitor fails, the previous configuration will be restored.

no — After the configuration is complete, the emulator will be held in the reset state.

Restrict to real-time runs?

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

no — All commands, regardless of whether or not they require a break to the emulation monitor, are accepted by the emulator.

yes — When runs are restricted to real-time and the emulator is running the user program, all commands that cause a break (except "reset", "break", "run", and "step") are refused. For example, the following commands are not allowed when runs are restricted to real-time:

- Display/modify registers.
- Display/modify internal I/O registers.
- Display/modify target system memory.
- Load/store target system memory.

Caution 

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

Processor series?

This configuration defines the microprocessor series to be emulated.

3003 When you are going to emulate H8/3003 series microprocessor, select this item.

3032 When you are going to emulate H8/3032 series microprocessor, select this item.

3042 When you are going to emulate H8/3042 series microprocessor, select this item.

Processor type?

This configuration defines the microprocessor type to be emulated.

H8/3003 series

3001 When you are going to emulate H8/3001 microprocessor, select this item.

3002 When you are going to emulate H8/3002 microprocessor, select this item.

3003T When you are going to emulate H8/3003 microprocessor, select this item.

3003 When you are going to emulate H8/3003 microprocessor with system clock divider, select this item.

3004 When you are going to emulate H8/3004 microprocessor, select this item.

3005 When you are going to emulate H8/3005 microprocessor, select this item.

H8/3032 series

3032 When you are going to emulate H8/3032 microprocessor, select this item.

3031 When you are going to emulate H8/3031 microprocessor, select this item.

3030 When you are going to emulate H8/3030 microprocessor, select this item.

H8/3042 series

3040 When you are going to emulate H8/3040 microprocessor, select this item.

3041 When you are going to emulate H8/3041 microprocessor, select this item.

3042 When you are going to emulate H8/3042 microprocessor, select this item.

Note 

Configuring this item will drive the emulator into the reset state.

Source for processor operation mode?

This configuration defines operation mode in which the emulator works.

internal	The emulator will work in selected operation mode regardless the setting by the target system.

Operation mode	**Description**
mode_1	The emulator will operate in mode 1. (expanded 1M bytes mode without internal ROM: 8 bit data bus)
mode_2	The emulator will operate in mode 2. (expanded 1M bytes mode without internal ROM:16 bit data bus)
mode_3	The emulator will operate in mode 3. (expanded 16M bytes mode without internal ROM: 8 bit data bus)
mode_4	The emulator will operate in mode 4. (expanded 16M bytes mode without internal ROM:16 bit data bus)
mode_5	The emulator will operate in mode 5. (expanded 1M bytes mode with internal ROM: 8 bit data bus)
mode_6	The emulator will operate in mode 6. (single chip normal mode)
mode_7	The emulator will operate in mode 7. (single chip advanced mode)

Note

When you configure to emulate 3003 series microprocessor, mode_5, mode_6 and mode_7 don't appear in this configuration.

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

Note 

When you emulate H8/3004/05, **external** mode is not available. You have to configure processor mode using **internal** mode.

Note 

It is recommended to configure this item as internal mode and select operation mode, since the emulator dose not work fine when MD0,MD1 and MD2 are not steady.

Note 

Configuring this item will drive the emulator into the reset state.

Memory Configuration

The memory configuration questions allow you to select the monitor type and to map memory. To access the memory configuration questions, you must answer "yes" to the following question.

Modify memory configuration?

Mapping Memory

The H8/3003 emulator contains high-speed emulation memory (no wait states required) that can be mapped at a resolution of 512 bytes.

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

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

Note

Direct memory access to emulation memory using external DMAC are not allowed.

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

The default emulator configuration maps location 0 hex through FFFF hex as emulation ROM.

Note

When you use internal ROM area, you **must** map that area as emulation memory. 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 and 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 display/modify memory commands.

Blocks of memory can also be characterized as guarded memory. Guarded memory accesses will generate "break to monitor" requests. Writes to ROM will generate "break to monitor" requests if the "Enable breaks on writes to ROM?" configuration item is enabled (see the "Debug/Trace Configuration" section which follows).

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 2fff 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:

```
delete all <RETURN>
1000h thru 2fffh emulation rom <RETURN>
0f000h thru 0f1ffh emulation ram <RETURN>
end <RETURN>
```

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

Note

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

Note

Configuring memory mapping will drive the emulator into the reset state.

Emulator Pod Configuration

To access the emulator pod configuration questions, you must answer "yes" to the following question.

Modify emulator pod configuration?

Enable bus arbitration?

The bus arbitration configuration question defines how your emulator responds to bus request signals from the target system during both foreground and background operation.

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

Note

You cannot perform DMA (direct memory access) transfers between your target system and emulation memory by using external DMA controller on your target system; the H8/3003 emulator does not support such a feature.

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

Enabling and disabling bus master arbitration can be useful to you in isolating target system problems. For example, you may have a situation where the processor never seems to execute any code. You

can disable bus arbitration to check and see if faulty arbitration circuitry in your target system is contributing to the problem.

Note 

This question does not appear when you select mode_6 or mode_7.

Note 

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

Note 

Configuring this item will drive the emulator into the reset state.

Enable NMI input from target system?

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

yes The emulator will respond to the NMI request from the target system.

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

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

Note

Configuring this item will drive the emulator into the reset state.

Enable reset input from target system?

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

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

yes The emulator will respond to /RES and /STBY inputs during foreground operation.

no The emulator will not respond to /RES and /STBY inputs from the target system.

Note

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

Note

Configuring this item will drive the emulator into the reset state.

Drive reset output to target system?

This configuration allows you to specify whether or not the emulator will drive /RESO signal to the target system during reset by the Watchdog timer.

yes Specifies that the emulator will an active level on the /RESO signal to the target system during reset by the Watchdog timer.

no Specifies that the emulator will not drive the /RESO signal to the target system.

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 in this configuration.

Drive background cycles to the target system?

This configuration allows you specify whether or not the emulator will drive the target system bus on background cycles.

no Only emulation processor's address cycles are driven to the target system during background monitor.

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

Note

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

Note

If you specify that the emulator will not drive background cycles to the target system, the emulator can't respond to /WAIT signal during background monitor.

Note

This question does not appear when you select mode_6 or mode_7.

Note

Configuring this item will drive the emulator into the reset state.

Reset value for stack pointer?

This question allows you to specify the value of stack pointer (SP) .

The address specified in response to this question must be a 32-bit hexadecimal even address outside internal I/O register area. Default value of stack pointer is ffffff10 hex.

Target memory access size?

This configuration allow you to specify the types of cycles that the emulation monitor uses when accessing target system memory.

any	Access size is depends upon a display/modify target memory command option. If option "long" is specified, access size is will be set to "words". Other target memory commands such as "load" and "store" will use as access size of "bytes".
bytes	Specify that the emulator will access target system memory by byte access.

words Specify that the emulator will access target system memory by word access.

Note

When the access size is **words**, 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 **any** or **bytes** in this configuration

Debug/Trace Configuration

The debug/trace configuration questions allows you to specify breaks on writes to ROM, and specify that the analyzer trace foreground/background execution, and bus release cycles. To access the trace/debug configuration questions, you must answer "yes" to the following question.

Modify debug/trace options?

Break processor on write to ROM?

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

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

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

Note

The **wrrom** trace command status options allow you to use "write to ROM" cycles as trigger and storage qualifiers. For example, you could use the following command to trace about a write to ROM:
trace about status wrrom <RETURN>

Note

If the 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.

Trace background or foreground operation?

This question allows you to specify whether the analyzer traces only foreground emulation processor cycles, only background cycles, or both foreground and background cycles. When background cycles are stored in the trace, all mnemonic lines are tagged as background cycles.

foreground	Specifies that the analyzer traces only foreground cycles. This option is specified by the default emulator configuration.
background	Specifies that the analyzer traces only background cycles. (This is rarely a useful setting.)
both	Specifies that the analyzer traces both foreground and background cycles.

Trace on-chip DMAC cycles?

This question allows you to specify whether or not the emulator traces internal DMAC cycles.

yes	Specifies that the analyzer traces internal DMAC cycles.
no	Specifies that the analyzer dose not trace internal DMAC cycles.

Note 

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

Trace refresh cycles?

This question allows you to specify whether or not the emulator traces refresh cycles.

yes Specifies that the analyzer traces refresh cycles.

no Specifies that the analyzer dose not trace refresh cycles.

Note 

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

Note 

This question does not appear when you select mode_6 or mode_7.

Simulated I/O Configuration

The simulated I/O feature and configuration options are described in the *Simulated I/O reference* manual.

Interactive Measurement Configuration

The interactive measurement configuration questions are described in the chapter on coordinated measurements in the *Softkey Interface Reference* manual. Examples of coordinated measurements that can be performed between the emulator and the emulation analyzer are found in the "Using the Emulator" chapter.

External Analyzer Configuration

The external analyzer configuration options are described in the *Analyzer Softkey Interface User's Guide.*

Saving a Configuration

The last configuration question allows you to save the previous configuration specifications in a file which can be loaded back into the emulator at a later time.

Configuration file name? <FILE>

The name of the last configuration file is shown, or no filename is shown if you are modifying the default emulator configuration.

If you press <RETURN> without specifying a filename, the configuration is saved to a temporary file. This file is deleted when you exit the Softkey Interface with the "end release_system" command.

When you specify a filename, the configuration will be saved to a file specified with extensions of ".EA"

Ending out of emulation (with the "end" command) saves the current configuration, including the name of the most recently loaded configuration file, into a "continue" file. The continue file is not normally accessed.

Loading a Configuration

Configuration files which have been previously saved may be loaded with the following Softkey Interface command.

```
load configuration <FILE> <RETURN>
```

This feature is especially useful after you have exited the Softkey Interface with the "end release_system" command; it saves you from having to modify the default configuration and answer all the questions again.

To reload the current configuration, you can enter the following command.

load configuration <RETURN>

5

Using the Emulator

Introduction

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

This chapter discusses:

- Features available via "pod_command".
- Register classes and names.
- Debugging C Programs
- Accessing target system devices using E clock synchronous instruction.

This chapter shows you how to:

- Store the contents of memory into absolute files.
- Make coordinated measurements.
- Use a command file.

Features Available via Pod Commands

Several emulation features available in the Terminal Interface but not in the Softkey Interface may be accessed via the following emulation commands.

```
display pod_command <RETURN>
pod_command '<Terminal Interface command>'
<RETURN>
```

Some of the most notable Terminal Interface features not available in the softkey Interface are:

- Copying memory.
- Searching memory for strings or numeric expressions.
- Performing coverage analysis.

Refer to your Terminal Interface documentation for information on how to perform these tasks.

Note

Be careful when using the "pod_command". The Softkey Interface, and the configuration files in particular, assume that the configuration of the HP 64700 pod is NOT changed except by the Softkey Interface. Be aware that what you see in "modify configuration" will NOT reflect the HP 64700 pod's configuration if you change the pod's configuration with this command. Also, commands which affect the communications channel should NOT be used at all. Other commands may confuse the protocol depending upon how they are used. The following commands are not recommended for use with "pod_command":

stty, po, xp - Do not use, will change channel operation and hang.
echo, mac -Usage may confuse the protocol in use on the channel.
wait -Do not use, will tie up the pod, blocking access.
init, pv -Will reset pod and force end release_system.
t - Do not use, will confuse trace status polling and unload.

Using a Command File

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

To create such a command file, type "**log**" and press TAB key. You will see a command line "**log_commands**" appears in the command field. Next, select "**to**" in the softkey label, and enter the command file name "sample.cmd". This set up a file to record all commands you execute. The commands will be logged to the file sample.cmd in the current directory. You can use this file as a command file to execute these commands automatically.

Suppose that your configuration file and program are named "cmd_rds". To load configuration:

```
load configuration cmd_rds <RETURN>
```

To load the program into memory:

```
load cmd_rds <RETURN>
```

To display memory 1000 hex through 1020 hex in mnemonic format:

```
display memory 1000h thru 1020h mnemonic
```

Now, to disable logging, type "**log**" and press TAB key, select "**off**", and press **Enter**. The command file you created looks like this:

```
load configuration cmd_rds
load cmd_rds
display memory 1000h thru 1020h mnemonic
```

If you would like to modify the command file, you can use any text editor on your host computer.

To execute this command file, type "sample.cmd", and press **Enter**.

Debugging C Programs

Softkey Interface has following functions to debug C programs.

- Including C source lines in memory mnemonic display
- Including C source lines in trace listing
- Stepping C sources

The following section describes such features.

Displaying Memory with C Sources

You can display memory in mnemonic format with C source lines. For example, to display memory in mnemonic format from address main with source lines, enter the following commands.

```
display memory main mnemonic <RETURN>
set source on <RETURN>
```

You can display source lines highlighted with the following command.

```
set source on inverse_video on <RETURN>
```

To display only source lines, use the following command.

```
set source only <RETURN>
```

Specifying Address with Line Numbers

You can specify addresses with line numbers of C source program. For example, to set a breakpoint to line 20 of "main.c" program, enter the following command.

```
modify software_breakpoints set main.c: line
20 <RETURN>
```

Displaying Trace with C Sources

You can include C source information in trace listing. You can use the same command as the case of memory display. For example, to display trace listing with source lines highlighted, enter the following command.

```
display trace <RETURN>
set source on inverse_video on <RETURN>
```

Stepping C Sources

You can direct the emulator to execute a line or a number of lines at a time. For example, to step one line from address main, enter the following command.

```
step source from main <RETURN>
```

To step 1 line from the current line, enter the following command.

```
step source <RETURN>
```

You can specify the number of lines to be executed. To step 5 lines from the current line, enter the following command.

```
step 5 source <RETURN>
```

Storing Memory Contents to an Absolute File

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

```
store memory 1000h thru 1042h to absfile
<RETURN>
```

The command above causes the contents of memory locations 1000 hex through 1042 hex to be stored in the absolute file "absfile.X". Notice that the ".X" extension is appended to the specified filename.

Coordinated Measurements

For information on coordinated measurements and how to use them, refer to the "Coordinated Measurements" chapter in the *Softkey Interface Reference* manual.

Register Classes and Names

Summary

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

<REG_CLASS>

<REG_NAME> Description

BASIC (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 *
ASTCR	2/3 state area control register
WCR	Wait control register
WCER	Wait controller enable register
BRCR	Bus release control register *

* Except 3030, 3031, 3032

RFSHC (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
DTCR0A	Data transfer control register 0A
MAR0B	Memory address register 0B
ETCR0B	Transfer count register 0B
IOAR0B	I/O address register 0B
DTCR 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
DTCR1A	Data transfer control register 1A
MAR1B	Memory address register 1B
ETCR1B	Transfer count register 1B
IOAR1B	I/O address register 1B
DTCR 1B	Data transfer control register 1B

DMAC2 (DMA controller 2)

The following registers don't exist in 3002, 3030, 3031,3032, and 3042.

MAR2A	Memory address register 2A
ETCR2A	Transfer count register 2A
IOAR2A	I/O address register 2A
DTCR2A	Data transfer control register 2A
MAR2B	Memory address register 2B
ETCR2B	Transfer count register 2B
IOAR2B	I/O address register 2B
DTCR 2B	Data transfer control register 2B

DMAC3 (DMA controller 3)

The following registers don't exist in 3002, 3030, 3031, 3032 and 3042.

MAR3A	Memory address register 3A
ETCR3A	Transfer count register 3A
IOAR3A	I/O address register 3A
DTCR3A	Data transfer control register 3A
MAR3B	Memory address register 3B
ETCR3B	Transfer count register 3B
IOAR3B	I/O address register 3B
DTCR 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)
P ADDR	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
TFCR	Timer function control register
TOER	Timer output master control register
TOCR	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: 0xxffa5h)
NDRA0	Next data register A (address: 0xxffa7h)
NDERB	Next data enable register B
NDRB	Next data register B (address: 0xxffa4h)
NDRB2	Next data register B (address: 0xxffa6h)

WDT (Watch dog timer)

WDTCSR	Timer control/status register
WDTCNT	Timer counter
RSTCSR	Reset control/status register

SCI0 (Serial communication interface 0)

RDR0	Receive data 0 (Read Only)
TDR0	Transmit data register 0
SMR0	Serial mode register 0
SCR0	Serial control register 0
SSR0	Serial status register 0
BRR0	Bit rate register 0

SCI1 (Serial communication interface 1)

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

RDR1	Receive data 1 (Read Only)
TDR1	Transmit data register 1
SMR1	Serial mode register 1
SCR1	Serial control register 1
SSR1	Serial status register 1
BRR1	Bit rate register 1

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 don't 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

OTHER

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 R0 H
R0H	Register R0 L
R1H	Register R1 H
R1L	Register R1 L
R2H	Register R2 H
R2L	Register R2 L
R3H	Register R3 H
R3L	Register R3 L
R4 H	Register R4 H
R4L	Register R4 L
R5H	Register R5 H
R5L	Register R5 L
R6H	Register R6 H
R6L	Register R6 L
R7H	Register R7 H
R7L	Register R7 L

Notes

Index

Notes