
HP 64791/2

70208H/70216H Emulator Softkey Interface

User's Guide



HP Part No. 64791-97011

Printed in U.S.A.

July 1994

Edition 4

Notice

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

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

© Copyright 1991, 1993, 1994, Hewlett-Packard Company.

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

UNIX is a registered trademark of AT&T.

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

V40™ and V50™ are trademarks of NEC Electronics Inc.

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

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013.

Hewlett-Packard Company, 3000 Hanover Street, Palo Alto, CA 94304 U.S.A. Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

Printing History

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

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

Edition 1	64791-97002, August 1991
Edition 2	64791-97005, November 1991
Edition 3	64791-97008, December 1993
Edition 4	64791-97011, July 1994

Using this Manual

This manual will show you how to use the following emulators with the Softkey Interface.:

- HP 64791A 70208 emulator
- HP 64792A 70216 emulator
- HP 64791B 70208H emulator
- HP 64792B 70216H emulator

For the most part, these emulators all operate the same way. Differences between the emulators are described where they exist. These 70208, 70208H, 70216 and 70216H emulators will be referred to as the "70216 emulator" in this manual where they are alike. In the specific instances where 70208, 70208H and 70216H emulator differs from the 70216 emulator, it will be referred as the "70208 emulator", "70208H emulator" and "70216H emulator".

This manual will:

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

This manual will not:

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

Organization

- Chapter 1** **Introduction to the 70216 Emulator.** This chapter briefly introduces you to the concept of emulation and lists the basic features of the 70216 emulator.
- 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 install the emulator probe into a target system and how to use "in-circuit" emulation features.
- Chapter 4** **Configuring the Emulator.** This chapter shows you how to: restrict the emulator to real-time execution, select a target system clock source, allow the target system to insert wait states, and select foreground or background monitor.
- Chapter 5** **Using the Emulator.** This chapter describes emulation topics which are not covered in the "Getting Started" chapter.
- Appendix A** **Using the Foreground Monitor.** This appendix describes the advantages and disadvantages of foreground and background monitors and how to use foreground monitor.

Contents

1	Introduction to the 70216 Emulator	
	Introduction	1-1
	Purpose of the Emulator	1-1
	Features of the 70216 Emulator	1-3
	Supported Microprocessors	1-3
	Clock Speeds	1-3
	Emulation memory	1-4
	Analysis	1-4
	Registers	1-4
	Single-Step	1-4
	Breakpoints	1-5
	Reset Support	1-5
	Configurable Target System Interface	1-5
	Foreground or Background Emulation Monitor	1-5
	Real-Time Operation	1-6
	Easy Products Upgrades	1-6
	Limitations, Restrictions	1-7
	DMA Support	1-7
	TC bit of DMA Status Register	1-7
	User Interrupts	1-7
	Interrupts While Executing Step Command	1-7
	Evaluation chip	1-7
2	Getting Started	
	Introduction	2-1
	Before You Begin	2-2
	Prerequisites	2-2
	A Look at the Sample Program	2-3
	Entering the Softkey Interface	2-5
	From the "pmon" User Interface	2-5
	From the HP-UX Shell	2-6
	Configure the Emulator for Examples	2-8

On-Line Help	2-9
Softkey Driven Help	2-9
Pod Command Help	2-10
Loading Absolute Files	2-11
Displaying Symbols	2-12
Global	2-12
Local	2-13
Source Lines	2-14
Displaying Memory in Mnemonic Format	2-15
Symbols in the Display	2-16
Source Lines in the Display	2-17
Using Software Breakpoints	2-18
Enabling/Disabling Software Breakpoints	2-19
Setting a Software Breakpoint	2-19
Running the Program	2-20
From Transfer Address	2-20
From Reset	2-21
Stepping Through the Program	2-22
Modifying Memory	2-23
Breaking into the Monitor	2-24
Displaying Registers	2-25
Stepping Through the Program	2-26
Using the Analyzer	2-28
Specifying a Simple Trigger	2-28
Displaying the Trace	2-29
Displaying Trace with Time Count Absolute	2-31
Displaying Trace with Compress Mode	2-32
Changing the Trace Depth	2-33
Emulator Analysis Status Qualifiers	2-33
Resetting the Emulator	2-34
Exiting the Softkey Interface	2-34
End Release System	2-34
Ending to Continue Later	2-34
Ending Locked from All Windows	2-35
Selecting the Measurement System Display or Another Module	2-35

3	'In-Circuit' Emulation	
	Introduction	3-1
	Prerequisites	3-1
	Installing the Target System Probe	3-2
	Auxiliary Output Lines	3-3
	Installing into a PLCC Type Socket	3-5
	Installing into a PGA Type Socket	3-6
	In-Circuit Configuration Options	3-8
	Running the Emulator from Target Reset	3-9
	Target System Interface	3-10
4	Configuring the Emulator	
	Introduction	4-1
	General Emulator Configuration	4-4
	Micro-processor Clock Source?	4-4
	Enter Monitor After Configuration?	4-4
	Restrict to Real-Time Runs?	4-6
	Memory Configuration	4-7
	Monitor Type?	4-7
	Mapping Memory	4-10
	Emulator Pod Configuration	4-12
	Respond to DMARQ0-3 from target system in background?	4-12
	Use FPP on target system?	4-12
	Memory display mnemonic? (70208/70208H Emulator)	4-13
	Memory display mnemonic? (70216/70216H Emulator)	4-13
	Dis-assembler mode?	4-14
	Segment algorithm ?	4-15
	Reset value for the stack pointer?	4-16
	Respond to RESET from target system?	4-16
	Respond to NMI from target system?	4-17
	Respond to READY from target system for accessing to emulation memory?	4-18
	Respond to HLDRQ from target system?	4-18
	Target memory access size?	4-19
	Debug/Trace Configuration	4-20
	Break Processor on Write to ROM?	4-20
	Trace Background or Foreground Operation?	4-21
	Trace Internal DMA cycles?	4-21
	Trace bus cycles in HOLD state ?	4-21
	Trace refresh cycles?	4-22
	Simulated I/O Configuration	4-23

External Analyzer Configuration	4-23
Interactive Measurement Configuration	4-23
Saving a Configuration	4-23
Loading a Configuration	4-24

5 Using the Emulator

Introduction	5-1
Register Names and Classes	5-2
BASIC(*) class	5-2
SIO class (70208/70216 Emulator)	5-2
SIO class (70208H/70216H Emulator)	5-3
ICU class	5-4
TCU class	5-4
SCU class	5-5
DMA71 class	5-5
DMA37 class (70208H/70216H Emulator only)	5-6
Features Available via Pod Commands	5-7
Storing Memory Contents to an Absolute File	5-8
Coordinated Measurements	5-8

A Using the Foreground Monitor

Introduction	A-1
Comparison of Foreground and Background Monitors	A-1
Background Monitors	A-2
Foreground Monitors	A-2
An Example Using the Foreground Monitor	A-3
Modify EQU Statement	A-3
Assemble and Link the Monitor	A-4
Modifying the Emulator Configuration	A-4
Load the Program Code	A-6
Single Step and Foreground Monitors	A-7
Limitations of Foreground Monitors	A-8
Synchronized MeasurementsCMB	A-8

Illustrations

- Figure 1-1. HP 64792 Emulator for uPD70216 1-2
- Figure 2-1. The "cmd_rds.c" Sample Program 2-4
- Figure 2-2. Softkey Interface Display 2-7
- Figure 3-1. Auxiliary Output Lines (70216 Emulator) 3-3
- Figure 3-2. Installing into a PLCC type socket 3-5
- Figure 3-3. Installing into a PGA type socket 3-7

Notes

6-Contents



Introduction to the 70216 Emulator

Introduction

The topics in this chapter include:

- Purpose of the emulator
- Features of the emulator
- Limitations and Restrictions of the emulator

Purpose of the Emulator

The 70216 emulator is designed to replace the 70216 microprocessor in your target system to help you debug/integrate target system software and hardware. The emulator performs just like the processor which it replaces, but at the same time, it gives you information about the bus cycle operation of the processor. The emulator gives you control over target system execution and allows you to view or modify the contents of processor registers, target system memory, and I/O resources.



RS-232/RS-422
Connection

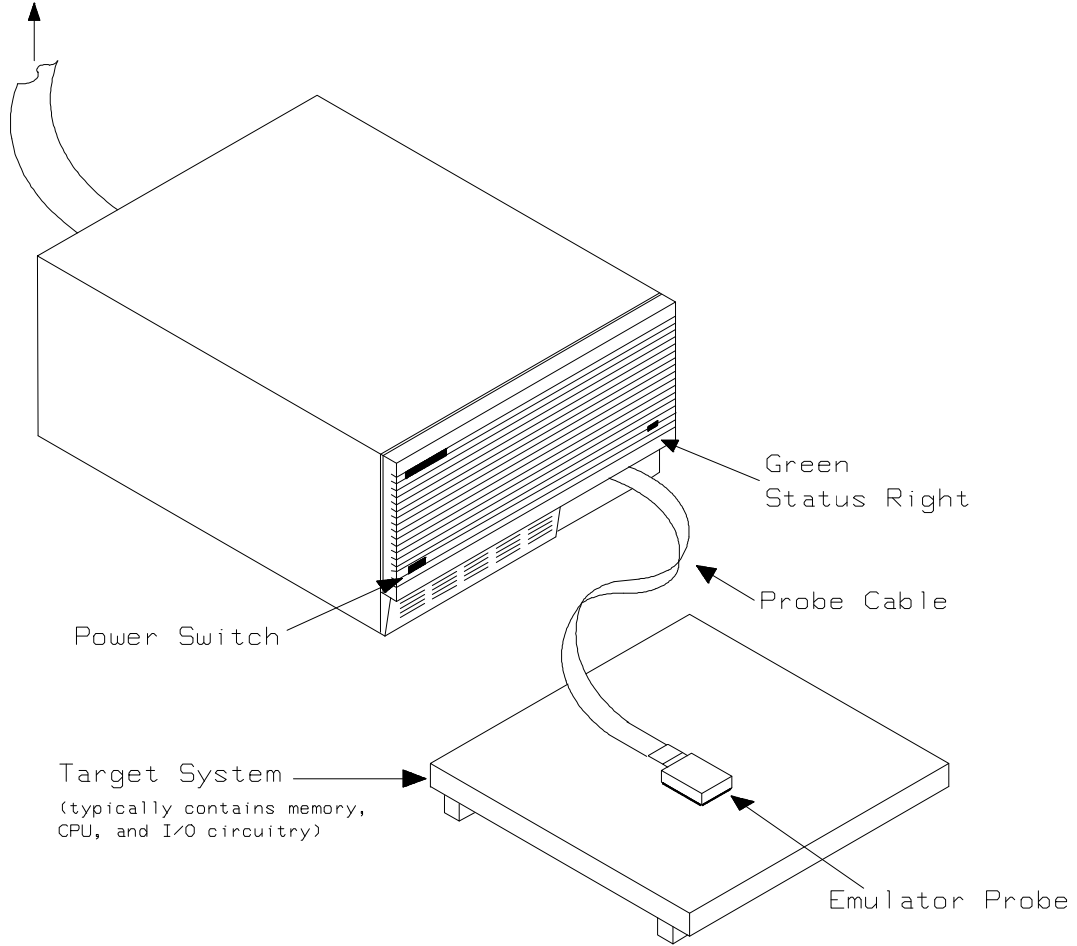


Figure 1-1. HP 64792 Emulator for uPD70216

1-2 Introduction

Features of the 70216 Emulator

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

Supported Microprocessors

The HP 64791/2 emulator supports the following packages of microprocessor.

Model No.	Microprocessor	Package
HP 64791A	uPD70208	68-pin PLCC 68-pin PGA
HP 64792A	uPD70216	68-pin PLCC 68-pin PGA
HP 64791B	uPD70208H	68-pin PLCC 68-pin PGA
HP 64792B	uPD70216H	68-pin PLCC 68-pin PGA

The HP 64791/2 emulator probe has a 68-pin PLCC connector. When you use 68-pin PGA type microprocessor, you must use with PLCC to PGA adapter; refer to the "In-Circuit Emulation Topics" chapter in this manual.

Clock Speeds

The 70208 and 70216 emulator runs with an internal clock speed of 8MHz (system clock), or with target system clocks from 2 to 10 MHz.

The 70208H and 70216H emulator runs with an internal clock speed of 16 MHz (system clock) or with target system clocks from 1 to 16 MHz.



Emulation memory

The HP 70216 emulator is used with one of the following Emulation Memory Cards.

- HP 64726 128K byte Emulation Memory Card
- HP 64727 512K byte Emulation Memory Card
- HP 64728 1M byte Emulation Memory Card
- HP 64729 2M byte Emulation Memory Card

When you use the HP 64729, You can only use 1M byte for emulation memory.

You can define up to 16 memory ranges (at 128 byte boundaries and at least 128 byte in length). You can characterize memory ranges as emulation RAM, emulation ROM, target system RAM, target system ROM, or as guarded memory. The emulator generates an error message when accesses are made to guarded memory locations. You can also configure the emulator so that writes to memory defined as ROM cause emulator execution to break out of target program execution.

Analysis

The HP 70216 emulator is used with one of the following analyzers which allows you to trace code execution and processor activity.

- HP 64703 64-channel Emulation Bus Analyzer and 16-channel State/Timing Analyzer
- HP 64704 80-channel Emulation Bus Analyzer
- HP 64794A/C/D Deep Emulation Bus Analyzer

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

Registers

You can display or modify the 70216 internal register contents.

Single-Step

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



Breakpoints

You can set up the emulator/analyzer interaction so that when the analyzer finds a specific state, emulator execution will break to the background monitor.

You can also define software breakpoints in your program. The emulator uses the BRK 3 instruction(CC hex) as software breakpoint interrupt instruction. When you define a software breakpoint, the emulator places the breakpoint interrupt instruction (CC hex) at the specified address; after the breakpoint interrupt instruction causes emulator execution to break out of your program, the emulator replaces the original opcode.

Reset Support

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

Configurable Target System Interface

You can configure the emulator so that it honors target system wait requests when accessing emulation memory. You can configure the emulator so that it presents cycles to, or hides cycles from, the target system when executing in background.

Foreground or Background Emulation Monitor

The emulation monitor is a program that is executed by the emulation processor. It allows the emulation controller to access target system resources. For example, when you display target system memory, it is the monitor program that executes 70216 instructions which read the target memory locations and send their contents to the emulation controller.

The monitor program can execute in *foreground*, the mode in which the emulator operates as would the target processor. The foreground monitor occupies processor address space and executes as if it were part of the target program.

The monitor program can also execute in *background*, the emulator mode in which foreground operation is suspended so that emulation processor can be used to access target system resources. The background monitor does not occupy any processor address space.



Real-Time Operation

Real-time operation signifies continuous execution of your program without interference from the emulator. (Such interference occurs when the emulator temporarily breaks to the monitor so that it can access register contents or target system memory or I/O.)

You can restrict the emulator to real-time execution. When the emulator is executing your program under the real-time restriction, commands which display/modify registers, display/modify target system memory or I/O are not allowed.

Easy Products Upgrades

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

Limitations, Restrictions



DMA Support	Direct memory access to emulation memory by external DMA controller is not permitted.
TC bit of DMA Status Register	While using the uPD71071 or the uPD71037 DMA mode on the 70208H emulator, or using the uPD71037 DMA mode on the 70216H emulator, when the emulator read the other than DST register, the TC bit of the DST is reset. If you know the DMA Status, you have to use the count register in the place of the TC bit.
User Interrupts	If you use the background monitor, NMI and INTP1-7 from the target system are suspended until the emulator goes into foreground operation.
Interrupts While Executing Step Command	While executing user program code in stepping in the foreground monitor, interrupts are accepted if they are enabled in the foreground monitor program. When using the background monitor the emulator will fail to step, if the interrupts are acknowledged before stepping user program code.
Evaluation chip	Hewlett-Packard makes no warranty of the problem caused by the 70208/70208H/70216/70216H Evaluation chip in the emulator.



Notes

1-8 Introduction

Getting Started

Introduction

This chapter will lead you through a basic, step by step tutorial that shows how to use the HP 64792 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 examples.

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 HP 64700 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 70216 emulator.

A Look at the Sample Program

The sample program used in this chapter is shown in Figure 2-1. The program continuously reads values from **Cmd_Input**; when a value other than NULL is found, the program calls the **Write_Msg** function to copy a string to the **Msg_Dest** array.

The sample program and the associated output files, including the HP format absolute files, have been shipped with the Softkey Interface; copy these files to the current directory with the following command:

```
$ cp /usr/hp64000/demo/emul/hp64791/* .  
  (70208,70208H)  
$ cp /usr/hp64000/demo/emul/hp64792/* .  
  (70216,70216H)
```

The file *cmd_rds.X* contains the absolute code of the program. The file *cmd_rds.L* contains the list of global symbols. The files *cmd_rds.A* contains the list of local symbols for the respective files.

The user interface provides source line referencing if line information is present in the local symbol file.

```

1  volatile char Cmd_Input;
2  char Msg_Dest[0x20];
3
4  void Write_Msg (const char *s)
5  {
6      char *Dest_Ptr;
7
8      Dest_Ptr = Msg_Dest;
9      while (*s != '\0')
10     {
11         *Dest_Ptr = *s;
12         Dest_Ptr++;
13         s++;
14     }
15 }
16
17 main ()
18 {
19     static char Msg_A[] = "Command A Entered           ";
20     static char Msg_B[] = "Entered B Command         ";
21     static char Msg_I[] = "Invalid Command          ";
22     char c;
23
24     for (;;)
25     {
26         Cmd_Input = '\0';
27         while ((c = Cmd_Input) == '\0');
28         switch (c) {
29             case 'A' :
30                 Write_Msg (Msg_A);
31                 break;
32             case 'B' :
33                 Write_Msg (Msg_B);
34                 break;
35             default :
36                 Write_Msg (Msg_I);
37                 break;
38         }
39     }
40 }

```

Figure 2-1. The "cmd_rds.c" Sample Program

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 through the **pmon** User Interface Software or from the HP-UX shell.

- If you have used previous HP 64000-UX emulators (for example, HP 64200 Series), you may be more familiar with the **pmon**, **msinit**, and **msconfig** method of entering the emulation interface.
- If you wish to run the Softkey Interface in multiple windows, you must enter from the HP-UX shell using the **emul700** command. Refer to the *Softkey Interface Reference* manual for more information on running in multiple windows.

From the "pmon" User Interface

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

```
$ pmon <RETURN>
```

If you have not already created a measurement system for the 70216 emulator, you can do so with the following commands. First you must initialize the measurement system with the following command.

```
MEAS_SYS msinit <RETURN>
```

After the measurement system has been initialized, enter the configuration interface with the following command.

```
msconfig <RETURN>
```

To define a measurement system for the 70216 emulator, enter:

```
make_sys emv50 <RETURN>
```

Now, to add the emulator to the measurement system, enter:

```
add <module_number> naming_it n70216 <RETURN>
```

Enter the following command to exit the measurement system configuration interface.

```
end <RETURN>
```

If the measurement system and emulation module are named "emv50" and "n70216" as shown above, you can enter the emulation system with the following command:

```
emv50 default n70216 <RETURN>
```

If this command is successful, you will see a display similar to figure 2-2. 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 **pmon** User Interface. Error messages are described in the *Softkey Interface Reference* manual.

For more information on creating measurements systems, refer to the *Softkey Interface Reference* manual.

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`).

For example, the emulator name in the device table entry shown below is "v40" for n70208, "v40h" for n70208h, "v50" for n70216 and "v50h" for n70216h.

#	logical name (14 chars)	processor type	physical device	xpar mode	baud rate	parity	flow	stop bits	char size
#				OFF		NONE	XON RTS	2	8
v40		n70208	/dev/emcom23	OFF	230400	NONE	RTS	2	8
v40h		n70208h	/dev/emcom23	OFF	230400	NONE	RTS	2	8
v50		n70216	/dev/emcom23	OFF	230400	NONE	RTS	2	8
v50h		n70216h	/dev/emcom23	OFF	230400	NONE	RTS	2	8

```
HPB3066-19309 A.05.20 11May93
70216/70216H EMULATION SERIES 64700

A Hewlett-Packard Software Product
Copyright Hewlett-Packard Co. 1992

All Rights Reserved. Reproduction, adaptation, or translation without prior
written permission is prohibited, except as allowed under copyright laws.

RESTRICTED RIGHTS LEGEND

Use , duplication , or disclosure by the Government is subject to
restrictions as set forth in subparagraph (c) (1) (II) of the Rights
in Technical Data and Computer Software clause at DFARS 52.227-7013.
HEWLETT-PACKARD Company , 3000 Hanover St. , Palo Alto, CA 94304-1181

STATUS:  Loaded configuration file_____...R....

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

Figure 2-2. Softkey Interface Display

If this command is successful, you will see a display similar to figure 2-2. 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.

Configure the Emulator for Examples

To do operations described in this chapter (loading absolute program into emulation memory, displaying memory contents, etc), you need to configure the emulator as below. For detailed description of each configuration options (question), refer to the "*Configuring the Emulator*" chapter.

To get into the configure session of the emulator, enter the following command.

```
modify configuration <RETURN>
```

The answer to series of questions as below.

```
Micro-processor clock source? internal <RETURN>  
Enter monitor after configuration? yes <RETURN>  
Restrict to real-time runs? no <RETURN>  
Modify memory configuration? yes <RETURN>  
Monitor type? background <RETURN>
```

Now you should be facing memory mapping screen. Three mapper terms must be specified for the sample program.

```
0h thru 0fffh emulation ram <RETURN>  
10000h thru 1ffffh emulation ram <RETURN>  
80000h thru 80ffffh emulation rom <RETURN>  
end <RETURN>  
Modify emulator pod configuration? no <RETURN>  
Modify debug/trace options? no <RETURN>  
Modify simulated I/O configuration? no <RETURN>  
Modify external analyzer configuration? no <RETURN>  
Modify interactive measurement specification? no <RETURN>  
Configuration file name? cmd_rds <RETURN>
```

If you wish to save the configuration specified above, answer this question as shown.

Now you are ready to go ahead. Above configuration is used throughout this chapter.

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---
?
help                displays the possible help files
                    displays the possible help files
!
!<shell command>  fork a shell (specified by shell variable SH)
                    fork a shell and execute a shell command
pwd
cd <directory>    print the working directory
                    change the working directory
pws
cws <SYMB>        print the default symbol scope
                    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
                    of 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 cf' <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).

Note



If you want to use the Terminal Interface command by entering from keyboard directly, you can do it after entering the following command.

pod_command keyboard

Pod Commands

```
Time          Command
bgdma        - enable/disable DMA cycle in background
clk          - select internal(32m/20m/116m) or external emulation clock
fpp          - enable/disable FPP support mode
tghld        - enable/disable target hold
mne          - select mnemonic for memory display
mode         - select assembler format
mon          - select foreground or background monitor
nmi          - enable/disable NMI signal from the target system
rad          - segment:offset translation method
rdy          - relationship between emulator and target ready (lk or unlk)
rrt          - enable/disable restrict to real time runs
rsp          - specify the stack after emulation reset
rst          - enable/disable RESET signal from the target system
tdma         - enable/disable DMA cycle trace
thold        - enable/disable hold acknowledge cycle trace
trfsh        - enable/disable refresh cycle trace

STATUS:      n70216--Running in monitor_____...R....
pod_command "help cf"

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

Loading Absolute Files

The "load" command allows you to load absolute files into emulation or target system memory. You can load absolute files in the following formats:

- HP absolute.
- Intel Object Module Format (OMF-86).

The "load" command has no special options for loading different absolute file formats; instead, the contents of the file are examined to determine the format being used.

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" or "user_mem".

To load the emulator sample program absolute file, enter the following command:

```
load cmd_rds <RETURN>
```

Displaying Symbols

If symbol information is present in the absolute file, it is loaded along with the absolute file (unless you use the "nosymbols" syntax). Both global symbols and symbols that are local to a program module can be displayed.

Global To display global symbols, enter the following command.

```
display global_symbols <RETURN>
```

Listed are: address ranges associated with a symbol, the segment that the symbol is associated with, and the offset of that symbol within the segment.

```
Global symbols in cmd_rds.X
Procedure symbols
Procedure name _____ Address range ___ Segment _____ Offset
Write_Msg                804C:000C - 004C   PROG                0000
_div_by_0_trap           8000:0088 - 00A2   PROG                0000
_exec_funcs              8046:003D - 005D   PROG                0000
_exit_msg                8000:02A8 - 02D5   PROG                0000
_fp_trap                 8000:013A - 0241   PROG                0000
atexit                   8046:0004 - 003C   PROG                0000
main                     804C:004D - 00EC   PROG                0041

Static symbols
Symbol name _____ Address range ___ Segment _____ Offset
Cmd_Input                1009:000C         DATA                0000
Err_Handler              803F:0058         PROG                0000
MM_CHECK_L               1000:0000         DATA                0000
MM_CHECK_X               1000:0000         DATA                0000
MONITOR_MESSAGE          1000:000A         DATA                0000

STATUS:  n70216--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 module in which the symbols are defined. For example:

display local_symbols_in cmd_rds.c: <RETURN>

As you can see, the procedure symbols and static symbols in "cmd_rds.c" are displayed.

If there is more than a screenful of information, you can use the up arrow, down arrow, <Next> or <Prev> keys to scroll the information up or down on the display.

```
Symbols in /usr/hp64000/demo/emul/hp64792/cmd_rds.c:
Procedure symbols
Procedure name _____ Address range __ Segment _____ Offset
Write_Msg                804C:000C - 004C   PROG          0000
main                     804C:004D - 00EC   PROG          0041

Static symbols
Symbol name _____ Address range __ Segment _____ Offset
Cmd_Input                1009:000C          DATA         0000
Msg_A                    1009:002D          DATA         0021
Msg_B                    1009:004E          DATA         0042
Msg_Dest                 1009:000D          DATA         0001
Msg_I                    1009:006F          DATA         0063
_Cmd_Input               1009:000C          DATA         0000
_Msg_Dest                1009:000D          DATA         0001
_Write_Msg               804C:000C          PROG          0000
_main                    804C:004D          PROG          0041

STATUS:  cws: cmd_rds.c:_____...R....
display  local_symbols_in cmd_rds.c:

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

Source Lines

To display the address ranges associated with the program's source file, you must display the local symbols in the file. For example:

```
display local_symbols_in cmd_rds.c: <RETURN>
```

And scroll the information down on the display with up the arrow, or <Next>key.

```
Symbols in /usr/hp64000/demo/emul/hp64792/cmd_rds.c:
Symbol name _____ Address range __ Segment _____ Offset

Source reference symbols
Line range _____ Address range __ Segment _____ Offset
#1-#5          804C:000C - 0015  PROG          0000
#6-#8          804C:0016 - 001F  PROG          000A
#9-#9          804C:0020 - 0025  PROG          0014
#10-#11       804C:0026 - 0031  PROG          001A
#12-#12       804C:0032 - 0035  PROG          0026
#13-#13       804C:0036 - 0039  PROG          002A
#14-#14       804C:003A - 0048  PROG          002E
#15-#15       804C:0049 - 004C  PROG          003D
#16-#18       804C:004D - 0056  PROG          0041
#19-#24       804C:0057          PROG          004B
#25-#26       804C:0058 - 005D  PROG          004C
#27-#27       804C:005E - 0076  PROG          0052
#28-#28       804C:0077 - 0099  PROG          006B

STATUS:  n70216--Running in monitor_____...R....
display  local_symbols_in cmd_rds.c:

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 sample program,

```
display memory main mnemonic <RETURN>
```

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

```
Memory :mnemonic :file = /usr/hp64000/demo/emul/hp64792/cmd_rds.c:
  address  data
804C 004D  C8020000  PREPARE 0002,00
804C 0051  1E          PUSH DS0
804C 0052  B80910     MOV AW,1009
804C 0055  8ED890     MOV DS0,AW | NOP
804C 0058  C6060C0000 MOV 000C,00
804C 005D  90         NOP
804C 005E  EB05      BR SHORT 00065
804C 0060  90         NOP
804C 0061  90         NOP
804C 0062  90         NOP
804C 0063  90         NOP
804C 0064  90         NOP
804C 0065  A00C00    MOV AL,000C
804C 0068  90         NOP
804C 0069  90         NOP
804C 006A  90         NOP

STATUS:  n70216--Running in monitor_____...R....
display  memory main mnemonic

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

Symbols in the Display

The "set" command allows you to include symbols in mnemonic memory displays and in the trace displays. For example:

```
set symbols on <RETURN>
```

```
Memory :mnemonic :file = /usr/hp64000/demo/emul/hp64792/cmd_rds.c:
address label      data
804C 004D  PROG|_main  C8020000  PREPARE 0002,00
804C 0051                1E        PUSH DS0
804C 0052                B80910    MOV AW,1009
804C 0055                8ED890    MOV DS0,AW | NOP
804C 0058                C6060C0000 MOV 000C,00
804C 005D                90        NOP
804C 005E                EB05      BR SHORT PROG|main+00018
804C 0060                90        NOP
804C 0061                90        NOP
804C 0062                90        NOP
804C 0063                90        NOP
804C 0064                90        NOP
804C 0065                A00C00    MOV AL,000C
804C 0068                90        NOP
804C 0069                90        NOP
804C 006A                90        NOP

STATUS:  n70216--Running in monitor_____...R....
set symbols on

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

Source Lines in the Display

The "set" command also allows you to include source lines in mnemonic memory displays and in the trace displays. For example:

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

```
Memory :mnemonic :file = /usr/hp64000/demo/emul/hp64792/cmd_rds.c:
address label      data
 16
 17   main ()
 18   {
804C 004D   PROG|_main  C8020000   PREPARE 0002,00
804C 0051           1E           PUSH DS0
804C 0052           B80910      MOV AW,1009
804C 0055           8ED890      MOV DS0,AW | NOP
 25           {
 26           Cmd_Input = '\0';
804C 0058           C6060C0000  MOV 000C,00
804C 005D           90           NOP
 27           while ((c = Cmd_Input) == '\0');
804C 005E           EB05        BR SHORT PROG|main+00018
804C 0060           90           NOP
804C 0061           90           NOP
804C 0062           90           NOP

STATUS:  n70216--Running in monitor_____...R....
set source on

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

Using Software Breakpoints

Software breakpoints are handled by the the 70208/70216 single byte interrupt facility. When you define or enable a software breakpoint to a specified address, the emulator will replace the opcode with a BRK 3 instruction.

When the software breakpoints are enabled and the emulator detects the breakpoint interrupt instruction (CC hex), user program breaks to the monitor, and the original opcode will be replaced at the software breakpoint address.

Since the system controller knows the locations of the defined software breakpoints, it can determine whether the breakpoint interrupt instruction was generated by an enabled software breakpoint or by a single-byte interrupt instruction in your target system.

If the single-byte interrupt was generated by a software breakpoint, execution breaks to the monitor, and the breakpoint interrupt instruction (BRK 3) is replaced by the original opcode. A subsequent run or step command will execute from this address.

If the single-byte interrupt was generated by a BRK 3 instruction in the target system, execution still breaks to the monitor, and an "Undefined software breakpoint" message is displayed.

Caution



Software breakpoints should not be set, cleared, enabled, or disabled while the emulator is running user code. If any of these commands are entered while the emulator is running user code, and the emulator is executing code in the area where the breakpoint is being modified, program execution may be unreliable.

Note

Because software breakpoints are implemented by the replacing opcodes with the breakpoint interrupt instruction (CC hex), you can not define the software breakpoints in the target ROM.

However you can copy target ROM into the emulation memory which does allow you to use software breakpoints. Once target ROM is copied into the emulation memory, software breakpoints may be used normally at the addresses in these emulation memory locations. (see the "Target ROM Debug Topics" section of the "In-Circuit Emulation" chapter in the *Terminal Interface User's Guide* manual.)

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.

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>
```

Setting a Software Breakpoint

To set a software breakpoint at the address of global symbol "main" or (or source line 17), enter the following command.

```
modify software_breakpoints set main  
<RETURN>
```

or:

```
modify software_breakpoints set line 17  
<RETURN>
```

```

Memory :mnemonic :file = /usr/hp64000/demo/emul/hp64792/cmd_rds.c:
address label      data
16
17   main ()
18   {
*804C 004D   PROG|_main  CC          BRK 3
804C 004E           0200        ADD AL,(BW)(IX)
804C 0050           001EB809    ADD 09B8,BL
804C 0054           108ED890    ADDC (BP-6F28),CL
25           {
26           Cmd_Input = '\0';
804C 0058           C6060C0000 MOV 000C,00
804C 005D           90          NOP
27           while ((c = Cmd_Input) == '\0');
804C 005E           EB05        BR SHORT PROG|main+00018
804C 0060           90          NOP
804C 0061           90          NOP
804C 0062           90          NOP

STATUS:  n70216--Running in monitor.....R....
modify  software_breakpoints  set line 17

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

```

Notice that an asterisk (*) appears next to the breakpoint address. The asterisk shows that a software breakpoint is pending at that address.

Running the Program

The "run" command causes the emulator to execute the user program. 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). Enter:

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



```

Memory :mnemonic :file = /usr/hp64000/demo/emul/hp64792/cmd_rds.c:
address label      data
16
17   main ()
18   {
>804C 004D   PROG|_main  C8020000   PREPARE 0002,00
804C 0051           1E         PUSH DS0
804C 0052           B80910     MOV AW,1009
804C 0055           8ED890     MOV DS0,AW | NOP
25           {
26           Cmd_Input = '\0';
804C 0058           C6060C0000 MOV 000C,00
804C 005D           90         NOP
27           while ((c = Cmd_Input) == '\0');
804C 005E           EB05       BR SHORT PROG|main+00018
804C 0060           90         NOP
804C 0061           90         NOP
804C 0062           90         NOP

STATUS:  n70216--Running in monitor      Software break: 0804c:0004d____.R....
run from transfer_address

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

```

Notice the highlighted bar on the screen; it shows the current program counter.

Notice also that the asterisk is no longer next to the breakpoint address; this shows that the breakpoint has been hit and is no longer active.

From Reset The "run from reset" command specifies that the emulator begin executing from reset vector as actual microprocessor does.
 (See "Running the Emulator From Target Reset" section in the "In-Circuit Emulation" chapter).

Stepping Through the Program

The step command allows you to step through program execution an instruction or a number of instructions at a time. You can step through the instructions associated with high-level program source lines. 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 source <RETURN>

```
Memory :mnemonic :file = /usr/hp64000/demo/emul/hp64792/cmd_rds.c:
address label      data
 16
 17   main ()
 18   {
804C 004D   PROG|_main  C8020000   PREPARE 0002,00
804C 0051           1E         PUSH DS0
804C 0052           B80910     MOV AW,1009
804C 0055           8ED890     MOV DS0,AW | NOP
 25           {
 26           Cmd_Input = '\0';
804C 0058           C6060C0000 MOV 000C,00
804C 005D           90         NOP
 27           while ((c = Cmd_Input) == '\0');
804C 005E           EB05     BR SHORT PROG|main+00018
804C 0060           90         NOP
804C 0061           90         NOP
804C 0062           90         NOP

STATUS:  n70216--Stepping complete_____...R....
step source

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

Notice that the highlighted bar (the current program counter) moves to the instructions associated with the next source line.

Enter the "step source" command again by pressing:

<RETURN>, <RETURN>

Notice that the emulator continues to step through the program and that the message "assembly steps taken: XXX" appears on the status line.

This happens because the "while" test remains true, and the emulator never completes the execution of the assembly instructions associated with that source line. To stop the "step source" command, enter:

<CTRL>-c

Continue user program execution with the "run" command.

```
run <RETURN>
```

Modifying Memory

The sample program is a simple command interpreter. Commands are sent to the sample program through a "char" sized memory location, global variable **Cmd_Input**. You can use the modify memory feature to send a command to the sample program.

For example, to enter the command "A" (41H), use the following command:

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

or:

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

To verify that the program correctly copied the message "Command A Entered" to the **Msg_Dest** array, display the contents of the array with the following command:

```
display data Msg_Dest thru +1fh char  
<RETURN>
```

Enter the following commands to verify that the program works for the other possible command inputs.

```
modify memory Cmd_Input strings to 'B'  
<RETURN>
```

```
modify memory Cmd_Input strings to 'C'  
<RETURN>
```

Notice that the display is updated when the memory contents change due (indirectly) to the "modify memory" command.

```
Data :update
  address label      type      data
1009 000D  DA|_Msg_Dest char[]  Command A Entered
```

```
STATUS:  n70216--Running user program_____...R....
display  data Msg_Dest thru +1fh char
```

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

Breaking into the Monitor

The "break" command causes emulator execution to break 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>

Displaying Registers

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

display registers <RETURN>

Refer to "Register Names and Classes" section in chapter 5.

```
Registers
Next_PC 804C:0068H
PC 0068 SP 7EEA IX 0000 IY 004D BP 7EEE PSW F246
PS 804C SS 1112 DS0 1009 DS1 1009 [MD ... V DIR IE BRK S Z .AC . P . C]
AW 1000 BW 0000 CW 0001 DW 1009 1 111 . . 1 . . 1 . . . 1 1 .

STATUS: n70216--Running in monitor.....R....
display registers

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

Stepping Through the Program

You can step through sample program instructions while displaying registers. For example, entering several step commands will give you a display similar to the following.

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



Note



There are a few cases in which the emulator can not step. Step command is not accepted between each of the following instructions and the next instruction.

- 1) Manipulation instructions for sreg:
MOV sreg,reg16; MOV sreg,mem16; POP sreg.
 - 2) Prefix instructions:
PS:, SS:, DS0:, DS1:,
REPC, REPNC, REP, REPE, REPZ, REPNE, REPNZ.
 - 3) EI, RETI, DI, BUSLOCK.
-

Registers

```
Next_PC 804C:006BH
PC 006B SP 7EEA IX 0000 IY 004D BP 7EEE PSW F246
PS 804C SS 1112 DS0 1009 DS1 1009 [MD ... V DIR IE BRK S Z .AC . P . C]
AW 1000 BW 0000 CW 0001 DW 1009 1 111 . . 1 . . 1 . . . 1 1 .

Step_PC 804C:006BH MOV (BP-02),AL
Next_PC 804C:006EH
PC 006E SP 7EEA IX 0000 IY 004D BP 7EEE PSW F246
PS 804C SS 1112 DS0 1009 DS1 1009 [MD ... V DIR IE BRK S Z .AC . P . C]
AW 1000 BW 0000 CW 0001 DW 1009 1 111 . . 1 . . 1 . . . 1 1 .

Step_PC 804C:006EH OR AL,AL
Next_PC 804C:0070H
PC 0070 SP 7EEA IX 0000 IY 004D BP 7EEE PSW F246
PS 804C SS 1112 DS0 1009 DS1 1009 [MD ... V DIR IE BRK S Z .AC . P . C]
AW 1000 BW 0000 CW 0001 DW 1009 1 111 . . 1 . . 1 . . . 1 1 .

STATUS: n70216--Stepping complete_____...R....
step

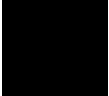
run trace step display modify break end ---ETC--
```

Continue user program execution with the "run" command.

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 look at the execution of the sample program after the address of the first instruction in the **Write_Msg** function (cmd_rds.c : line 4). To trigger on this address, enter:

```
trace after line 4 <RETURN>
```

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

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

The status line now shows "Emulation trace complete".

Displaying the Trace To display the trace, enter:

display trace <RETURN>

```
Trace List      Depth=512      Offset=0
Label:         Address      Data      Opcode or Status w/ Source Lines  time count
Base:         symbols      hex       mnemonic w/symbols                relative
after  PROG|_Write_Msg      04C8      04C8      fetch                               840      nS
#####../demo/emul/hp64792/cmd_rds.c - line 1 thru 5 #####
volatile char Cmd_Input;
char Msg_Dest[0x20];

void Write_Msg (const char *s)
{
+001  PROG|_Write_Msg      0000      PREPARE  0004,00                          520      nS
+002  |Write_Msg+00002     0000      0000      fetch                               360      nS
+003  |Write_Msg+00004     B81E      B81E      fetch                               880      nS
+004  |Write_Msg+00004     7EEE      PUSH      DS0                                760      nS
+005  ct5CAAa03:+07EE0    7EEE      7EEE      memory write                        120      nS
+006  |Write_Msg+00005     1009      MOV       AW,1009                             880      nS
+007  ct5CAAa03:+07EDA    1009      1009      memory write                        240      nS
+008  |Write_Msg+00006     1009      1009      fetch                               880      nS

STATUS:  n70216--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.

If there is data that does not appear on the screen, you can use <CTRL>f and <CTRL>g to roll the display left and right. The trace labels, shown on the second line of the display, are described earlier in this section.

To display the remaining lines of the trace, press the <PGDN> or <NEXT> key.

Displaying Trace with No Symbol

The trace listing shown above has symbol information because of the "set symbols on" setting before in this chapter. To see the trace listing with no symbol information, enter the following command.

set symbols off

```
Trace List      Depth=512      Offset=0
Label:  Address  Data      Opcode or Status w/ Source Lines      time count
Base:   hex      hex              mnemonic                                relative
after   804CC      04C8      04C8  fetch                                840      nS
#####.../demo/emul/hp64792/cmd_rds.c - line      1 thru      5 #####
volatile char Cmd_Input;
char Msg_Dest[0x20];

void Write_Msg (const char *s)
{
+001    804CC      0000      PREPARE 0004,00                                520      nS
+002    804CE      0000      0000  fetch                                360      nS
+003    804D0      B81E      B81E  fetch                                880      nS
+004    804D0      7EEE      PUSH   DS0                                760      nS
+005    19000      7EEE      7EEE  memory write                             120      nS
+006    804D1      1009      MOV    AW,1009                             880      nS
+007    18FFA      1009      1009  memory write                             240      nS
+008    804D2      1009      1009  fetch                                880      nS

STATUS:  n70216--Running user program      Emulation trace complete_____...R...
set symbols off

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

As you can see, the analysis trace display shows the trace list without symbol information.

Displaying Trace with Time Count Absolute

Enter the following command to display count information relative to the trigger state.

display trace count absolute <RETURN>

```
Trace List      Depth=512      Offset=0
Label:  Address  Data      Opcode or Status w/ Source Lines      time count
Base:   hex      hex      mnemonic                                     absolute
after  804CC      04C8      04C8  fetch                                     -----
#####.../demo/emul/hp64792/cmd_rds.c - line 1 thru 5 #####
volatile char Cmd_Input;
char Msg_Dest[0x20];

void Write_Msg (const char *s)
{
+001   804CC      0000      PREPARE 0004,00                          + 520      nS
+002   804CE      0000      0000  fetch                                     + 880      nS
+003   804D0      B81E      B81E  fetch                                     + 1.76     uS
+004   804D0      7EEE      PUSH   DS0                                + 2.52     uS
+005   19000      7EEE      7EEE  memory write                             + 2.64     uS
+006   804D1      1009      MOV    AW,1009                            + 3.52     uS
+007   18FFA      1009      1009  memory write                             + 3.76     uS
+008   804D2      1009      1009  fetch                                     + 4.64     uS

STATUS:  n70216--Running user program  Emulation trace complete_____R...
display  trace  count  absolute

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

Displaying Trace with Compress Mode

If you want to see more executed instructions on a display, the 70216 emulator Softkey Interface provides **compress mode** for analysis display. To see trace display with compress mode, enter the following command:

display trace compress on <RETURN>

As you can see, the analysis trace display shows the analysis trace lists without prefetch cycles. With this command you can examine program execution easily.

If you want to see all of cycles including fetch cycles, enter following command:

display trace compress off <RETURN>

The trace display shows you all of the cycles the emulation analyzer have captured.

```
Trace List      Depth=512      Offset=0
Label:  Address  Data      Opcode or Status w/ Source Lines      time count
Base:   hex      hex      mnemonic      absolute
#####.../demo/emul/hp64792/cmd_rds.c - line      1 thru      5 #####
volatile char Cmd_Input;
char Msg_Dest[0x20];

void Write_Msg (const char *s)
{
+001      804CC      0000  PREPARE 0004,00      + 520      nS
+004      804D0      7EEE  PUSH    DS0      + 2.52      uS
+005      19000      7EEE  7EEE  memory write      + 2.64      uS
+006      804D1      1009  MOV     AW,1009      + 3.52      uS
+007      18FFA      1009  1009  memory write      + 3.76      uS
+010      804D4      46C7  MOV     DS0,AW      + 6.00      uS
#####.../demo/emul/hp64792/cmd_rds.c - line      6 thru      8 #####
      char *Dest_Ptr;

STATUS:  n70216--Running user program      Emulation trace complete_____...R...
display trace compress on

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

Note



When the analysis trace is displayed with compress mode, the time count may not indicate correct time counts. This happens when time count is **relative**. Since the compress mode feature is implemented by eliminating prefetch cycles when displaying analysis trace, relative time count shows incorrect value. If you are interested in the time count, display with time count **absolute**. Absolute value of time count always show correct value. Keep this note in your mind when display the trace with compress mode.

Changing the Trace Depth

The default states displayed in the trace list is 256 states. To reduce the number of states, use the "display trace depth" command.

```
display trace depth 512 <RETURN>
```

Emulator Analysis Status Qualifiers

The following analysis status qualifiers may also be used with the 70216 emulator.

Qualifier	Status Bits	Description
exec	0xxx0xxxxxxxxxy	execute instruction
fetch	0xxx1xxxx001x100y	program fetch
read	0xxx1xxxxxx0xx01y	read
write	0xxx1xxxxxx0xx10y	write
mem	0xxx1xxxxxx0xlxy	memory access
intio	0xxx1xxxx00000xxy	internal I/O access
extio	0xxx1xxxx00010xxy	external I/O access
cpu	0xxx1xxxx00xxxxxy	cpu cycle
dma	0xxx1xxxx10x01xy	DMA memory access
casdma	0xxx1xxxx1010111y	cascaded DMA cycle
refresh	0xxx1xxxx0100101y	refresh cycle
holdack	0xxx1xxxx11xxxxxy	hold acknowledge
intack	0xxx1xxxx001x000y	interrupt acknowledge
haltack	0xxx1xxxxxx1011y	halt acknowledge
em80	0xx1xxxxxxxxxxxxxy	8080 emulation mode
native	0xx0xxxxxxxxxxxxxy	native mode
ds0	0xxx1xx11xxxxxxxxxy	ds0 use cycle
ds1	0xxx1xx00xxxxxxxxxy	ds1 use cycle
ss	0xxx1xx01xxxxxxxxxy	ss use cycle
ps	0xxx1xx10xxxxxxxxxy	ps use cycle
rom	0xxx1x0xxxxxxxxxy	rom access
grd	0xxx10xxxxxxxxxy	guarded memory access
usr	0xlxxxxxxxxxxxxxy	user cycle
mon	0x0xxxxxxxxxxxxxy	monitor cycle

Resetting the Emulator

To reset the emulator, enter the following command.

```
reset <RETURN>
```

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 **pmon** and **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

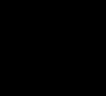
When you enter the Softkey Interface via **pmon** and **MEAS_SYS**, you have the option to select the measurement system display or another module in the measurement system when exiting the Softkey Interface. This type of exit is also "locked"; that is, you can continue the emulation session later. For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



"In-Circuit" Emulation

Introduction

The emulator is *in-circuit* when it is plugged into the target system. This chapter covers topics which relate to in-circuit emulation.

This chapter will:

- Describe the issues concerning the installation of the emulator probe into target systems.
- Show you how to install the emulator probe.
- Show you how to use features related to in-circuit emulation.

Prerequisites

Before performing the tasks described in this chapter, you should be familiar with how the emulator operates in general. Refer to the *Concepts of Emulator and Analysis* manual and the "Getting Started" chapter of this manual.

Installing the Target System Probe

The 70216 emulator probe has a 68-pin PLCC connector; The 70216 emulator is shipped with a pin protector over the target system probe. This guard is designed to prevent impact damage to the pins and should be left in place while you are not using the emulator.

Caution



DAMAGE TO THE EMULATOR CIRCUITRY MAY RESULT IF THESE PRECAUTIONS ARE NOT OBSERVED. The following precautions should be taken while using the 70216 emulator.

Power Down Target System. Turn off power to the user target system and to the 70216 emulator before inserting the user plug to avoid circuit damage resulting from voltage transients or mis-insertion of the user plug.

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

Protect Against Static Discharge. The 70216 emulator contains devices which are susceptible to damage by static discharge. Therefore, operators should take precautionary measures before handling the user plug to avoid emulator damage.

Protect Target System CMOS Components. If your target system includes any CMOS components, turn on the target system first, then turn on the 70216 emulator; when powering down, turn off the emulator first, then turn off power to the target system.

Auxiliary Output Line

One auxiliary output line, "TARGET BUFFER DISABLE", is provided with the 70216 emulator.

Caution



DAMAGE TO THE EMULATOR PROBE WILL RESULT IF THE AUXILIARY OUTPUT LINES ARE INCORRECTLY INSTALLED.

When installing the auxiliary output line into the end of the emulator probe cable, make sure that the ground pin on the auxiliary output line (labeled with white dots) is matched with the ground receptacles in the end of the emulator probe cable.

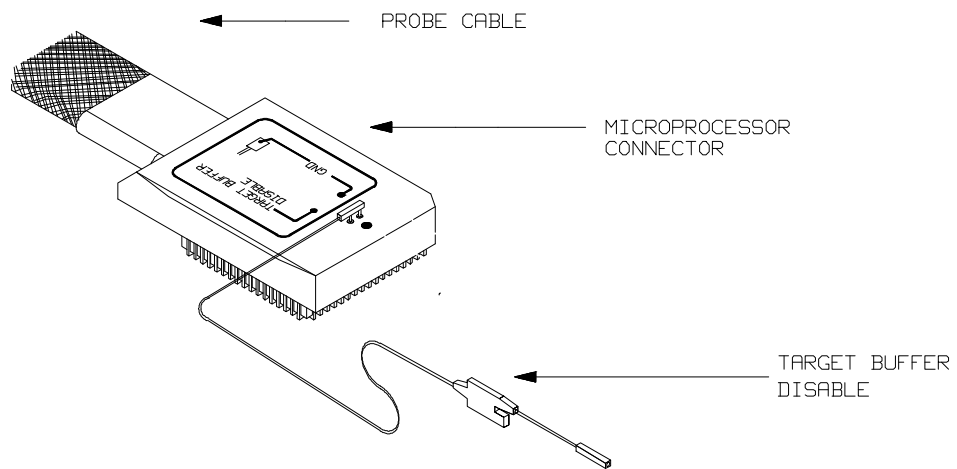
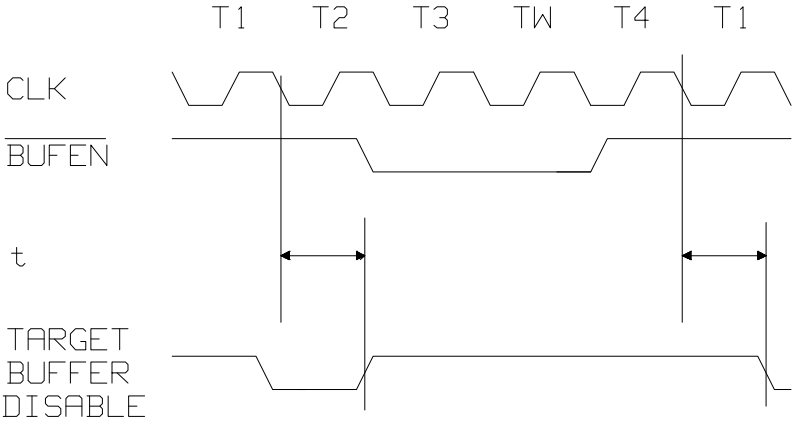


Figure 3-1. Auxiliary Output Lines (70216 Emulator)

TARGET BUFFER DISABLE ---This active-high output is used when the co-processor memory accesses to emulation memory will be operated. This output is used to tristate (in other words, select the high Z output) any target system devices on the 70216 data bus. Target system devices should be tristated because co-processor memory reads from emulation memory will cause data to be output on the user probe.

This "TARGET BUFFER DISABLE" output will be driven with the following timing in the co-processor memory access cycle.



The time 't' is

30 nsec MAX. (70208/70216/
70208H/70216H Emulator)

3-4 In-Circuit Emulation Topics

Installing into a PLCC Type Socket

To connect the microprocessor connector to the target system, proceed with the following instructions.

- Remove the 70216 microprocessor (PLCC type) from the target system socket. Note the location of pin 1 on the microprocessor and on the target system socket.
- Store the microprocessor in a protected environment (such as antistatic form).
- Install the microprocessor connector into the target system microprocessor socket.

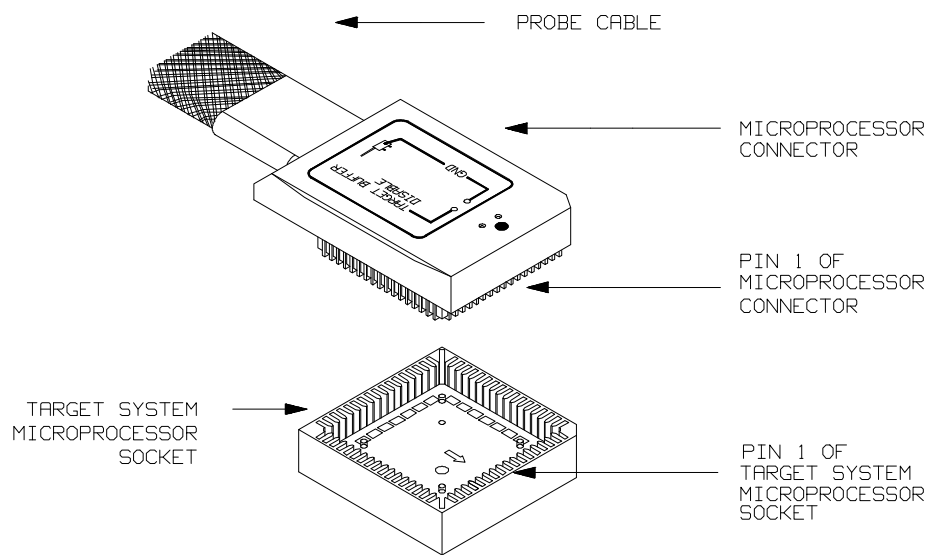


Figure 3-2. Installing into a PLCC type socket

Installing into a PGA Type Socket

You can use an ITT CANNON "LCS-68-12" PLCC connector to plug into the target system socket of an PGA type. You may use this socket with the pin protector to connect the microprocessor connector to the target system.

To connect the microprocessor connector to the target system, proceed with the following instructions.

- Remove the 70216 microprocessor (PGA type) from the target system socket. Note the location of pin A1 on the microprocessor and on the target system socket.
- Store the microprocessor in a protected environment (such as antistatic form).
- Place the microprocessor connector with a PLCC-to-PGA socket and a pin protector (see figure 3-3), attached to the end of the probe cable, into the target system microprocessor socket.

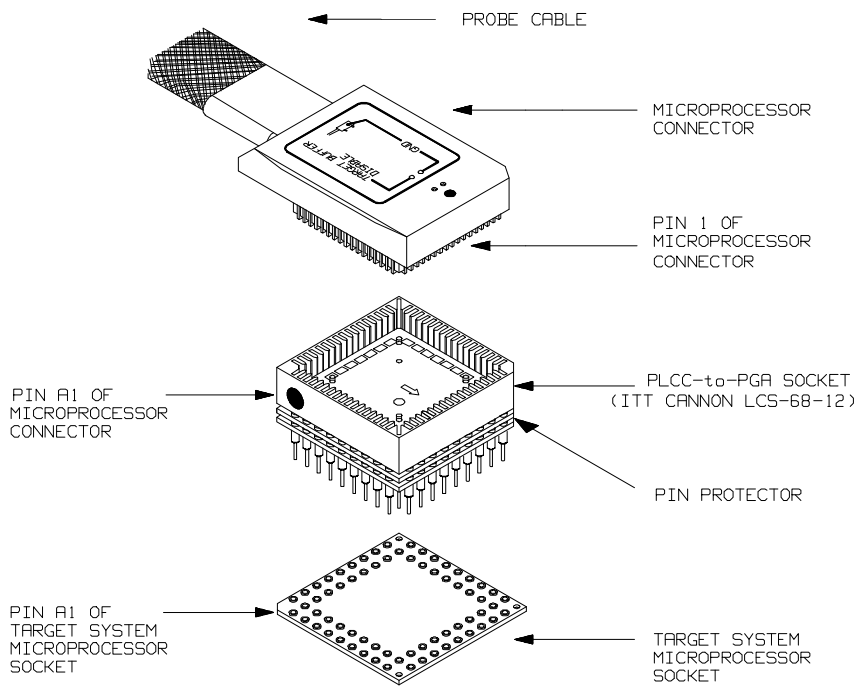


Figure 3-3. Installing into a PGA type socket

In-Circuit Configuration Options

The 70216 emulator provide configuration options for the following in-circuit emulation issues. Refer to the chapter on "Configuring the Emulator" for more information on these configuration options.

Using the Target System Clock Source

In the 70208/70216 Emulator, the default emulator configuration selects the internal 8 MHz (system clock speed) clock as the emulator clock source. In the 70208H/70216H Emulator, the default emulator configuration selects the internal 16 MHz (system clock speed) clock as the emulator clock source. You should configure the emulator to select an external target system clock source for the "in-circuit" emulation.

Allowing the Target System to Insert Wait States

High-speed emulation memory provides no-wait-state operation. However, the emulator may optionally respond to the target system ready line while emulation memory is being accessed.

Note



When you use the i8087 coprocessor on your target system connected to 70216 microprocessor, the i8087 can access 70216 emulation memory on coprocessor memory read/write cycles.

In this case, you should reset the target system to connect the 70216 emulator to the i8087 coprocessor before starting emulation session.

Enabling NMI and RESET Input from the Target System

You can configure whether the emulator should accept or ignore the NMI and RESET signals from the target system.

Running the Emulator from Target Reset

You can specify that the emulator begins executing from target system reset. When the target system RESET 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 RESET signal by the target system (see the "Enable RESET inputs from target system?" configuration in Chapter 4 of this manual).

To specify a run from target system reset, select:

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

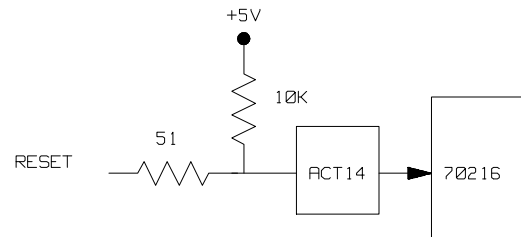
The status now shows that the emulator is "Awaiting target reset". After the target system is reset, the status line message will change to show the appropriate emulator status.



Target System Interface

$\overline{\text{RESET}}$

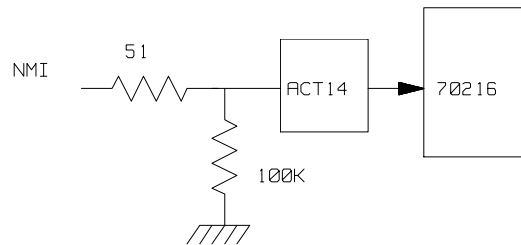
This signal is connected to 70216 through ACT14, 51ohm and 10K ohm pull-up register.



N

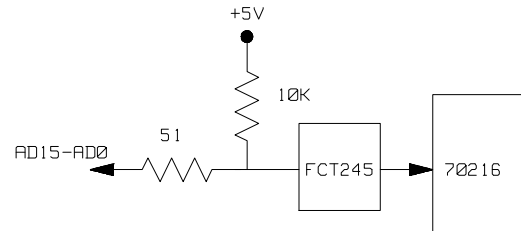
MI

This signal is connected to 70216 through ACT14, 51 ohm and 100K ohm pull-down register.



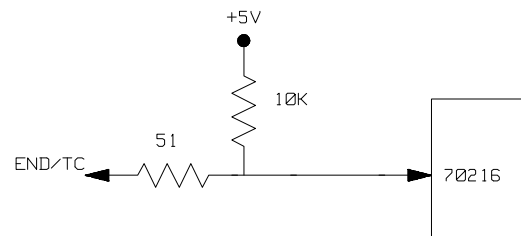
AD15-AD0

These signals are connected to 70216 through FCT245, 51 ohm and 10K ohm pull-up register.



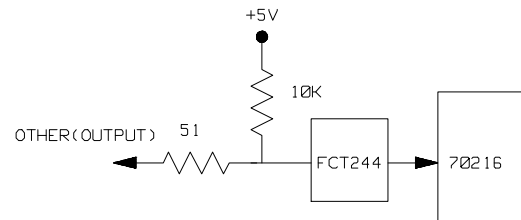
$\overline{\text{END/TC}}$

This signal is connected to 70216 through 51 ohm and 10K ohm pull-up register.



OTHER(OUTPUT)

These signals are connected to 70216 through FCT244, 51 ohm and 10K ohm pull-up registers.



Notes



Configuring the Emulator

Introduction

Your 70216 emulator can be used in all stages of target system development. For instance, you can run the emulator out-of-circuit when developing target system software, or you can use the emulator in-circuit when integrating software with target system hardware. Emulation memory can be used in place of, or along with, target system memory. You can use the emulator's internal clock or the target system clock. You can execute target programs in real-time or allow emulator execution to be diverted into the monitor when commands request access of target system resources (target system memory, register contents, etc.)

The emulator is a flexible instrument and it may be configured to suit your needs at any stage of the development process. This chapter describes the options available when configuring the 70216 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.

Memory Configuration:

- Selecting the emulation monitor type.
- Mapping memory.

Emulator Pod Configuration:

- Enabling responding to DMARQ0-3 from target system in background cycles.
- Enabling using to FPP (Floating Point Processor) on target system.
- Selecting mnemonic type for memory display.
- Selecting dis-assembler mode for assembler format.
- Selecting segment algorithm for physical run addresses.
- Specifying Reset value for the stack pointer.
- Enabling RESET inputs from target system.
- Enabling NMI inputs from target system.
- Enabling READY inputs from target system.
- Enabling HLDRQ (Hold Request) inputs from target system.
- Selecting target memory and I/O access size.

4-2 Configuring the Emulator

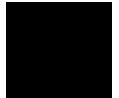
Debug/Trace Configuration:

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

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

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

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



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. In the 70208/70216 Emulator, the emulators' internal clock speed is 8MHz (system clock). In the 70208H/70216H Emulator, the emulators' internal clock speed is 16MHz (system clock).

external

Selects an external target system clock source, from 4 MHz up to 20 MHz can be entered in using the 70208/70216 emulator.

In using the 70208H/70216H emulator, from 2 to 32 MHz can be entered.

Note



Changing the clock source drives the emulator into the reset state. If you answer "yes" to the "Enter monitor after configuration?" question that follows, the emulator resets (due to the clock source change) then breaks into the monitor when the configuration is saved.

Enter Monitor After Configuration?

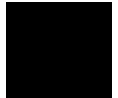
This question allows you to select whether the emulator will be running in the monitor or held in the reset state upon completion of the emulator configuration.

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 target system memory.
 - Display/modify I/O.



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.

Memory Configuration

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

Modify memory configuration?

Monitor Type?

The monitor is a program which is executed by the emulation processor. It allows the emulation system controller to access target system resources. For example, when you enter a command that requires access to target system resources (display target memory, for example), the system controller writes a command code to a communications area and breaks the execution of the emulation processor into the monitor. The monitor program then reads the command from the communications area and executes the processor instructions which access the target system. After the monitor has performed its task, execution returns to the user program. Monitor program execution can take place in the "background" or "foreground" emulator modes.

In the *foreground* emulator mode, the emulator operates as would the target system processor.

In the *background* emulator mode, foreground execution is suspended so that the emulation processor may be used for communication with the system controller, typically to perform tasks which access target system resources.

A *background monitor* program operates entirely in the background emulator mode; that is, the monitor program does not execute as if it were part of the target program. The background monitor does not take up any processor address space and does not need to be linked to the target program. The monitor resides in dedicated background memory.

A *foreground monitor* program performs its tasks in the foreground emulator mode; that is, the monitor program executes as if it were part of the target program. Breaks into the monitor always put the emulator in the background mode; however, foreground monitors switch back to the foreground mode before performing monitor functions.

Note



All memory mapper terms are deleted when the monitor type is changed!

background The default emulator configuration selects the background monitor. A memory overlay is created and the background monitor is loaded into that area.

Note



While running in background monitor, the 70216 emulator ignores target system reset.

When the background monitor is selected, the execution of the monitor is hidden from the target system (except for background cycles). When you select the background monitor and the current monitor type is "foreground", you are asked the next question.

1. Reset map (change of monitor type requires map reset)?

This question will be asked if you change the monitor type (in this case, you have changed the monitor type from "foreground" to "background"). This question reminds you that the map will be reset and allows you to confirm your decision.

no The memory map is not reset, and the monitor type is not changed.

yes This memory map is reset due to the change in monitor type.

foreground When you select the foreground monitor, processor address space is taken up. The foreground monitor takes up 4K bytes of memory. When the foreground monitor is selected, breaking into the monitor still occurs in a brief background state, but the rest of the monitor program, the saving of registers and the dispatching of emulation commands, is executed in foreground.

Note



You must **not** use the foreground monitor if you wish to perform coordinated measurements.

When you select the foreground monitor and the current monitor type is "background", you are asked the next question.

1. Reset map (change of monitor type requires map reset)?

This question will be asked if you change the monitor type (in this case, you have changed the monitor type from "background" to "foreground"). This question reminds you that the map will be reset and allows you to confirm your decision.

no The memory map is not reset, and the monitor type is not changed.

yes This memory map is reset due to the change in monitor type.

2. Foreground monitor location?

You can relocate the monitor to any 4K byte boundary. The location of a foreground monitor is important because it will occupy part of the processor address space. Foreground monitor locations must not overlap the locations of target system programs. When entering monitor block addresses, you must only specify addresses on 4K byte boundaries; otherwise, the configuration will be invalid, and the previous configuration will be restored.

Note



You should not load the foreground monitor provided with the 70216 emulator at the base address 0 or 0ff000 hex; the 70216 microprocessor's vector table is located.

3. Monitor filename?

This question allows you to specify the name of the foreground monitor program absolute file. Remember that you must assemble and link your foreground monitor starting at the 4K byte boundary specified for the previous "Foreground monitor location?" question.

The monitor program will loaded after you have answered all the configuration questions.

Only the 4 kilobytes of memory reserved for the monitor are loaded at the end of configuration; therefore, you should not link the foreground monitor to the user program. If it is important that the symbol database contain both monitor and user program symbols, you can create a different absolute file in which the monitor and user program are linked. Then, you can load this file after configuration.

Using the Foreground Monitor. When using the foreground monitor, your program should set up a stack. The foreground monitor assumes that there is a stack in the foreground program, and this stack is used to save PS, PC, and PSW upon entry into the monitor.



Mapping Memory

Depending on the memory model number, emulation memory consists of 128, 512 or 1024 kilobytes, mappable in 256 byte blocks. However, you may use 124, 508 or 1020 kilobytes of emulation memory for your target system, because 4 kilobytes of emulation memory specified by the "Foreground monitor location?" question is required for the execution of the monitor. The emulation memory system does not introduce wait states.

Note



You can insert wait states on accessing emulation memory. Refer to the "Respond to READY from the target system for accessing to emulation memory?" section in this chapter.

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.

When a foreground monitor selected, a 4 kilobyte block is automatically mapped at the address specified by the "Foreground monitor location?" question.

Note



Target system accesses to emulation memory are not allowed.

Target system devices that take control of the bus (for example, DMA controllers) cannot access emulation memory.

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 "Break Processor on writes to ROM?" configuration item is enabled (see the "Debug/Trace Configuration" section which follows).

Determining the Locations to be Mapped

Typically, assemblers generate relocatable files and linkers combine relocatable files to form the absolute file. The linker load map listing will show what locations your program will occupy in memory.

Emulator Pod Configuration

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

Modify emulator pod configuration?

Respond to DMARQ0-3 from target system in background?

This configuration allows you to specify whether or not the emulator accepts DMARQ0-3 (DMA Request 0-3) signal generated by the target system in background.

yes The emulator accepts DMARQ0-3 signals. When the DMARQ0-3 are accepted, the emulator will respond as actual microprocessor.

no The emulator ignores DMARQ0-3 signals from target system completely in background. The 70216 emulator ignored DMA request from internal DMA controller until the emulator goes into foreground operation.

Use FPP on target system?

This configuration allows you to use FPP(Floating Point co-Processor) and to specify whether the emulator will drive the target system bus during ANY bus cycle.

yes Specifies your target system has FPP to work with the emulator. The i8087 on your target system can read co-processor instructions on the emulation memory.

no Specifies target system does not have FPP. The data bus signals are not driven to the target system when the emulator access to the emulation memory.

When "Yes" is selected, a special hardware mode which allows the emulator to support a floating point co-processor is enabled. When a floating point co-processor is present, it must monitor all address and data that the emulation processor inputs and outputs. Because of this, it is necessary to enable data bus drivers to the target system for all

emulation memory read cycles. This is normally done only on write cycles, and is not done on read cycles to avoid bus contention problems between the emulator and the target system. When this mode is enabled, the USER output from the pod should be used to disable user buffers that would normally be turned on when the emulator is reading from emulation memory. Also you should also select "yes" at the "Respond to HLDRQ from target system" configuration question for target hold signal input.

**Memory display mnemonic?
(70208/70208H Emulator)**

This configuration specifies the type of mnemonic that are used by the monitor program to display memory. When a command requests the monitor to display memory, the monitor program will look at the mnemonic type setting to determine whether uPD70208 (V40) or iAPX88/10 (8088) mnemonic should be used.

70208 Selecting the 70208 mnemonic type specifies that the emulator will display memory with uPD70208 (V40) mnemonic.

8088 Selecting the 8088 mnemonic type specifies that the emulator will display memory with iAPX88/10 (8088) mnemonic.

The default emulator configuration selects the **70208** mnemonic type at power up initialization.

**Memory display mnemonic?
(70216/70216H Emulator)**

This configuration specifies the type of mnemonic that are used by the monitor program to display memory. When a command requests the monitor to display memory, the monitor program will look at the mnemonic type setting to determine whether uPD70216 (V50) or iAPX86/10 (8086) mnemonic should be used.

70216 Selecting the 70216 mnemonic type specifies that the emulator will display memory with uPD70216 (V50) mnemonic.

8086 Selecting the 8086 mnemonic type specifies that the emulator will display memory with iAPX86/10 (8086) mnemonic.

The default emulator configuration selects the **70216** mnemonic type at power up initialization.

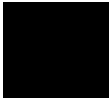
Dis-assembler mode?

This configuration specifies the mode of dis-assembler that are used by the monitor program to display assembler format. When a command requests the monitor to display memory, the monitor program will look at the dis-assembler mode setting to determine whether AxLS(HP64873) or OLS(HP64853) assembler format should be used.

native Selecting the native mode specifies that the emulator will display dis-assembler with AxLS(HP64873) assembler format.

64853 Selecting the 64853 mode specifies that the emulator will display dis-assembler with OLS(HP64853) assembler format.

The default emulator configuration selects the **native** mode at power up initialization.



Segment algorithm ?

The run and step commands allow you to enter addresses in either logical form (segment:offset, e.g., 0F000H:0000H) or physical form (e.g., 0F000H). When a physical address (non-segmented) is entered with either a run or step command, the emulator must convert it to a logical (segment:offset) address.

minseg Specifies that the physical run address is converted such that the low 16 bits of the address become the offset value. The physical address is right-shifted 4 bits and ANDed with 0F000H to yield the segment value.

```
logical_addr = ((phys_addr >> 4) & 0xf000):(phys_addr & 0xffff)
```

maxseg Specifies that the low 4 bits of the physical address become the offset. The physical address is right-shifted 4 bits to yield the segment value.

```
logical_addr = (phys_addr >> 4):(phys_addr & 0xf)
```

curseg Specifies that the value entered with either a run or step command (0 thru 0ffff hex) becomes the offset. In this selecting, the current segment value is not changed.

```
logical_addr = (current segment):(entered value)
```

If you use logical addresses other than the three methods which follow, you must enter run and step addresses in logical form.

Reset value for the stack pointer?

This question allows you to specify the value to which the stack segment (SS) and stack pointer (SP) will be set on entrance to the emulation monitor initiated RESET state (the "Emulation reset" status).

The address specified in response to this question must be a physical address. The emulator convert it to a logical address (<SP>:<SS>). When you enter "phys_addr" to this configuration, SS and SP will be set as follows.

```
SS = (phys_addr >> 4) & 0xf000
SP = phys_addr & 0xffff
```

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

Note



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

Respond to RESET from target system?

The 70216 emulator can respond or ignore target system reset while running in user program or waiting for target system reset (refer to "run from reset" command in the *Softkey Interface Reference* manual). While running in background monitor, the 70216 emulator ignores target system reset completely independent on this setting.

yes Specify that, this is a default configuration, make the emulator to respond to reset from target system. In this configuration, emulator will accept reset and execute from reset vector (0FFFF0 hex) as same manner as actual microprocessor after reset is inactivated.

no The emulator ignores reset signal from target system completely, even while in foreground (executing user program).

**Respond to NMI
from target system?**

This question allows you to specify whether or not the emulation processor accepts NMI signal generated by the target system.

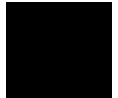
yes The emulator accepts NMI signal generated by the target system. When the NMI is accepted, the emulator calls the NMI procedure as actual microprocessor. Therefore, you need to set up the NMI vector table, if you want to use the NMI interrupt.

no The emulator ignores NMI signal from target system completely.

Note



When target NMI signal is enabled , it is in effect while the emulator is running in the target program. while the emulator is running monitor, NMI will be ignored until the monitor is finished.



**Respond to READY
from target system
for accessing to
emulation memory?**

High-speed emulation memory provides no-wait-state operation. However, the emulator may optionally respond to the target system ready line while emulation memory is being accessed.

- no** When the ready relationship is not locked to the target system, emulation memory accesses ignore ready signals from the target system (no wait states are inserted).
- yes** When the ready relationship is locked to the target system, emulation memory accesses honor ready signals from the target system (wait states are inserted if requested).

**Respond to
HLDRQ from target
system?**

This configuration allows you to specify whether or not the emulator accepts HLDRQ (Bus Hold Request) signal generated by the target system.

- no** The emulator ignores HLDRQ signal from target system completely.
- yes** The emulator accepts HLDRQ signal. When the HLDRQ is accepted, the emulator will respond as actual microprocessor.

Target memory access size?

This configuration specifies the type of microprocessor cycles that are used by the monitor program to access target memory or I/O locations. When a command requests the monitor to read or write to target system memory or I/O, the monitor program will look at the access mode setting to determine whether byte or word instructions should be used.

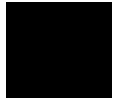
Bytes

Selecting the byte access mode specifies that the emulator will access target memory using upper and lower byte cycles (one byte at a time).

Words

Selecting the word access mode specifies that the emulator will access target memory using word cycles (one word at a time) at an even address. When the emulator read or write odd number of byte data, the emulator will read or write the last byte data using byte cycle. At an odd address, the emulator will access target memory using byte cycles.

The default emulator configuration selects the **byte** access size at power up initialization. Access mode specifications are saved; that is, when a command changes the access mode, the new access mode becomes the current default.



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. To access the debug/trace 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, they cannot prevent writes to target system RAM locations which are mapped as ROM, even though the write to ROM break is enabled.

- yes** Causes the emulator to break into the emulation monitor whenever the user program attempts to write to a memory region mapped as ROM.
- no** The emulator will not break to the monitor upon a write to ROM. The emulator will not modify the memory location if it is in emulation ROM.

Note



The **wrrom** trace command status option allows 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>

Trace Background or Foreground Operation?

This question allows you to specify whether the analyzer trace only foreground emulation processor cycles, only background cycles, or both foreground or background cycles.

foreground Specifies that the analyzer trace only foreground cycles. This option is specified by the default emulator configuration.

background Specifies that the analyzer trace only background cycles. (This is rarely a useful setting.)

both Specifies that the analyzer trace both foreground and background cycles. You may wish to specify this option so that all emulation processor cycles may be viewed in the trace display.

Trace Internal DMA cycles?

This question allows you to specify whether or not the analyzer trace the emulation processor's internal DMA cycles.

yes Specifies that the analyzer will trace the internal DMA cycles.

no Specifies that the analyzer will not trace the internal DMA cycles.

Trace bus cycles in HOLD state ?

This question allows you to specify whether or not the analyzer trace the emulation processor's bus cycles in HOLD state.

yes Specifies that the analyzer will trace bus cycle in HOLD state.

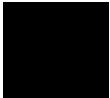
no Specifies that the analyzer will not trace bus cycles in HOLD state.

Trace refresh cycles?

This question allows you to specify whether or not the analyzer trace the emulation processor's refresh cycles.

- yes** Specifies that the analyzer will trace the refresh cycles.

- no** Specifies that the analyzer will not trace the refresh cycles.



Simulated I/O Configuration

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

External Analyzer Configuration

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

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.

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 two files; the filename specified with extensions of ".EA" and ".EB". The file with the ".EA" extension is the "source" copy of the file, and the file with the ".EB" extension is the "binary" or loadable copy of the file.

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>

Using the Emulator

Introduction

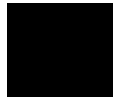
The "Getting Started" chapter shows you how to use the basic features of the 70216 emulator. This chapter describes the more in-depth features of the emulator.

This chapter discusses:

- Register names and classes.
- Features available via "pod_command".

This chapter shows you how to:

- Store the contents of memory into absolute files.
- Make coordinated measurements.



Register Names and Classes

The following register names and classes are used with the display/modify registers commands in 70216 emulator.

BASIC(*) class

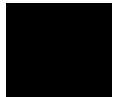
Register name	Description
AW, BW CW, DW BP, IX, IY DS0, DS1, SS SP, PC, PS, PSW	BASIC registers.

SIO class
(70208/70216 Emulator)
(System I/O registers)

Register name	Description
OPCN	On-chip peripheral connection register
OPSEL	On-chip peripheral selection register
OPHA	On-chip peripheral high address register
DULA	DMAU low address register
IULA	ICU low address register
TULA	TCU low address register
SULA	SCU low address register
WCY1	Programmable wait, cycle 1 register
WCY2	Programmable wait, cycle 2 register
WMB	Programmable wait, memory boundary register
RFC	Refresh control register
TCKS	Timer clock selection register

SIO class (System I/O registers)
(70208H/70216H
Emulator)

Register name	Description
OPCN	On-chip peripheral connection register
OPSEL	On-chip peripheral selection register
OPHA	On-chip peripheral high address register
DULA	DMAU low address register
IULA	ICU low address register
TULA	TCU low address register
SULA	SCU low address register
SCTL	System control register
WCY1	Programmable wait, cycle 1 register
WCY2	Programmable wait, cycle 2 register
WMB	Programmable wait, memory boundary register
RFC	Refresh control register
SBCR	Stand-by control register
TCKS	Timer clock selection register
EXWB	Extended wait block selection register
WSMB	Wait submemory block selection register
WIOB	Wait I/O block selection register
WCY3	Programmable wait, cycle 3 register
BRC	Boud rate counter
BADR	Bank address register
BSEL	Bank select register



ICU class (Interrupt Control Unit registers)

Register name	Description
IMKW	Interrupt mask word register
IRQ	Interrupt request register (Read only)
IIS	Interrupt in-service register (Read only)
IPOL	Interrupt polling register (Read only)
IPFW	Interrupt priority and finish word register (Write only)
IMDW	Interrupt mode word register (Write only)
IIW1	Interrupt initialize word 1 register (Write only)
IIW2	Interrupt initialize word 2 register (Write only)
IIW3	Interrupt initialize word 3 register (Write only)
IIW4	Interrupt initialize word 4 register (Write only)

Caution



When **ipol** register is displayed, interrupts are suspended until the FI command is published.

TCU class (Timer Control Unit registers)

Register name	Description
TCT0	Timer/counter 0 register
TST0	Timer status 0 register (Read only)
TCT1	Timer/counter 1 register
TST1	Timer status 1 register (Read only)
TCT2	Timer/counter 2 register
TST2	Timer status 2 register (Read only)
TMD	Timer/counter mode register (Write only)

SCU class (Serial Control Unit registers)

Register name	Description	
SRB	Serial receive data buffer	(Read only)
SST	Serial status register	(Read only)
STB	Serial transmit data buffer	(Write only)
SCM	Serial command register	(Write only)
SMD	Serial mode register	(Write only)
SIMK	Serial interrupt mask register	(Write only)

DMA71 class (DMA Control Unit registers (for uPD71071 mode))

Register name	Description	
DICM	DMA initialize register	(Write only)
DCH	DMA channel register	
DBC_DCC0	DMA base/current count register channel 0	
DBC_DCC1	DMA base/current count register channel 1	
DBC_DCC2	DMA base/current count register channel 2	
DBC_DCC3	DMA base/current count register channel 3	
DBA_DCA0	DMA base/current address register channel 0	
DBA_DCA1	DMA base/current address register channel 1	
DBA_DCA2	DMA base/current address register channel 2	
DBA_DCA3	DMA base/current address register channel 3	
DMD0	DMA mode control register channel 0	
DMD1	DMA mode control register channel 1	
DMD2	DMA mode control register channel 2	
DMD3	DMA mode control register channel 3	
DDC	DMA device control register	
DST	DMA status register	(Read only)
DMK	DMA mask register	

DMA37 class (DMA Control Unit register (for uPD71037mode))
(70208H/70216H
Emulator only)

Register name	Description
CMD	DMA read status/write command register
BANK0	DMA bank register channel 0
BANK1	DMA bank register channel 1
BANK2	DMA bank register channel 2
BANK3	DMA bank register channel 3
ADR0	DMA current address register channel 0
ADR1	DMA current address register channel 1
ADR2	DMA current address register channel 2
ADR3	DMA current address register channel 3
CNT0	DMA current count register channel 0
CNT1	DMA current count register channel 1
CNT2	DMA current count register channel 2
CNT3	DMA current count register channel 3
SFRQ	Software DMA write request register (Write only)
SMSK	DMA write single mask register (Write only)
MODE	DMA write mode register
CLBP	DMA clear byte pointer F/F (Write only)
INIT	DMA initialize register (Write only)
CMSK	DMA clear mask register (Write only)
AMSK	DMA write all mask register bit (Write only)

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.
- Sequencing in the analyzer.
- 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.

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 800h thru 84fh to absfile  
<RETURN>
```

The command above causes the contents of memory locations 800H-84FH 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.

Using the Foreground Monitor

Introduction

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

The foreground monitors are supplied with the emulation software and can be found in the following path:

```
/usr/hp64000/monitor/*
```

The monitor programs named **Nfmon70208.s**, **Nfmon70208h.s**, **Nfmon70216.s**, and **Nfmon70216h.s** are for the HP 64873 V series AxLS Cross Assembler/Linker.

Note



Use the appropriate monitor; "Nfmon70208.s" for the 70208, "Nfmon70208h.s" for the 70208H, "Nfmon70216.s" for the 70216H and "Nfmon70216h.s" for the 70216H emulator. "Nfmon70216.s" foreground monitor program is used in this example. If your emulator is for the other emulator, read this appendix by replacing "Nfmon70216" with appropriate monitor.

Comparison of Foreground and Background Monitors

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

Background Monitors

A *background* monitor is an emulation monitor which overlays the processor's memory space with a separate memory region.

Usually, a background monitor will be easier to work with in starting a new design. The monitor is immediately available upon powerup, and you don't have to worry about linking in the monitor code or allocating space for the monitor to use the emulator. No assumptions are made about the target system environment; therefore, you can test and debug hardware before any target system code has been written. All of the processor's address space is available for target system use, since the monitor memory is overlaid on processor memory, rather than subtracted from processor memory. Processor resources such as interrupts are not fully taken by the background monitor.

However, all background monitors sacrifice some level of support for the target system. For example, when the emulation processor enters the monitor code to display registers, it will not respond to target system interrupt requests. This may pose serious problems for complex applications that rely on the microprocessor for real-time, non-intrusive support. Also, the background monitor code resides in emulator firmware and can't be modified to handle special conditions.

Foreground Monitors

A *foreground* monitor may be required for more complex debugging and integration applications. A foreground monitor is a block of code that runs in the same memory space as your program. Foreground monitors allow the emulator to service real-time events, such as interrupts, while executing in the monitor. For most multitasking, interrupt intensive applications, you will need to use a foreground monitor.

You can tailor the foreground monitor to meet your needs, such as servicing target system interrupts. However, the foreground monitor does use part of the processor's address space, which may cause problems in some target systems. You must also properly configure the emulator to use a foreground monitor (see the "Configuring the Emulator" chapter and the examples in this appendix).

You may link the foreground monitor with your code. However, if possible, linking the monitor separately is preferred. This allows the

monitor to be downloaded before the rest of your program. Linking monitor programs separately is more work initially, but it should prove worthwhile overall, since the monitor can then be loaded efficiently during the configuration process at the beginning of a session.

An Example Using the Foreground Monitor

In the following example, we will illustrate how to use a foreground monitor with the sample program from the "Getting Started" chapter. By using the emulation analyzer, we will also show how the emulator switches from state to state using a foreground monitor.

For this example, we will be using the foreground monitor for the HP 64873 V series AxLS Cross Assembler/Linker. We will locate the monitor at 1000H; the sample program will be located at 10000H and 80000H.

```
$ cp /usr/hp64000/monitor/Nfmon70216.s .  
<RETURN>
```

Modify EQU Statement

To use the monitor, you must modify the EQU statement near the top of the monitor listing to point to the base address where the monitor will be loaded.

```
$ chmod 644 Nfmon70216.s <RETURN>  
$ vi Nfmon70216.s <RETURN>
```

Modifying Location of the Foreground Monitor

In this case, we will load the monitor at 1000H, so the modified EQU statement looks like this:

```
MONSEGMENT      EQU      00100H
```

You can load the monitor at any base address on a 4K byte boundary.

Note



You should not load the foreground monitor provided with the 70216 emulator at the base address 0 or 0ff000 hex; the 70216 microprocessor's vector table is located.

Assemble and Link the Monitor

You can assemble and link the foreground monitor program with the following commands (which assume that `/usr/hp64000/bin` is defined in the PATH environment variable):

```
$ asv20 -Lh Nfmon70216.s > Nfmon70216.lis
<RETURN>
$ ldv20 -c Nfmon70216.k -Lh > Nfmon70216.map
<RETURN>
```

The "Nfmon70216.k" linker command file is shown below.

```
LOAD      Nfmon70216.o
SEG       ??DATA1/??INIT=001ffdH
END
```

The "??DATA1/??INIT" is used in the HP 64873 V series AxLS Cross Assembler/Linker. You should set the "??DATA1/??INIT" to the value added the offset value (0FFDH) to the foreground monitor address (In this example, 1000H). When you want to relocate the foreground monitor, you should modify the "??DATA1/??INIT" value in the linker command file for the new foreground monitor address.

If you aren't ready to use the sample program, do that now. Refer to the "Getting Started" chapter to copy the sample program files to the current directory.

Modifying the Emulator Configuration

The following assumes you are modifying the default emulator configuration (that is, the configuration present after initial entry into the emulator or entry after a previous exit using "end release_system"). Enter all the default answers except those shown below.

Modify memory configuration? yes

You must modify the memory configuration so that you can select the foreground monitor and map memory.

Monitor type? foreground

Specifies that you will be using a foreground monitor program.

Reset map (change of monitor type requires map reset)? yes

You must answer this question as shown to change the monitor type to foreground.

Monitor address? 1000h

Specifies that the monitor will reside in the 4K byte block from 1000H through 1FFFH.

Monitor file name? Nfmon70216

Enter the name of the foreground monitor absolute file. This file will be loaded at the end of configuration.

Mapping Memory for the Example

When you specify a foreground monitor and enter the monitor address, all existing memory mapper terms are deleted and a term for the monitor block will be added. Add the additional term to map memory for the sample program, and "end" out of the memory mapper.

```
0h thru 0fffh emulation ram <RETURN>
10000h thru 1ffffh emulation ram <RETURN>
80000h thru 80fffh emulation rom <RETURN>
default target ram <RETURN>
end <RETURN>
```

Modify emulator pod configuration? yes

You must modify the pod configuration so that you specify the value of the stack segment and stack pointer.

Reset value for the stack pointer? 10000h

Specifies the value of the stack segment to 1000h and the value of the stack pointer to 0000h.

Configuration file name? fmoncfg

If you wish to save the configuration specified above, answer this question as shown.

Load the Program Code

Now it's time to load the sample program. You can load the sample program with the following command:

```
load cmd_rds <RETURN>
```

Before running the sample program, you need to initialize the stack pointer by breaking the emulator out of reset:

```
reset <RETURN>
```

```
break <RETURN>
```

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

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

Single Step and Foreground Monitors

To use the "step" command to step through processor instructions with the foreground monitor listed in this chapter, you must modify the processor's interrupt vector table. The entry that you **must** modify is the "BRK flag" interrupt vector, located at 4H thru 7H. The "BRK flag" interrupt vector must point to the identifier UEE_BRK_FLAG in the foreground monitor. For example, to modify the "BRK flag" interrupt vector, enter the following commands:

```
load symbols Nfmon70216 <RETURN>  
display local_symbols_in Nfmon70216: <RETURN>
```

To see the value of UEE_BRK_FLAG, press the <NEXT> key to page down until the UEE_BRK_FLAG is displayed. You will see that the value of UEE_BRK_FLAG is 0100:0B82 hex.

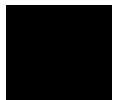
To modify the "BRK flag" interrupt vector to point to the UEE_BRK_FLAG, enter the following command:

```
modify memory 4h words to 0B82H,0100H  
<RETURN>
```

Now you can use the step feature. Enter:

```
load cmd_rds <RETURN>  
diplay registers <RETURN>  
step from transfer_address <RETURN>  
step <RETURN>
```

When you load the foreground monitor at the different base address, you should modify the "BRK flag" interrupt vector to point to the identifier UEE_BRK_FLAG with the same way.

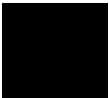


Limitations of Foreground Monitors

Listed below are limitations or restrictions present when using a foreground monitor.

Synchronized MeasurementsCMB

You cannot perform synchronized measurements over the CMB when using a foreground monitor. If you need to make such measurements, select the background monitor type when configuring the emulator.

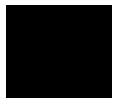


Index

- A**
 - absolute files
 - loading **2-11**
 - storing **5-8**
 - algorithm, current segment **4-15**
 - algorithm, maximum segment **4-15**
 - algorithm, minimum segment **4-15**
 - analyzer
 - configuring the external **4-23**
 - features of **1-4**
 - sequencing **5-7**
 - status qualifiers **2-33**
 - analyzer, using the **2-28**
 - assemblers **4-11**
 - assembling foreground monitor **A-4**
- B**
 - background **1-5, 4-7**
 - background cycles
 - tracing **4-21**
 - background monitor **4-7 - 4-8, A-2**
 - things to be aware of **4-8**
 - breaks
 - break command **2-24**
 - guarded memory accesses **4-11**
 - software breakpoints **2-18**
 - write to ROM **4-20**
- C**
 - caution statements
 - real-time dependent target system circuitry **4-6**
 - software breakpoint cmds. while running user code **2-18**
 - cautions
 - installing the target system probe **3-2**
 - characterization of memory **4-11**
 - clock source
 - external **3-8, 4-4**
 - internal **3-8, 4-4**
 - comparison of foreground/background monitors **A-1**
 - compress mode, trace display **2-32**

- configuration
 - example of using foreground monitor **A-4**
 - for running example program **2-8**
- configuration options
 - accept target NMI **4-17**
 - break processor on write to ROM **4-20**
 - dis-assembler mode **4-14**
 - enable READY input **4-18**
 - foreground monitor location **4-9**
 - honor target reset **4-16**
 - in-circuit **3-8**
 - mnemonic type(70208/70208H Emulator) **4-13**
 - mnemonic type(70216/70216H Emulator) **4-13**
 - monitor filename **4-10**
 - monitor type **4-7**
 - respond to DMARQ0-3 from target system in background **4-12**
 - respond to target HLDQR **4-18**
 - segment algorithm **4-15**
 - target access size **4-19**
 - trace background/foreground operation **4-21**
 - trace bus cycles in HOLD state **4-21**
 - trace internal DMA cycles **4-21**
 - trace refresh cycles **4-22**
 - use FPP on target system **4-12**
- coordinated measurements **4-23, 5-8**
- coprocessor
 - access emulation memory **3-8**
 - copy memory **5-7**
 - coverage analysis **5-7**
 - current segment algorithm **4-15**
- D** device table file **2-6**
- display command
 - memory mnemonic **2-15**
 - memory mnemonic with symbols **2-16**
 - registers **2-25**
 - symbols **2-12**
 - trace **2-29**
 - with source line **2-17**
- DMA **1-7**
 - external DMA controllers **4-11**
 - TC bit **1-7**

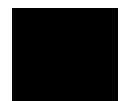
E emul700, command to enter the Softkey Interface **2-6, 2-35**
emulation analyzer **1-4**
emulation memory
 access by i8087 coprocessor **3-8**
 loading absolute files **2-11**
 note on target accesses **4-11**
 RAM and ROM characterization **4-11**
 size of **4-10**
emulation monitor
 foreground or background **1-5**
emulator
 before using **2-2**
 configuration **4-1**
 configure the emulator for example **2-8**
 device table file **2-6**
 feature list **1-3**
 prerequisites **2-2**
 purpose of **1-1**
 running from target reset **3-8 - 3-9**
 supported microprocessor package **1-3**
emulator configuration
 break processor on write to ROM **4-20**
 clock selection **4-4**
 for example **2-8**
 loading **4-24**
 monitor entry after **4-4**
 restrict to real-time runs **4-6**
 saving **4-23**
 stack pointer **4-16**
 trace background/foreground operation **4-21**
 trace bus cycles in HOLD state **4-21**
 trace internal DMA cycles **4-21**
 trace refresh cycles **4-22**
Emulator features
 emulation memory **1-4**
emulator probe
 installing **3-2**
END assembler directive (pseudo instruction) **2-20**
end command **2-34, 4-24**
Evaluation chip **1-7**
exit, Softkey Interface **2-34**



external analyzer
 configuration **4-23**
external clock source **4-4**

- F** file extensions
 .EA and .EB, configuration files **4-24**
files
 cmd_rds.A **2-3**
 cmd_rds.L **2-3**
foreground **1-5, 4-7**
foreground monitor **4-7 - 4-8, A-2**
 assembling/linking **A-4**
 configuration for sample program **A-4**
 example of using **A-3**
 location **4-9**
 location of shipped files **A-1**
 monitor program **4-10**
 relocating **A-3**
 single-step processor **A-7**
 things to be aware of **4-10**
 using the **A-1**
foreground operation, tracing **4-21**
- G** getting started **2-1**
 prerequisites **2-2**
global symbol **2-15**
global symbols
 displaying **2-12**
guarded memory accesses **4-11**
- H** help
 on-line **2-9**
 pod command information **2-10**
 softkey driven information **2-9**
- I** in-circuit configuration options **3-8**
in-circuit emulation **3-1**
installation **2-2**
 software **2-2**
interactive measurements **4-23**
internal clock source **4-4**
interrupt
 accepting NMI from target system **4-17**

- from target system **1-7, 3-8**
 - while stepping **1-7**
- L**
 - linkers **4-11**
 - linking foreground monitor **A-4**
 - load map **4-11**
 - loading absolute files **2-11**
 - loading emulator configurations **4-24**
 - local symbols
 - displaying **2-13**
 - location address
 - foreground monitor **4-9, A-4**
 - locked, end command option **2-35**
 - logical run address, conversion from physical address **4-15**
- M**
 - mapping memory **4-10**
 - maximum segment algorithm **4-15**
 - measurement system **2-35**
 - creating **2-5**
 - memory
 - characterization **4-11**
 - copying **5-7**
 - mapping **4-10**
 - mnemonic display **2-15**
 - mnemonic display with symbols **2-16**
 - modifying **2-23**
 - searching for strings or expressions **5-7**
 - with source line **2-17**
 - microprocessor package **1-3**
 - minimum segment algorithm **4-15**
 - mnemonic memory display **2-15**
 - modify command
 - configuration **4-1**
 - memory **2-23**
 - software breakpoints set **2-19**
 - module **2-35**
 - module, emulation **2-6**
 - monitor
 - background **4-7 - 4-8, A-2**
 - breaking into **2-24**
 - comparison of foreground/background **A-1**
 - description **4-7**

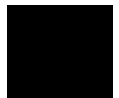


- foreground **4-7 - 4-8, A-2**
 - foreground monitor file **4-10**
 - foreground monitor location **4-9**
 - selecting entry after configuration **4-4**
 - using the foreground monitor **A-1**
- N**
- nosymbols **2-12**
 - note
 - pod command from keyboard **2-10**
 - notes
 - config. option for reset stack pointer recommended **4-16**
 - coordinated measurements require background. monitor **4-9**
 - mapper terms deleted when monitor type is changed **4-8**
 - pod commands that should not be executed **5-7**
 - selecting internal clock forces reset **4-4**
 - software breakpoints only at opcode addresses **2-19**
 - step not accepted **2-26**
 - target accesses to emulation memory **4-11**
 - use the appropriate foreground monitor program **A-1**
 - write to ROM analyzer status **4-20**
- O**
- OMF-86 absolute file format **2-11**
 - on-line help **2-9**
- P**
- PATH, HP-UX environment variable **2-5 - 2-6**
 - physical run address, conversion to logical run address **4-15**
 - Pin guard
 - target system probe **3-2**
 - pmon, User Interface Software **2-35**
 - pod_command **2-10**
 - features available with **5-7**
 - help information **2-10**
 - predefining stack pointer **4-16**
 - prerequisites for using the emulator **2-2**
 - program counter
 - mnemonic memory display **2-21**
- R**
- RAM, mapping emulation or target **4-11**
 - READY signal **4-18**
 - READY signals on accesses to emulation memory **4-10**
 - real-time execution
 - restricting the emulator to **4-6**
 - register commands **1-4**

- registers
 - classes **5-2**
 - display/modify **2-25**
 - names **5-2**
- release_system
 - end command option **2-34, 4-23 - 4-24**
- relocatable files **4-11**
- relocating foreground monitor **A-3**
- reset (emulator)
 - running from target reset **2-21, 3-9**
- reset (reset emulator) command **2-34**
- RESET signal **3-8, 4-16**
- restrict to real-time runs
 - emulator configuration **4-6**
 - permissible commands **4-6**
 - target system dependency **4-6**
- ROM
 - mapping emulation or target **4-11**
 - writes to **4-11**
- run address, conversion from physical address **4-15**
- run command **2-20**
- run from target reset **3-8 - 3-9, 4-16**

S

- sample program
 - description **2-3**
- saving the emulator configuration **4-23**
- sequencer, analyzer **5-7**
- softkey driven help information **2-9**
- Softkey Interface
 - entering **2-5**
 - exiting **2-34**
 - on-line help **2-9**
- software breakpoint
 - 70216 breakpoint interrupt instruction **2-18**
- software breakpoints **2-18**
 - enabling/disabling **2-19**
 - setting **2-19**
- software installation **2-2**
- source lines
 - displaying **2-14**
- ssimulated I/O **4-23**
- stack pointer, defining **4-16**



- stacks
 - using the foreground monitor **4-10**
- status qualifiers **2-33**
- step command **2-22, 2-26**
- string delimiters **2-10**
- symbols
 - displaying **2-12**
- synchronized measurement **A-8**
- system overview **2-2**

- T** target memory
 - loading absolute files **2-11**
 - RAM and ROM characterization **4-11**
- target reset
 - running from **3-9**
- target reset, running from **3-8**
- target system
 - dependency on executing code **4-6**
 - interface **3-10**
- Target system probe
 - pin guard **3-2**
- terminal interface **2-10, 5-7**
- trace, changing the trace depth **2-33**
- trace, displaying the **2-29**
- trace, displaying with time count absolute **2-31**
- trace, displaying with compress mode **2-32**
- tracing background operation **4-21**
- tracing bus cycles in HOLD state **4-21**
- tracing internal DMA cycles **4-21**
- tracing refresh cycles **4-22**
- transfer address, running from **2-20**
- trigger state **2-29**
- trigger, specifying **2-28**
- U** UEE_BRK_FLAG, foreground monitor label **A-7**
- user (target) memory
 - loading absolute files **2-11**
 - using the emulator **5-1**
- W** wait states, allowing the target system to insert **4-18**
- window systems **2-35**
- write to ROM break **4-20**