HP 64791/2

70208H/70216H Emulator PC Interface

User's Guide

HEWLETT PACKARD

HP Part No. 64791-97010
Printed in U.S.A.
July 1994

Edition 4

Notice

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

Printing History

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

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

Edition 1	64791-97001, August 1991
Edition 2	64791-97004, November 1991
Edition 3	64791-97007, December 1993
Edition 4	**64791-97010, July 1994**

Using this Manual

This manual covers the following emulators as used with the PC 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 70208,70216,70208H 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:

- 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 PC Interface command and option. The PC Interface is described in the *HP 64700 Emulator's PC Interface: User's Reference.*

Organization

Chapter 1 **"Introduction"**-This chapter lists the 70216 emulator features and describes how they can help you in developing new hardware and software.

Chapter 2 **"Getting Started"**-This chapter shows you how to use emulation commands by executing them on a sample program. This chapter describes the sample program and how to:

- load programs into the emulator
- map memory
- display and modify memory
- display registers
- step through programs
- run programs
- set software breakpoints
- search memory for data
- use the analyzer

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

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

Chapter 5 **"Using the Emulator"**-This chapter describes emulation topics that are not covered in the "Getting Started" chapter (for example, coordinated measurements and storing memory).

Appendix A. **"File Format Reader"**-This appendix describes how to use the File Format Reader from MS-DOS or PC Interface, load absolute files into the emulator, use global and local symbols with the PC Interface.

Contents

Illustrations

Tables

Notes

1

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.

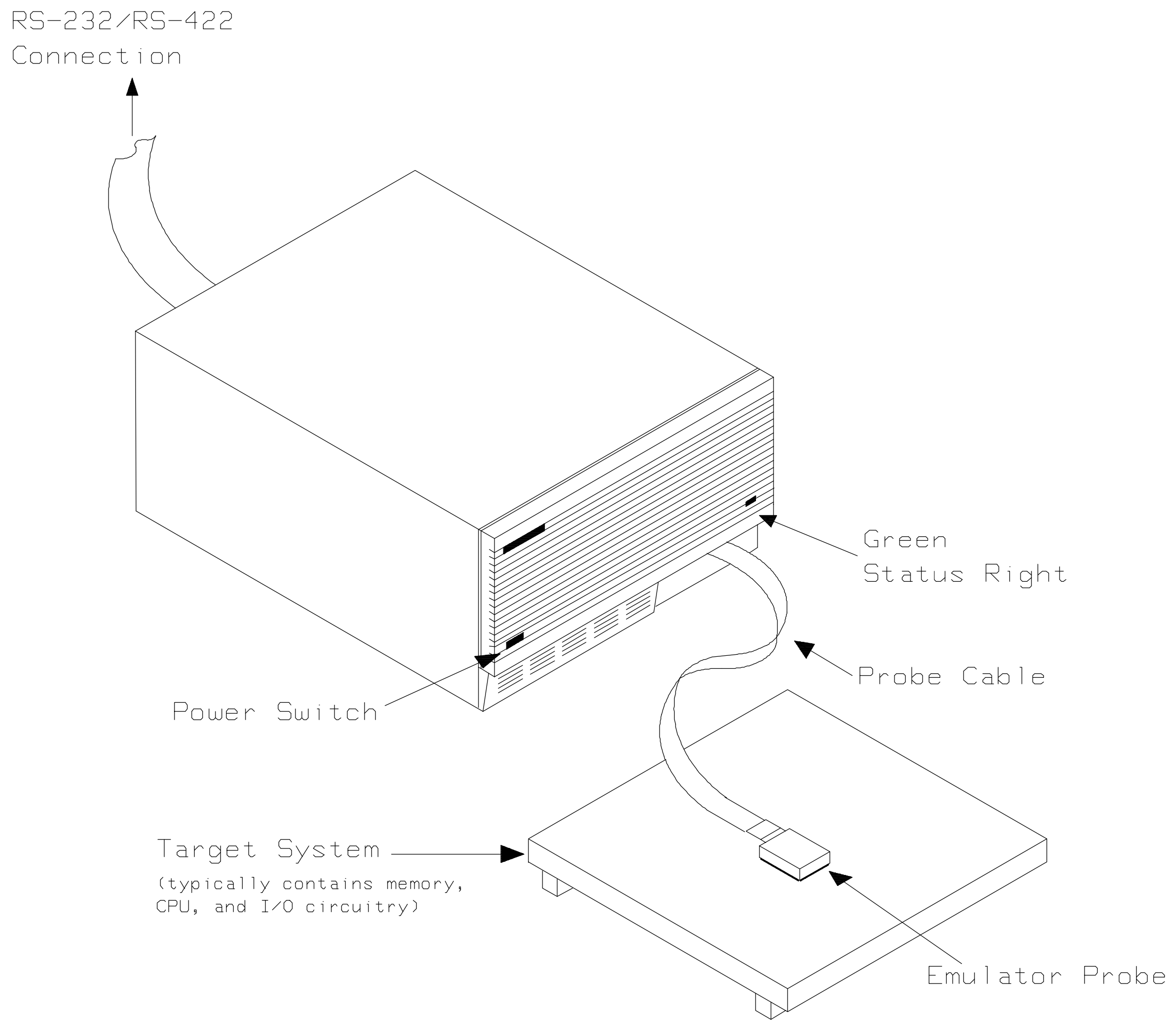


Figure 1-1. HP 64792 Emulator for uPD70216

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

2

Getting Started

Introduction

This chapter leads you through a basic tutorial that shows how to use the HP 64792 emulators for the 70216 microprocessors with the PC Interface.

This chapter will:

- Tell you what to do before you use the emulator in the tutorial.
- Describe the sample program used for this chapter’s examples.
- Briefly describe how to enter PC Interface commands and how emulator status is displayed.

This chapter will show you how to:

- Start up the PC Interface from the MS-DOS prompt.
- Define (map) emulation and target system memory.
- Load programs into emulation and target system memory.
- Enter emulation commands to view sample program execution.

Before You Begin

Prerequisites

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

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

2. Installed the PC Interface software on your computer. Software installation instructions are shipped with the media containing the PC Interface software. The *HP 64700 Emulators PC Interface: User's Reference* manual contains additional information on the installation and setup of the PC Interface.

3. In addition, it is recommended, although not required, that you read and understand the concepts of emulation presented in the *Concepts of Emulation and Analysis* manual. The *Installation/Service* also covers HP 64700 Series system architecture. A brief understanding of these concepts may help avoid questions later.

The sample program

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

We will show you how to use the emulator to:

- load this program into emulation memory
- execute the program
- monitor the program's operation with the analyzer
- simulate entry of different commands using the "**M**emory **M**odify" emulation command.

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                        1 "70116"
                        2                GLB     Msgs,Init,Cmd_Input,Msg_Dest
                        3
                        4                DATA
    0000                5 Msgs
    0000 436F6D6D61     6 Msg_A          DB      "Command A entered "
    0005 6E64204120
    000A 656E746572
    000F 656420
    0012 436F6D6D61     7 Msg_B          DB      "Command B entered "
    0017 6E64204220
    001C 656E746572
    0021 656420
    0024 496E76616C     8 Msg_I          DB      "Invalid Command   "
    0029 696420436F
    002E 6D6D616E64
    0033 202020
    0036                9 End_Msgs
                       10
                       11                PROG
                       12                ASSUME  DS0:DATA,DS1:COMN
                       13 *****************************************************
                       14 * The following instructions initialize segment
                       15 * regsiters and set up the stack pointer.
                       16 *****************************************************
    0000 B80000        17 Init           MOV     AW,SEG Msg_A
    0003 8ED8          18                MOV     DS0,AW
    0005 B80000        19                MOV     AW,SEG Cmd_Input
    0008 8EC0          20                MOV     DS1,AW
    000A 8ED0          21                MOV     SS,AW
    000C BC00F9        22                MOV     SP,OFFSET Stk
                       23 *****************************************************
                       24 * Clear previous command
                       25 *****************************************************
    000F 26C6060000    26 Rrad_Cmd       MOV     Cmd_Input,#0
    0014 0090
                       27 *****************************************************
                       28 * Read command input byte.  If no command has been
                       29 * entered, continue to scan for command input.
                       30 *****************************************************
    0016 26A00000      31 Scan           MOV     AL,Cmd_Input
    001A 3C00          32                CMP     AL,#0
    001C 74F8          33                BE      Scan
                       34 *****************************************************
                       35 * A command has been entered.  Check if it is
                       36 * command A, command B, or invalid.
                       37 *****************************************************
    001E 3C41          38 Exe_Cmd        CMP     AL,#41H
    0020 7407          39                BE      Cmd_A
    0022 3C42          40                CMP     AL,#42H
    0024 740C          41                BE      Cmd_B
    0026 E91200        42                BR      Cmd_I
                       43 *****************************************************
                       44 * Command A is entered.  CW = the number of bytes in
                       45 * message A.  BP = location of the message.  Jump to
                       46 * the routine which writes the message.
```

Figure 2-1. Sample Program Listing

```
                    ******************************************************
0029 B91200         Cmd_A           MOV     CW,#Msg_B-Msg_A
002C BE0000                         MOV     IX,OFFSET Msg_A
002F E90F00                         BR      Write_Msg
                    ******************************************************
                    * Command B is entered.
                    ******************************************************
0032 B91200         Cmd_B           MOV     CW,#Msg_I-Msg_B
0035 BE0012                         MOV     IX,OFFSET Msg_B
0038 E90600                         BR      Write_Msg
                    ******************************************************
                    * An invalid command is entered.
                    ******************************************************
003B B91200         Cmd_I           MOV     CW,#End_Msgs-Msg_I
003E BE0024                         MOV     IX,OFFSET Msg_I
                    ******************************************************
                    * Message is written to the destination.
                    ******************************************************
0041 BF0001         Write_MSG       MOV     IY,OFFSET Msg_Dest
0044 F3A4                           REP MOVBKB
                    ******************************************************
                    * The rest of the destination area is filled
                    * with zeros.
                    ******************************************************
0046 C60500         Fill_Dest       MOV     BYTE PTR [IY],#0
0049 47                             INC     IY
004A 81FF0021                       CMP     IY,#Msg_Dest+20H
004E 75F6                           BNE     Fill_Dest
                    ******************************************************
                    * Go back and scan for next command
                    ******************************************************
0050 EBBD                           BR      Read_Cmd

                                    COMN
                    ******************************************************
                    * Command input byte.
                    ******************************************************
0000                Cmd_Input       DBS     1
                    ******************************************************
                    * Destination of the command message.
                    ******************************************************
0001                Msg_Dest        DDS     3EH
00F9                Stk             DWS     1       ; Stack area.
         <0000>                     END     Init
```

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

Data Declarations

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

Initialization

The program instructions from the **Init** label to the **Read_Cmd** label perform initialization. The segment registers are loaded and the stack pointer is set up.

Reading Input

The instruction at the **Read_Cmd** label clears any random data or previous commands from the **Cmd_Input** byte. The **Scan** loop continually reads the **Cmd_Input** byte to look for a command (a value other than 0 hex).

Processing Commands

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

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

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

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

The instructions at **Cmd_A**, **Cmd_B**, and **Cmd_I** each load register CW with the displayed message's length and register IX with the message's starting location. Then, execution transfers to **Write_Msg**, which writes the appropriate message to the destination location, **Msg_Dest**.

After the message is written, the instructions at **Fill_Dest** fill the remaining destination locations with zeros. (The entire destination area is 20 hex bytes long.) Then, the program jumps back to read the next command.

The Destination Area

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

Assembling and Linking the Sample Program

The sample program is written for the HP 64853 Cross Assembler/Linker.

Use the following command to assemble and link the sample program.

```
C> asm -oe cmd_rds.s > cmd_rds.o <RETURN>


C> lnk -o > cmd_rds.m <RETURN>

object files  cmd_rds.R <RETURN>
library files   <RETURN>
Load addresses: PROG,DATA,COMN 400H,600H,800H <RETURN>
more files (y or n)  N <RETURN>
absolute file name  cmd_rds.X <RETURN>
```

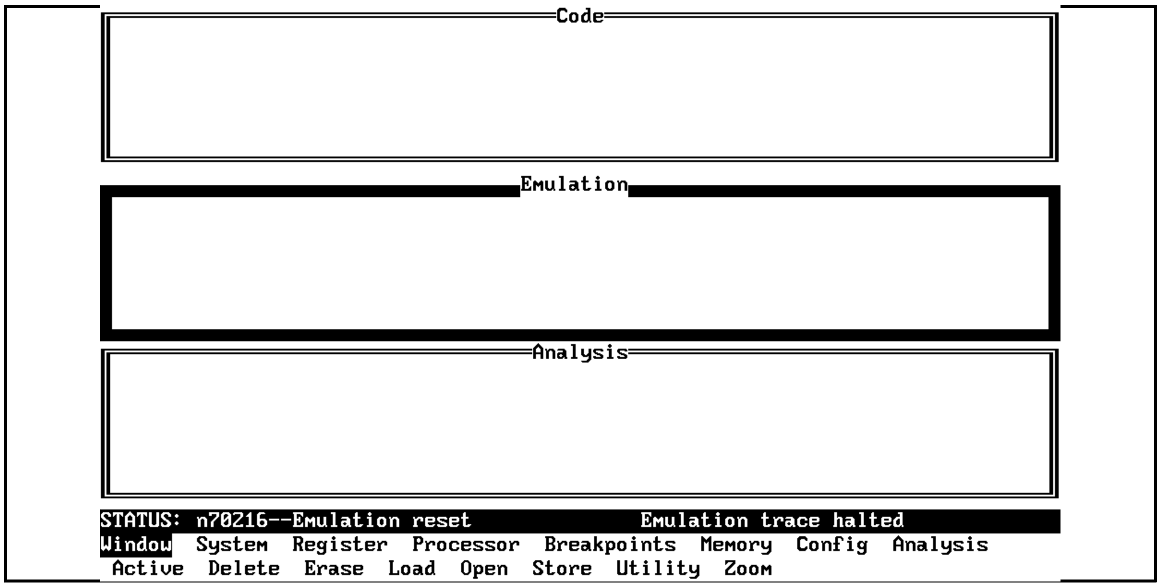
Starting Up the 70216 PC Interface

If you built the emulator device table and set the **HPTABLES** shell environment variable as shown in the *HP 64700 Emulators PC Interface: User's Reference*, you can start up the 70216 PC Interface by entering the following command from the MS-DOS prompt:

```
    C> pcv50 <emulname>
```

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

In the command above, **pcv50** is the command to start the PC Interface; "<emulname>" is the logical emulator name given in the emulator device table. (To start the version of the PC Interface that supports external timing analysis, substitute **ptv50** for **pcv50** in this command.) If this command is successful, you will see the display shown in figure 2-2. Otherwise, you will see an error message and return to the MS-DOS prompt.



```
STATUS: n70216--Emulation reset            Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Active  Delete  Erase  Load  Open  Store  Utility  Zoom
```

Figure 2-2. PC Interface Display

Selecting PC Interface Commands

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

When a command or option is highlighted, the bottom line of the display shows the next level of options or a short message describing the current option.

Emulator Status

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

Mapping Memory

The 70216 emulator contains high-speed emulation memory (no wait states required) that can be mapped at a resolution of 128 bytes.

Note

When you use the 8087 coprocessor on your target system connected to 70216 microprocessor, the 8087 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 8087 coprocessor before starting emulation session. Refer to "In-Circuit Emulation Topics" chapter for more information about accesses to emulation memory.

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.

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 "Enable breaks on writes to ROM?" configuration item is enabled (see the "Configuring the Emulator" chapter).
The memory mapper allows you to define up to 16 different map terms.

Which Memory Locations Should Be Mapped?

Typically, assemblers generate relocatable files and linkers combine relocatable files to form the absolute file. The linker load map listing will show what locations your program will occupy in memory. For example, the HP 64853 linker load map listing for the sample program is shown in figure 2-3.

```
HP 64000+ Linker


FILE/PROG NAME             PROGRAM    DATA       COMMON     ABSOLUTE
----------------------------------------------------------------------
CMD_RDS.R                  00000400   00000600   00000800

next address               00000452   00000636   000008FB
XFER address = 00000400  Defined by CMD_RDS.R
Absolute file name = CMD_RDS.X
Total number of bytes loaded = 00000183
```

Figure 2-3. Sample Program Load Map Listing

From the load map listing, you can see that the sample program occupies locations in three address ranges. The program area, which contains the opcodes and operands which make up the sample program, occupies locations 400 hex through 451 hex. The data area, which contains the ASCII values of the messages the program displays, is occupies locations 600 hex through 635 hex. The destination area, which contains the command input byte and the locations of the message destination and the stack, occupies locations 800 hex through 8FA hex.

Two mapper terms will be specified for the example program. Since the program writes to the destination locations, the mapper block containing the destination locations should not be characterized as ROM memory.

To map memory for the sample program, select:

```
Config, Map, Modify
```

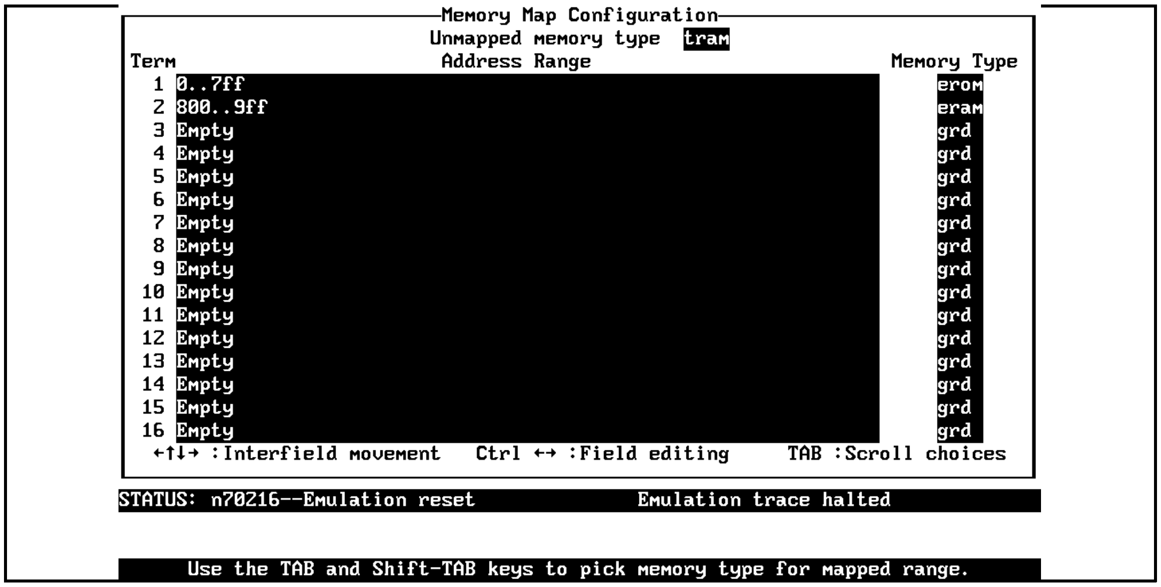
Using the arrow keys, move the cursor to the "address range" field of term 1. Enter:

```
0..07ff
```

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

```
0800..09ff
```

Move the cursor to the "memory type" field of term 2, and press the TAB key to select the **eram** (emulation RAM) type. To save your memory map, use the right arrow key or the **Enter** key to exit the field in the lower right corner. (The **End** key on Vectra keyboards moves the cursor directly to the last field.) The memory configuration display is shown in figure 2-4.



```
                          Memory Map Configuration
                         Unmapped memory type  tram
Term                     Address Range                          Memory Type
  1 0..7ff                                                        erom
  2 800..9ff                                                      eram
  3 Empty                                                         grd
  4 Empty                                                         grd
  5 Empty                                                         grd
  6 Empty                                                         grd
  7 Empty                                                         grd
  8 Empty                                                         grd
  9 Empty                                                         grd
 10 Empty                                                         grd
 11 Empty                                                         grd
 12 Empty                                                         grd
 13 Empty                                                         grd
 14 Empty                                                         grd
 15 Empty                                                         grd
 16 Empty                                                         grd
   ←↑↓→ :Interfield movement   Ctrl ←→ :Field editing     TAB :Scroll choices

STATUS: n70216--Emulation reset              Emulation trace halted


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

Figure 2-4. Memory Map Configuration

For your programs (not the sample), you may want to map emulation memory locations containing programs and constants (locations that should not be written to) as ROM. This will prevent programs and constants from being written over accidentally, and will cause breaks when instructions attempt to do so.

Note

The memory mapper reassigns blocks of emulation memory after the insertion or deletion of mapper terms. Suppose you modified the contents of 400H-7FFH above, deleted term 1, then displayed locations 400H-7FFH. You'll notice the contents of those locations differ before and after you delete the mapper term.

Loading Programs into Memory

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

```
Memory, Load
```

File Format

Use **Tab** and **Shift-Tab** to select the format of your absolute file. The emulator accepts absolute files in the following formats:

- Intel OMF86 absolute.
- NEC30 absolute.
  - (This absolute file is generated by NEC LK70116 linker for uPD70208 and uPD70216.)
- HP64000 absolute.
- Raw HP64000 absolute.
- Intel hexadecimal.
- Motorola S-records.
- Tektronix hexadecimal.

For this tutorial, choose the HP64000 format.

Target Memory Type for Memory Load

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

Since emulation memory is mapped for sample program locations, you can select either "emulation" or "both". Use **Tab** key and **Shift-Tab** to cycle through the choices.

Force the Absolute File to Be Read

This option is only available for the Intel OMF86, NEC30, and HP64000 absolute file formats.

It forces the file format reader to regenerate the emulator absolute file (.hpa) and symbol database (.hps) before loading the code. Normally, these files are only regenerated whenever the file you specify (the output of your language tools) is newer than the emulator absolute file and symbol database.

For more information, refer to the "File Format Readers" appendix.

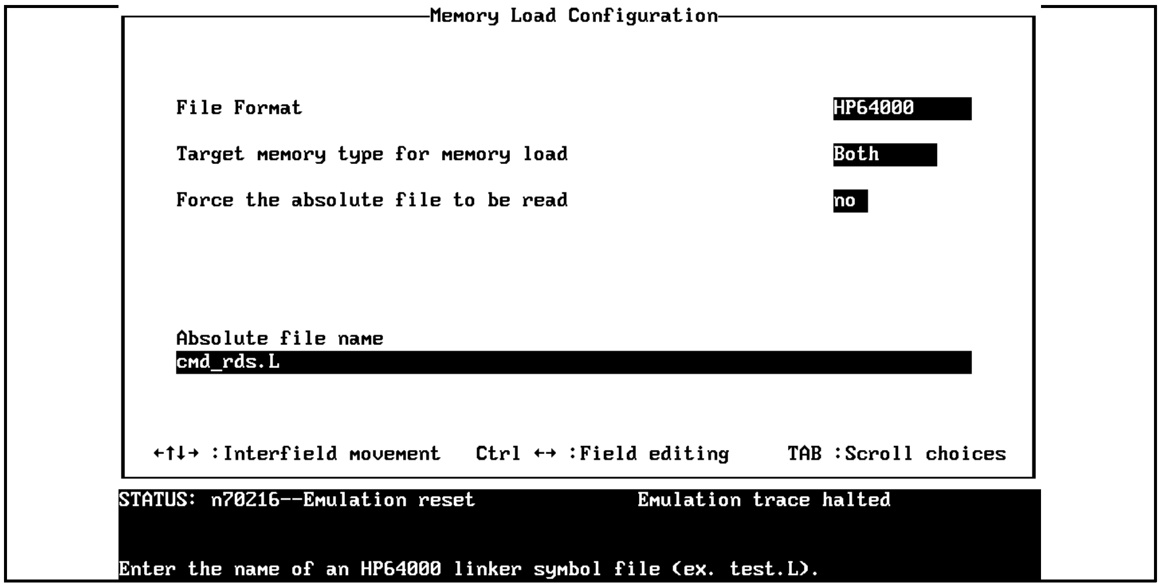
File Format Options

Some of the formats, such as the Intel OMF86 format, have special options.

Refer to the "File Format Readers" appendix of this manual for more information.

Absolute File Name

For most formats, you enter the name of your absolute file in the last field. The HP64000 format requires the linker symbol filename instead. Type **cmd_rds.l**, and press **Enter** to start the memory load.



```
Memory Load Configuration

File Format                                   HP64000
Target memory type for memory load            Both
Force the absolute file to be read            no

Absolute file name
cmd_rds.L

←↑↓→ :Interfield movement   Ctrl ←→ :Field editing    TAB :Scroll choices
STATUS: n70216--Emulation reset           Emulation trace halted

Enter the name of an HP64000 linker symbol file (ex. test.L).
```

Displaying Symbols

Symbol files are created when you generate absolute files with the HP 64000-PC Cross Assembler/Linkers. When you assemble a source file, an assembler symbol file (with the same base name as the source file and a “.a” extension) is created. The assembler symbol file contains local symbol information. When you link relocatable assembly modules, a linker symbol file (with the same base name as the absolute file and a “.l” extension) is created. The linker symbol file contains global symbol information and information about the relocatable assembly modules that combine to form the absolute file.

When you load a file using the HP64000 file format, the file format reader collects global symbol information from the linker symbol file and local symbol information from the assembler symbol files. It uses this information to create a single symbol database with the extension .hps.

If you load a file using the following formats, the file format reader obtains all the global and local symbol information from the absolute file and builds a symbol database with extension .hps.

- Intel OMF86 absolute.
- NEC30 absolute.

The following pages show you how to display global and local symbols for the sample program. For more information on symbol display, refer to the *PC Interface Reference*.

Displaying Global Symbols

When you load a file using the following formats into the emulator, the corresponding symbol database is also loaded.

- Intel OMF86 absolute.
- NEC30 absolute.
- HP64000 absolute.

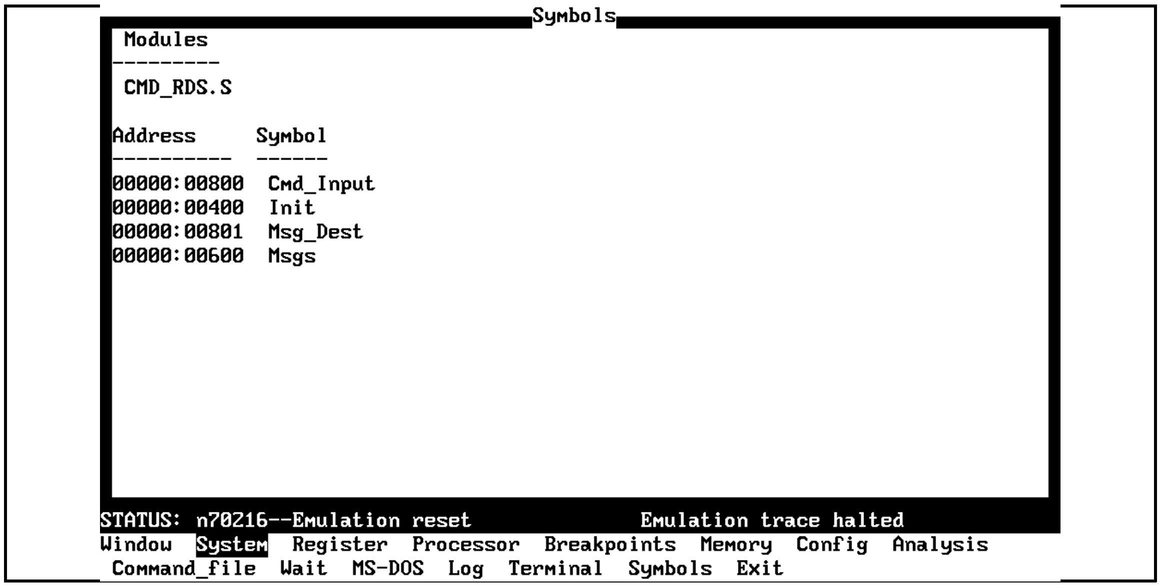
The symbol database also can be loaded with the "**S**ystem, **S**ymbols, **G**lobal, **L**oad" command. Use this command when you load multiple absolute files into the emulator. You can load the various symbol databases corresponding to each absolute file. When you load a symbol database, information from a previous symbol database is lost. That is, only one symbol database can be present at a time.

After a symbol database is loaded, both global and local symbols can be used when entering expressions. You enter global symbols as they appear in the source file or in the global symbols display.

To display global symbols, select:

System, **S**ymbols, **G**lobal, **D**isplay

The symbols window automatically becomes the active window because of this command. You can press <CTRL>**z** to zoom the window. The resulting display follows.



```
                                  Symbols
 Modules
---------
 CMD_RDS.S

Address      Symbol
----------   ------
00000:00800   Cmd_Input
00000:00400   Init
00000:00801   Msg_Dest
00000:00600   Msgs

STATUS:  n70216--Emulation reset              Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Command_file  Wait  MS-DOS  Log  Terminal  Symbols  Exit
```

The global symbols display has two parts. The first part lists all the modules that were linked to produce this object file. These module names are used by you when you want to refer to a local symbol, and are case-sensitive. The second part of the display lists all global symbols in this module. These names can be used in measurement specifications, and are case-sensitive. For example, if you wish to make a measurement using the symbol **Cmd_Input**, you must specify **Cmd_Input**.

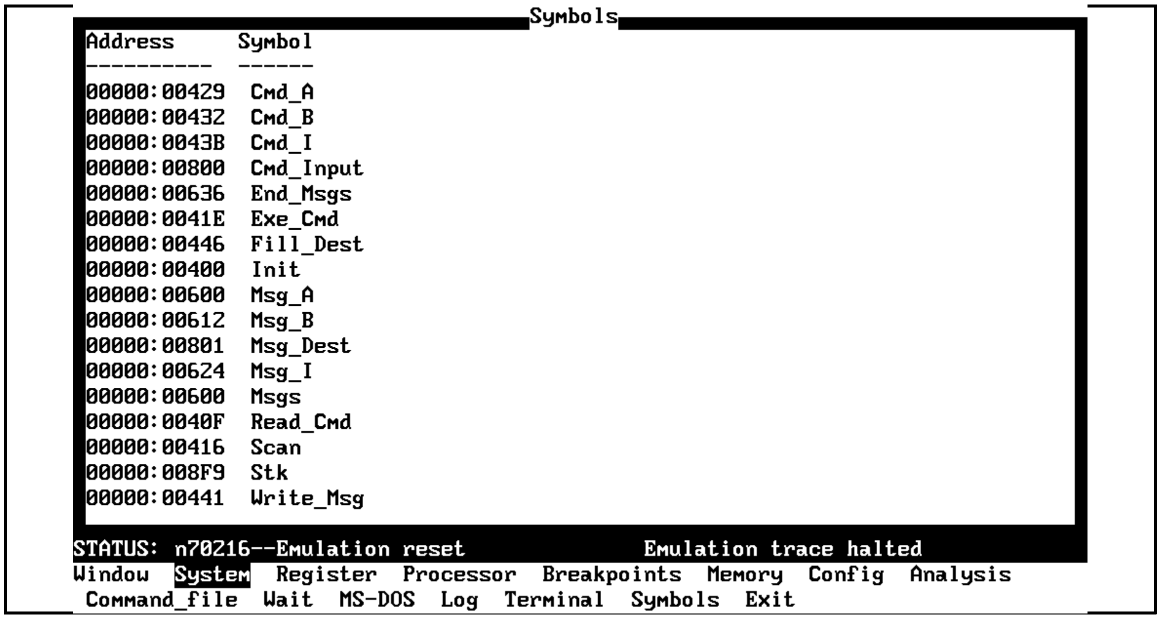
The strings **cmd_input** and **CMD_INPUT** are not valid symbol names here.

Loading and Displaying Local Symbols

To display local symbols, select:

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

Enter the name of the module you want to display (from the first part of the global symbols list; in this case, **CMD_RDS.S**) and press **Enter**. The resulting display follows.



```
                                   Symbols
Address     Symbol
----------  ------
00000:00429  Cmd_A
00000:00432  Cmd_B
00000:0043B  Cmd_I
00000:00800  Cmd_Input
00000:00636  End_Msgs
00000:0041E  Exe_Cmd
00000:00446  Fill_Dest
00000:00400  Init
00000:00600  Msg_A
00000:00612  Msg_B
00000:00801  Msg_Dest
00000:00624  Msg_I
00000:00600  Msgs
00000:0040F  Read_Cmd
00000:00416  Scan
00000:008F9  Stk
00000:00441  Write_Msg

STATUS:  n70216--Emulation reset              Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Command_file  Wait  MS-DOS  Log  Terminal  Symbols  Exit
```

After you display local symbols with the “**S**ystem **S**ymbols **L**ocal **D**isplay” command, you can enter local symbols as they appear in the source file or local symbol display. When you display local symbols for a given module, that module becomes the default local symbol module.

If you have not displayed local symbols, you can still enter a local symbol by including the name of the module:

```
module_name:symbol
```

Remember that the only valid module names are those listed in the first part of the global symbols display, and are case-sensitive for compatibility with other systems (such as HP-UX).

When you include the name of an source file with a local symbol, that module becomes the default local symbol module, as with the "**S**ystem **S**ymbols **L**ocal **D**isplay" command.

Local symbols must be from assembly modules that form the absolute whose symbol database is currently loaded. Otherwise, no symbols will be found (even if the named assembler symbol file exists and contains information).

One thing to note: It is possible for a symbol to be local in one module and global in another, which may result in some confusion. For example, suppose symbol "XYZ" is a global in module A and a local in module B and that these modules link to form the absolute file. After you load the absolute file (and the corresponding symbol database), entering "XYZ" in an expression refers to the symbol from module A. Then, if you display local symbols from module B, entering "XYZ" in an expression refers to the symbol from module B, **not the global symbol**. Now, if you again want to enter "XYZ" to refer to the global symbol from module A, you must display the local symbols from module A (since the global symbol is also local to that module). Loading local symbols from a third module, if it was linked with modules A and B and did not contain an "XYZ" local symbol, would also cause "XYZ" to refer to the global symbol from module A.

Transfer Symbols to the Emulator

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

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

Transfer the local symbol information for all modules by entering:

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

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

Changing the Disassembler Mode

The emulator has two sets of syntax to display memory contents or trace listing in mnemonic format.

- HP64853 Cross Assembler.
- NEC Assembler.

The disassembler mode allow you to select which syntax the disassembler should use in mnemonic memory, trace, and register displays. The default disassembler mode selects NEC assembler syntax.

Before getting into the main emulation session, you may change the disassembler mode to select the HP64853 syntax because it is suitable for your language tool in this chapter.

To change the disassembler mode, select:

```
    Config, General
```

Use the arrow key to move the cursor to the "Disassembler Mode" field, and use **TAB** key to select "**64853**". Press **End** and **Enter** consecutively to exit the configuration.

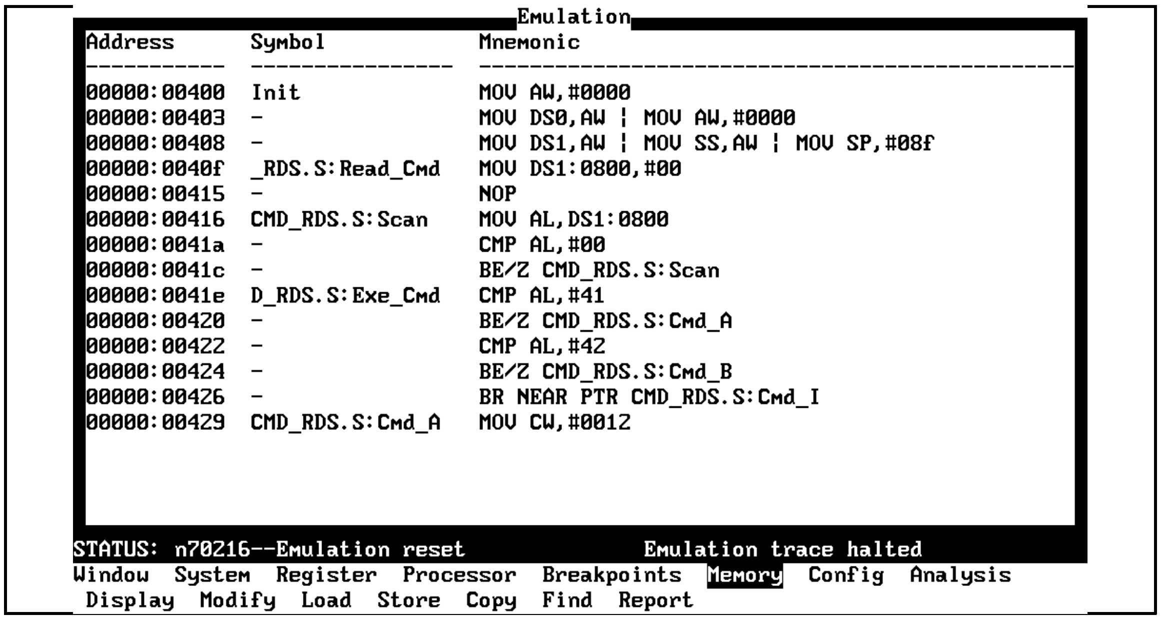
Displaying Memory in Mnemonic Format

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

```
Memory, Display, Mnemonic
```

Enter the address range "400H..429H". You could also specify this address range using symbols.
For example,
"**Init..Cmd_A**" or "**Init..Init+29H**".
The Emulation window remains active. You can press <CTRL>**z** to zoom the memory window. The resulting display follows.

If you want to see the rest of the sample program memory locations, you can select "**M**emory, **D**isplay, **M**nemonic" command and enter the range from 42AH to 451H.



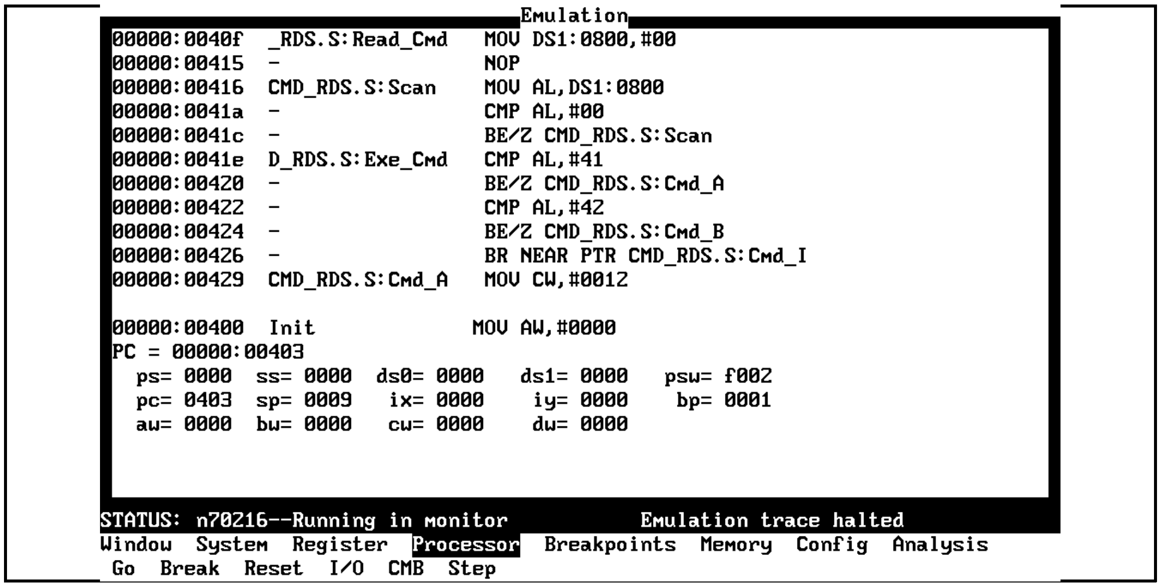
```
                                   Emulation
Address      Symbol            Mnemonic
-----------  ----------------  ----------------------------------------------
00000:00400  Init              MOV AW,#0000
00000:00403  -                 MOV DS0,AW ¦ MOV AW,#0000
00000:00408  -                 MOV DS1,AW ¦ MOV SS,AW ¦ MOV SP,#08f
00000:0040f  _RDS.S:Read_Cmd   MOV DS1:0800,#00
00000:00415  -                 NOP
00000:00416  CMD_RDS.S:Scan    MOV AL,DS1:0800
00000:0041a  -                 CMP AL,#00
00000:0041c  -                 BE/Z CMD_RDS.S:Scan
00000:0041e  D_RDS.S:Exe_Cmd   CMP AL,#41
00000:00420  -                 BE/Z CMD_RDS.S:Cmd_A
00000:00422  -                 CMP AL,#42
00000:00424  -                 BE/Z CMD_RDS.S:Cmd_B
00000:00426  -                 BR NEAR PTR CMD_RDS.S:Cmd_I
00000:00429  CMD_RDS.S:Cmd_A   MOV CW,#0012

STATUS: n70216--Emulation reset              Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Display  Modify  Load  Store  Copy  Find  Report
```

Stepping Through the Program

The emulator allows you to execute one instruction or a number of instructions with step command. To begin stepping through the sample program, select:

```
Processor, Step, Address
```

Enter a step count of 1, enter the symbol **Init** (defined as a global in the source file), and press **Enter** to step from program's first address, 400H. The Emulation window remains active. Press <CTRL>**z** to view a full screen of information. The executed instruction, the program counter address (PS:PC), and the resulting register contents are displayed as shown in the following.



```
                                   Emulation
00000:0040f  _RDS.S:Read_Cmd   MOV DS1:0800,#00
00000:00415  -                 NOP
00000:00416  CMD_RDS.S:Scan    MOV AL,DS1:0800
00000:0041a  -                 CMP AL,#00
00000:0041c  -                 BE/Z CMD_RDS.S:Scan
00000:0041e  D_RDS.S:Exe_Cmd   CMP AL,#41
00000:00420  -                 BE/Z CMD_RDS.S:Cmd_A
00000:00422  -                 CMP AL,#42
00000:00424  -                 BE/Z CMD_RDS.S:Cmd_B
00000:00426  -                 BR NEAR PTR CMD_RDS.S:Cmd_I
00000:00429  CMD_RDS.S:Cmd_A   MOV CW,#0012

00000:00400  Init             MOV AW,#0000
PC = 00000:00403
  ps= 0000  ss= 0000  ds0= 0000   ds1= 0000   psw= f002
  pc= 0403  sp= 0009   ix= 0000    iy= 0000    bp= 0001
  aw= 0000  bw= 0000   cw= 0000    dw= 0000

STATUS: n70216--Running in monitor           Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Go  Break  Reset  I/O  CMB  Step
```

Note

You cannot display registers if the processor is reset.
Use the "**P**rocessor **B**reak" command to cause the emulator to start executing in the monitor.

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

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, BUSLOCK.
3) EI, RETI, DI.

To continue stepping through the program, you can select:

```
Processor, Step, Pc
```

After selecting this command, you can change the previous step count. If you wish to step the same number of times, just press **Enter** to start the step.

To save time when single-stepping, you can use the function key macro <F1>, which executes the command:

```
Processor, Step, Pc, 1
```

For more information, see the *Emulator PC Interface Reference* chapter on Function Key Macros.

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

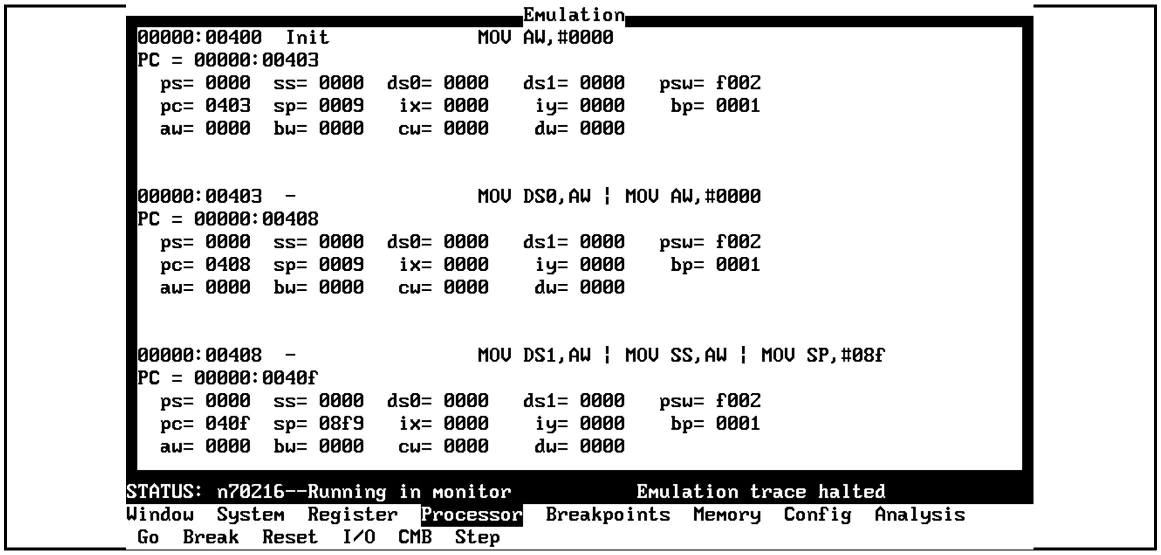
Specifying a Step Count

If you want to step sevral times from the current program counter, select:

```
Processor, Step, Pc
```

The previous step count is displayed in the "number of instructions" field. You can enter a number from 1 through 99 to specify the number times to step. Type 5 into the field, and press **Enter**. The resulting display follows.

When you specify step counts greater than 1, only the last instruction and the register contents after that instruction are displayed.



```
                                        Emulation
00000:00400  Init            MOV AW,#0000
PC = 00000:00403
  ps= 0000  ss= 0000  ds0= 0000   ds1= 0000   psw= f002
  pc= 0403  sp= 0009   ix= 0000    iy= 0000    bp= 0001
  aw= 0000  bw= 0000   cw= 0000    dw= 0000


00000:00403  -               MOV DS0,AW ¦ MOV AW,#0000
PC = 00000:00408
  ps= 0000  ss= 0000  ds0= 0000   ds1= 0000   psw= f002
  pc= 0408  sp= 0009   ix= 0000    iy= 0000    bp= 0001
  aw= 0000  bw= 0000   cw= 0000    dw= 0000


00000:00408  -               MOV DS1,AW ¦ MOV SS,AW ¦ MOV SP,#08f
PC = 00000:0040f
  ps= 0000  ss= 0000  ds0= 0000   ds1= 0000   psw= f002
  pc= 040f  sp= 08f9   ix= 0000    iy= 0000    bp= 0001
  aw= 0000  bw= 0000   cw= 0000    dw= 0000
STATUS: n70216--Running in monitor           Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Go  Break  Reset  I/O  CMB  Step
```

Modifying Memory

The preceding step commands show the sample program is executing in the **Scan** loop, where it continually reads the command input byte to look for a command.

To simulate the entry of a sample program command, you can modify the command input byte by selecting:

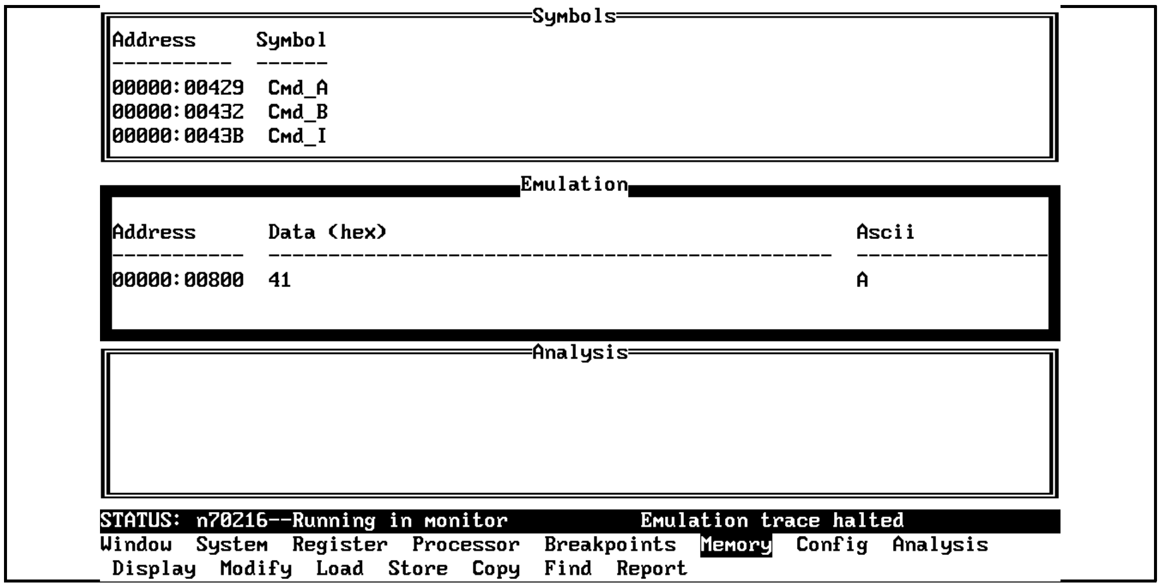
```
Memory, Modify, Byte
```

Now enter the address of the memory location to be modified, an equal sign, and new value of that location, for example, **Cmd_Input="A"**. (The **Cmd_Input** label was defined as a global symbol in the source file.)

To verify that "A" was indeed written to **Cmd_Input** (800 hex), select:

```
Memory, Display, Byte
```

Type the address 800H or the symbol **Cmd_Input**, and press **Enter**. The resulting display is shown below.



```
                              Symbols
Address     Symbol
----------  ------
00000:00429  Cmd_A
00000:00432  Cmd_B
00000:0043B  Cmd_I

                              Emulation
Address      Data (hex)                                        Ascii
-----------  ------------------------------------------------  ---------------
00000:00800  41                                                A

                              Analysis

STATUS: n70216--Running in monitor           Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Display  Modify  Load  Store  Copy  Find  Report
```

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

Running the Program

To start the sample program, select:

```
Processor, Go, Pc
```

The status line will show that the emulator is "Running user program".

Searching Memory for Data

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

```
Memory, Find
```

Enter the range of the memory locations to be searched, "800H..820H", and enter the data " **A** " or 20H, 41H, and 20H. The resulting information in the Emulation window shows you that the message write occurred correctly. The message is:

```
Pattern match at address: 0000808
```

To verify that the sample program works for the other allowed commands, you can modify the command input byte to "B" and search for " B " (20 hex, 42 hex, and 20 hex), or you can modify the command input byte to "C" and search for "d C" (64 hex, 20 hex, and 43 hex).

Breaking into the Monitor

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

```
Processor, Break
```

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

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

Using Software Breakpoints

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 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 breakpint 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 brekpoint, execution breaks to the monitor, and the brekpoint 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 geneated by a BRK 3 instruction in the target system, execution still breaks to the monitor, and an "Undefined software breakpoint" message is displayed.

Note

Because software brekpoints are implemented by the replacing opcodes with the brekpoint 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 brekpoints. 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 only set software breakpoints at memory locations which contain instruction opcodes (not operands or data). If a software breakpoint is set at a memory location which is not an instruction opcode, the software breakpoint instruction will never be executed and the break will never occur.

Note

NMI will be ignored, when software breakpoint and NMI occur at the same time.

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

Software breakpoint will be ignored, when software breakpoint and other emulation break (for example, break command, trigger command, etc.) occur at the same time. Refer to *PC Interface: User's Reference* manual.

Defining a Software Breakpoint

To define a breakpoint at the address of the **Cmd_I** label of the sample program (43B hex), select:

```
Breakpoints, Add
```

Enter the local symbol "Cmd_I". After the breakpoint is added, the Emulation window becomes active and shows that the breakpoint is set.

You can add multiple breakpoints in a single command by separating them with a semicolon. For example, you could type "**2010h;2018h;2052h**" to set three breakpoints.

Run the program by selecting:

```
Processor, Go, Pc
```

The status line shows that the emulator is running the user program. Modify the command input byte to an invalid command by selecting:

```
Memory, Modify, Byte
```

Enter an invalid command, such as "Cmd_Input=75h". The following messages result:

```
ALERT:  Software breakpoint: 00000:0043b
STATUS: Running in monitor
```

To continue program execution, select:

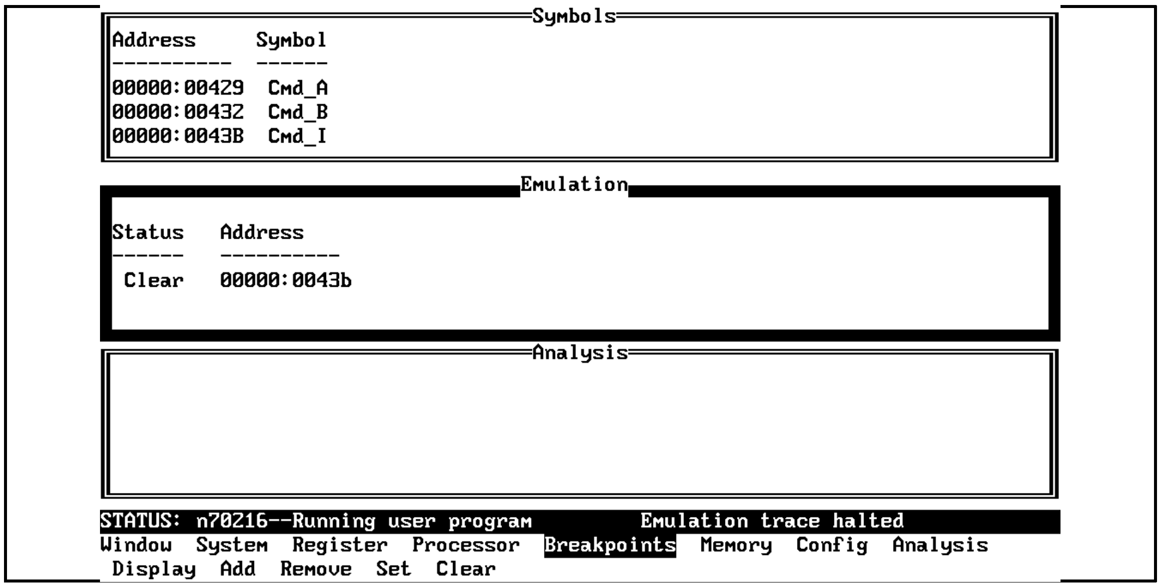
```
Processor, Go, Pc
```

Displaying Software Breakpoints

To view the status of the breakpoint, select:

```
Breakpoints, Display
```

The display shows that the breakpoint was cleared.


```
Symbols
Address     Symbol
----------  ------
00000:00429  Cmd_A
00000:00432  Cmd_B
00000:0043B  Cmd_I

Emulation
Status   Address
------   ----------
 Clear   00000:0043b

Analysis

STATUS: n70216--Running user program         Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Display  Add  Remove  Set  Clear
```

Setting a Software Breakpoint

A breakpoint is disabled when it is hit. To re-enable the software breakpoint, you can select:

```
Breakpoints, Set, Single
```

The address of the breakpoint you just added is still in the address field. To set this breakpoint again, press **Enter**.

As with the "**B**reakpoints **A**dd" command, the Emulation window becomes active and shows that the breakpoint is set.

Clearing a Software Breakpoint

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

```
Breakpoints, Clear, Single
```

The address of the breakpoint set in the previous section is still in the address field. To clear this breakpoint, press **Enter**.

Using the Analyzer

The analyzer collects data at each pulse of a clock signal, and saves the data (a trace state) if it meets a "storage qualification" condition.

Note

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

Resetting the Analysis Specification

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

```
Analysis, Trace, Reset
```

Specifying a Simple Trigger

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

```
Analysis, Trace, Modify
```

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

You'll enter the pattern expression menu. Press the up arrow key until the **addr** field directly opposite the pattern **a=** is highlighted. Type the address of the command input byte, using either the global symbol **Cmd_Input** or address 800H, and press **Enter**.

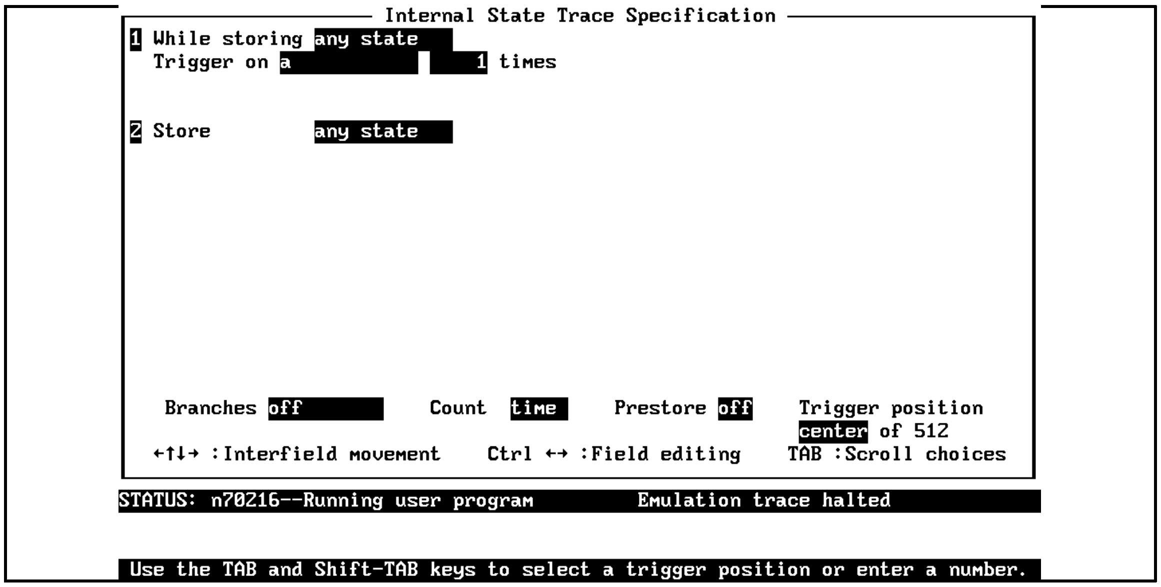
The "Data" field is now highlighted. Type 0XX42 and press **Enter**. "42" is the hexadecimal value of the "B" command and the "X"s specify "don't care" values. When 42H is read from the command input byte (800H), a lower byte read is performed because the address is even. If the address is odd, you must specify the data to 42XX.

Now the "Status" field is highlighted. Use the TAB key to view the status qualifier choices.

70216 Analysis Status Qualifiers

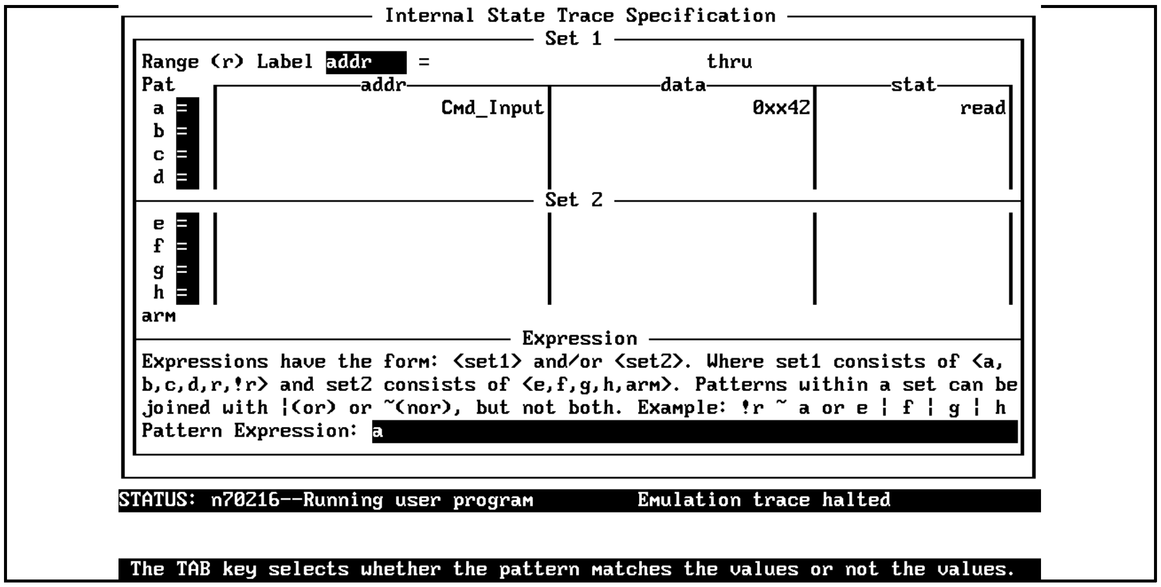
This trace command example uses the status qualifier "read". The following analysis status qualifiers also can be used with the 70216 emulator.

Qualifier	Status Bits	Description
exec	0xxx0xxxxxxxxxxxy	execute instruction
fetch	0xxx1xxxx001x100y	program fetch
read	0xxx1xxxxxx0xx01y	read
write	0xxx1xxxxxx0xx10y	write
mem	0xxx1xxxxxx0x1xxy	memory access
intio	0xxx1xxxx00000xxy	internal I/O access
extio	0xxx1xxxx00010xxy	external I/O access
cpu	0xxx1xxxx00xxxxxy	cpu cycle
dma	0xxx1xxxx10x01xxy	DMA memory access
casdma	0xxx1xxxx1010111y	cascaded DMA cycle
refresh	0xxx1xxxx0100101y	refresh cycle
holdack	0xxx1xxxx11xxxxxy	hold acknowledge
intack	0xxx1xxxx001x000y	interrupt acknowledge
haltack	0xxx1xxxxxxx1011y	halt acknowledge
em80	0xx1xxxxxxxxxxxxy	8080 emulation mode
native	0xx0xxxxxxxxxxxxy	native mode
ds0	0xxx1xx11xxxxxxxy	ds0 use cycle
ds1	0xxx1xx00xxxxxxxy	ds1 use cycle
ss	0xxx1xx01xxxxxxxy	ss use cycle
ps	0xxx1xx10xxxxxxxy	ps use cycle
rom	0xxx1x0xxxxxxxxxy	rom access
grd	0xxx10xxxxxxxxxxy	guarded memory access
usr	0x1xxxxxxxxxxxxxy	user cycle
mon	0x0xxxxxxxxxxxxxy	monitor cycle



```
                    Internal State Trace Specification
1 While storing any state
  Trigger on a                     1 times


2 Store         any state











   Branches off          Count  time     Prestore off    Trigger position
                                                         center of 512
  ←↑↓→ :Interfield movement    Ctrl ←→ :Field editing    TAB :Scroll choices
STATUS: n70216--Running user program         Emulation trace halted


 Use the TAB and Shift-TAB keys to select a trigger position or enter a number.
```

Figure 2-5. Modifying the Trace Specification



```
                    Internal State Trace Specification
                                   Set 1
 Range (r) Label addr   =                       thru
 Pat          ---------addr-----------------data-----------------stat------
   a =                      Cmd_Input                    0xx42            read
   b =
   c =
   d =
                                   Set 2
   e =
   f =
   g =
   h =
 arm
                                 Expression
 Expressions have the form: <set1> and/or <set2>. Where set1 consists of <a,
 b,c,d,r,!r> and set2 consists of <e,f,g,h,arm>. Patterns within a set can be
 joined with ¦(or) or ~(nor), but not both. Example: !r ~ a or e ¦ f ¦ g ¦ h
 Pattern Expression: a
STATUS: n70216--Running user program         Emulation trace halted


 The TAB key selects whether the pattern matches the values or not the values.
```

Figure 2-6. Modifying the Pattern Specification

Note

You can combine qualifiers to form more specific qualifiers. For example, the expression **memory&&read** matches only memory reads. See the *Emulator PC Interface Reference* for more information.

Select the **read** status and press **Enter**.

The resulting analysis specification is shown in figure 2-5. To save the new specification, use **End Enter** to exit the field in the lower right corner. You'll return to the trace specification. Press **End** to move the "trigger position" field. Use the TAB key until it says **center**, then press **Enter** to exit the trace specification.

Starting the Trace

To start the trace, select:

```
Analysis, Begin
```

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

```
Memory, Modify, Byte
```

Enter **Cmd_Input="B".** The status line now shows that the trace is complete. (If you have problems, you may be running in monitor. Select **P**rocessor **G**o **P**c to return to the user program.)

Change the Analyzer Display Format

If you have transferred the symbol database information to the emulator by entering the following commands:

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

you should change the display format to make better use of the trace display.

To change the analyzer display format, enter the command:

```
Analysis, Format
```

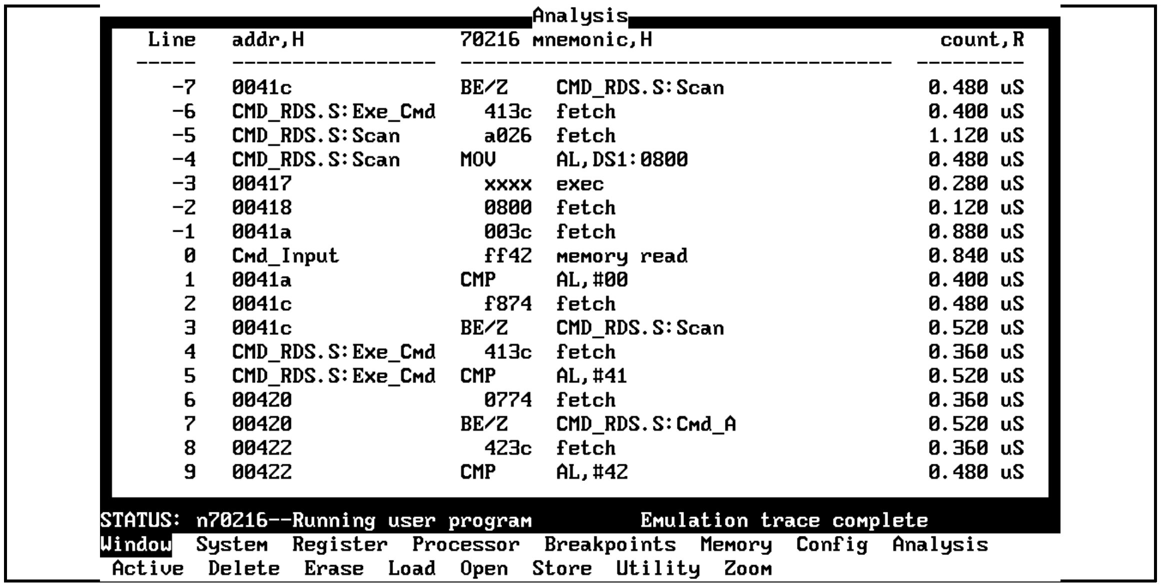
Use the down arrow key to move to the field labeled **addr**. And, use the right arrow key to move the field labeld **Width** above. The default width of the address column is six characters. A width of 17 characters is often wide enough to accommodate most symbol names. Type **17** to change the width of the address column, and press **End**, then **Enter.**

Displaying the Trace

To display the trace, select:

```
Analysis, Display
```

You are now given two fields in which to specify the states to display. Use the **End** key to move the cursor to the "Ending state to display" field. Type 60 into the press **Enter**. The resulting trace is similar to trace shown in the following display (use <CTRL>**z** to zoom the trace window). You may need to press the **Home** key to get to the top of the trace.

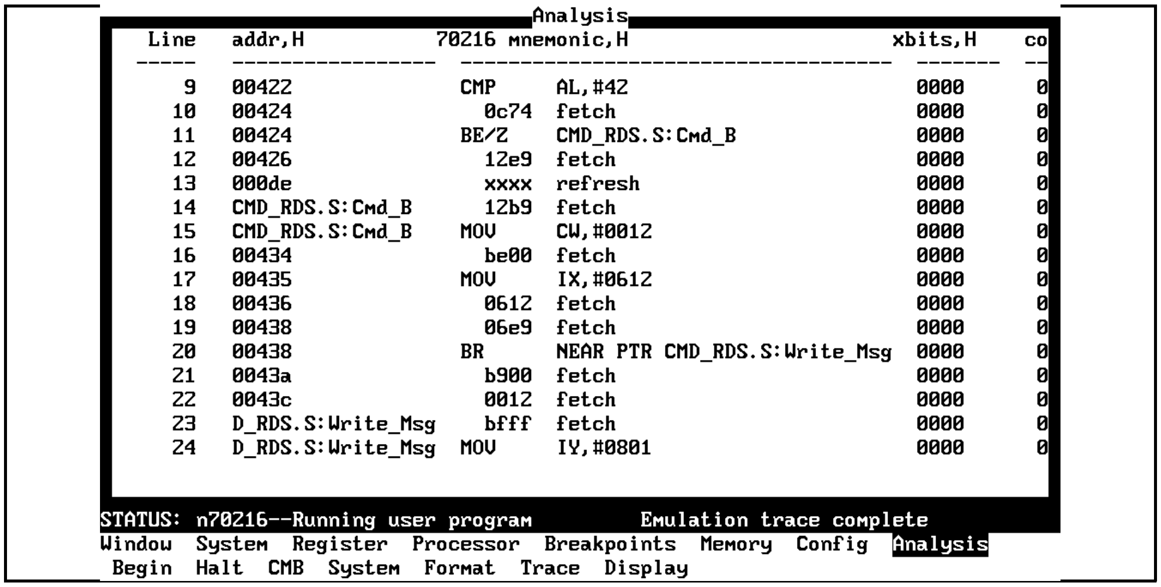


```
                                   Analysis
  Line   addr,H            70216 mnemonic,H                             count,R
 -----   -----------------  ------------------------------------  ---------
    -7   0041c             BE/Z     CMD_RDS.S:Scan                      0.480 uS
    -6   CMD_RDS.S:Exe_Cmd   413c   fetch                               0.400 uS
    -5   CMD_RDS.S:Scan      a026   fetch                               1.120 uS
    -4   CMD_RDS.S:Scan    MOV      AL,DS1:0800                         0.480 uS
    -3   00417               xxxx   exec                                0.280 uS
    -2   00418               0800   fetch                               0.120 uS
    -1   0041a               003c   fetch                               0.880 uS
     0   Cmd_Input           ff42   memory read                         0.840 uS
     1   0041a             CMP      AL,#00                              0.400 uS
     2   0041c               f874   fetch                               0.480 uS
     3   0041c             BE/Z     CMD_RDS.S:Scan                      0.520 uS
     4   CMD_RDS.S:Exe_Cmd   413c   fetch                               0.360 uS
     5   CMD_RDS.S:Exe_Cmd CMP      AL,#41                              0.520 uS
     6   00420               0774   fetch                               0.360 uS
     7   00420             BE/Z     CMD_RDS.S:Cmd_A                     0.520 uS
     8   00422               423c   fetch                               0.360 uS
     9   00422             CMP      AL,#42                              0.480 uS

STATUS:  n70216--Running user program         Emulation trace complete
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Active  Delete  Erase  Load  Open  Store  Utility  Zoom
```

Note

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

Line 0 in the above trace list shows the analyzer trigger state. The trigger state is always on line 0. The other states show the exit from the **Scan** loop and the **Exe_Cmd** instructions. Press the **PgDn** or **Next** key to see more lines of the trace. Notice that prefetches of instructions which do not get executed are included in the trace list.



```
                                   Analysis
  Line    addr,H           70216 mnemonic,H                             xbits,H    co
 -----    ----------------  ------------------------------------  -------   --
     9    00422             CMP      AL,#42                         0000      0
    10    00424               0c74   fetch                          0000      0
    11    00424             BE/Z     CMD_RDS.S:Cmd_B                0000      0
    12    00426               12e9   fetch                          0000      0
    13    000de               xxxx   refresh                        0000      0
    14    CMD_RDS.S:Cmd_B     12b9   fetch                          0000      0
    15    CMD_RDS.S:Cmd_B   MOV      CW,#0012                       0000      0
    16    00434               be00   fetch                          0000      0
    17    00435             MOV      IX,#0612                       0000      0
    18    00436               0612   fetch                          0000      0
    19    00438               06e9   fetch                          0000      0
    20    00438             BR       NEAR PTR CMD_RDS.S:Write_Msg   0000      0
    21    0043a               b900   fetch                          0000      0
    22    0043c               0012   fetch                          0000      0
    23    D_RDS.S:Write_Msg   bfff   fetch                          0000      0
    24    D_RDS.S:Write_Msg MOV      IY,#0801                       0000      0

STATUS:  n70216--Running user program         Emulation trace complete
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Begin  Halt  CMB  System  Format  Trace  Display
```

The resulting display shows the **Cmd_B** instructions, the branch to **Write_Msg**, and the beginning of the instructions that move the "Entered B command " message to the destination locations.

For a Complete Description

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

Copying Memory

You can copy the contents of one range of memory to another. This is a useful feature to test things like the relocatability of programs. To test if the sample program is relocatable within the same segment, copy the program to an unused, but mapped, area of emulation memory. For example, select:

```
Memory, Copy
```

Enter 400H through 452H as the source memory range to be copied, and enter 500H as the destination address.

To verify that the program is relocatable, run it from its new address by selecting:

```
Processor, Go, Address
```

Enter 500H. The status line shows that the emulator is "Running user program". You may wish to trace program execution or enter valid and invalid commands and search the message destination area (shown earlier in this chapter) to verify that the program works correctly at its new address.

Resetting the Emulator

To reset the emulator, select:

```
Processor, Reset, Hold
```

The emulator is reset (suspended) until you enter a "**P**rocessor **B**reak", "**P**rocessor **G**o", or "**P**rocessor **S**tep" command. A CMB execute signal also will run the emulator if reset.

You also can specify that the emulator begin executing in the monitor after reset instead of remaining in the suspended state.

To do this, select:

```
Processor, Reset, Monitor
```

Exiting the PC Interface

There are three different ways to exit the PC Interface. You can exit the PC Interface using the "locked" option which restores the current configuration next time you start the PC Interface. You can select this option as follows.

```
    System, Exit, Locked
```

Another way to execute the PC Interface is with the "unlocked" option, which presents the default configuration the next time you start the PC Interface. You can select this option with the following command.

```
    System, Exit, Unlocked
```

Or , you can exit the PC Interface without saving the current configuration using the command:

```
    System, Exit, No_Save
```

See the *Emulator PC Interface Reference* for a complete description of the system exit options and their effect on the emulator configuration.

Notes

3

"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 Emualtion 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

OBSERVE THESE PRECAUTIONS TO AVOID EMULATOR CIRCUIT DAMAGE. Take the following precautions 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, take precautions 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.

Pin Protector

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

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.

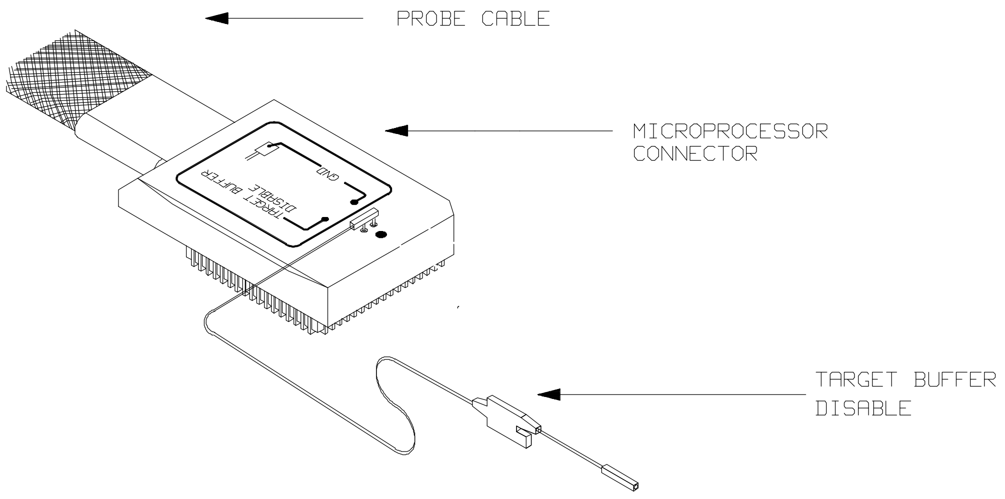
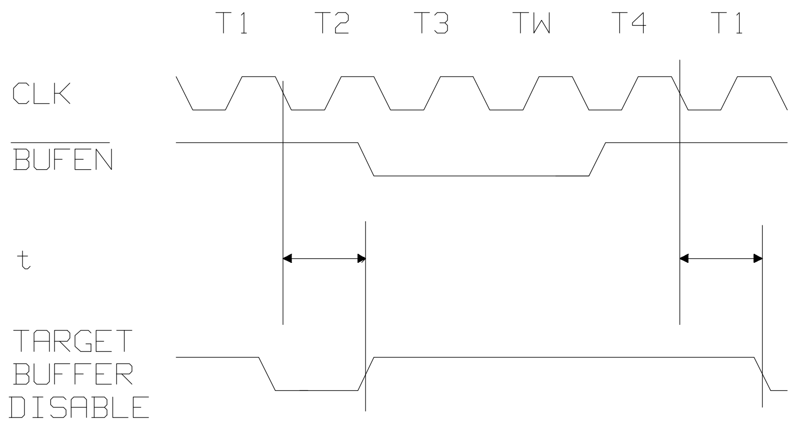


Figure 3-1. Auxiliary Output Lines

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)

Installing into a PLCC Type Socket

To connect the microprocessor connector to the target system, proceeded 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.

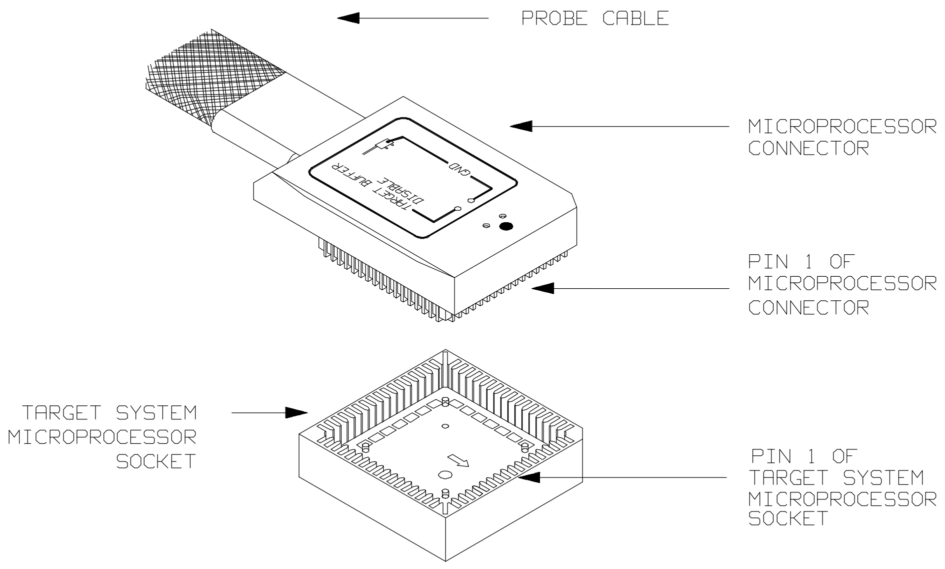

PROBE CABLE
MICROPROCESSOR CONNECTOR
GND
TARGET BUFFER DISABLE
PIN 1 OF MICROPROCESSOR CONNECTOR
TARGET SYSTEM MICROPROCESSOR SOCKET
PIN 1 OF TARGET SYSTEM MICROPROCESSOR SOCKET

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, proceeded 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.

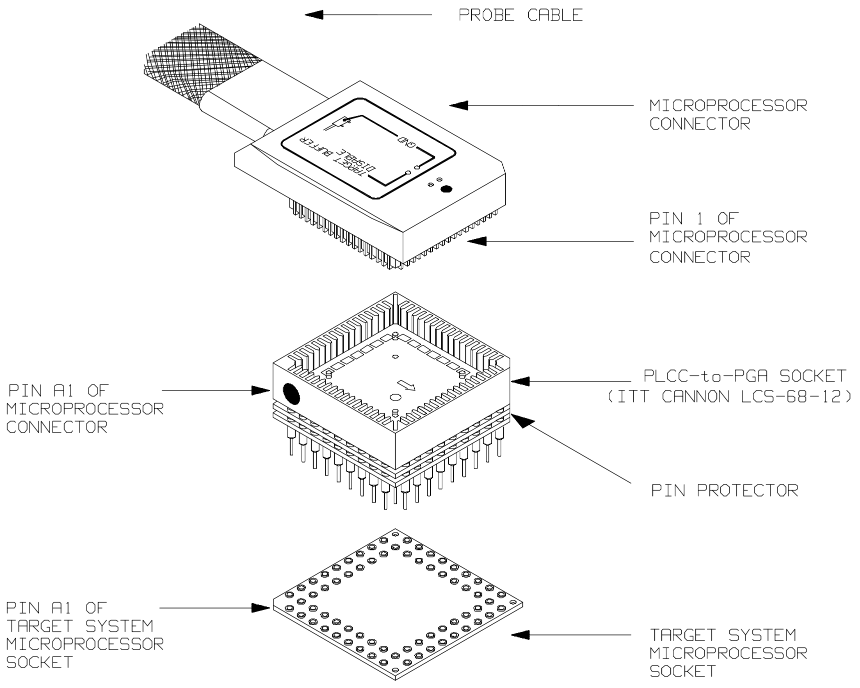

PROBE CABLE
MICROPROCESSOR CONNECTOR
TARGET BUFFER DISABLE
GND
PIN 1 OF MICROPROCESSOR CONNECTOR
PIN A1 OF MICROPROCESSOR CONNECTOR
PLCC-to-PGA SOCKET (ITT CANNON LCS-68-12)
PIN PROTECTOR
PIN A1 OF TARGET SYSTEM MICROPROCESSOR SOCKET
TARGET SYSTEM MICROPROCESSOR SOCKET

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

The default 70208 and 70216 emulator configuration selects the internal 8 MHz (system clock speed) clock as the emulator clock source. The default 70208H and 70216H 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 Inputs 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 Input From Target?" configuration in Chapter 4 of this manual).

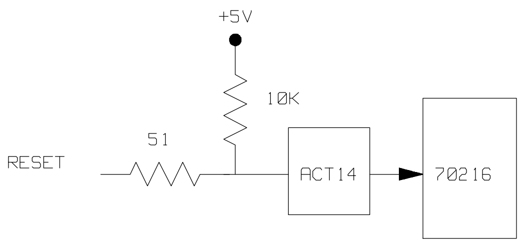
To specify a run from target system reset, select:

```
Processor, Go, Reset
```

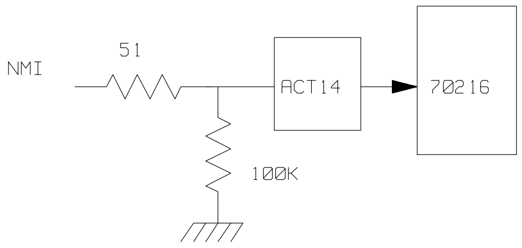
The status now shows that the emulator is "Awaiting target reset". After the target system is reset, the status line message will change to show the appropriate emulator status.

Target System Interface

$\overline{\text{RESET}}$ This singal is connected to 70216 through ACT14, 51ohm and 10K ohm pull-up register.

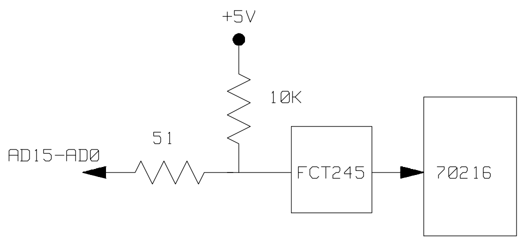


NMI This singal is connected to 70216 through ACT14, 51 ohm and 100K ohm pull-down register.



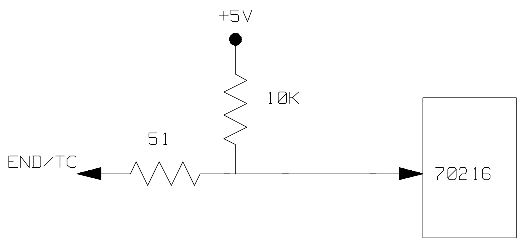
AD15-AD0

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



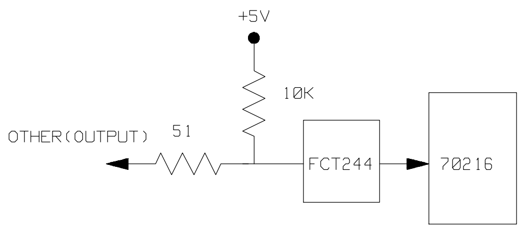
$\overline{\text{END}}/\overline{\text{TC}}$

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



OTHER(OUTPUT)

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



4

Configuring the Emulator

Introduction

Your 70216 emulator can help you in all stages of target system development. For instance, you can run the emulator out-of-circuit when developing target system software and in-circuit when integrating software with hardware. You can use the emulator's internal clock or your target system clock. Emulation memory can be used with your target system memory, and it can be mapped as RAM or ROM. You can execute your 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 versatile instrument and may be configured to suit your needs at any stage of the development process. This chapter describes the emulator configuration options.

This chapter will:

- Show you how to access the emulator configuration options.
- Describe the emulator configuration options.
- Show you how to save a particular emulator configuration, and load it again at a later time.

Prerequisites

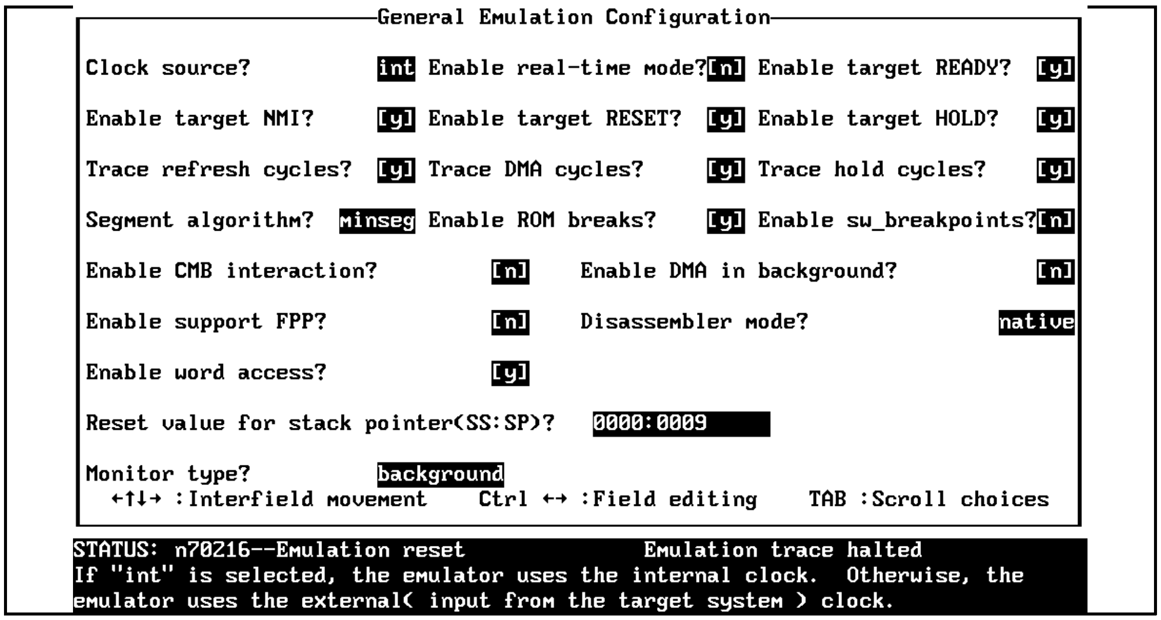
Before performing the tasks described in this chapter, you should be familiar with how the emulator operates in general. Refer to the *HP 64700 Emulators: Concepts of Emulation and Analysis* manual and the "Getting Started" chapter of this manual.

Accessing the Emulator Configuration Options

Select:

Config, **G**eneral

When you position the cursor to a configuration item, a brief description of the item appears at the bottom of the display.



```
General Emulation Configuration
Clock source?           int  Enable real-time mode?[n] Enable target READY?  [y]
Enable target NMI?      [y]  Enable target RESET?  [y] Enable target HOLD?   [y]
Trace refresh cycles?   [y]  Trace DMA cycles?     [y] Trace hold cycles?    [y]
Segment algorithm?   minseg  Enable ROM breaks?    [y] Enable sw_breakpoints?[n]
Enable CMB interaction?          [n]  Enable DMA in background?          [n]
Enable support FPP?              [n]  Disassembler mode?              native
Enable word access?              [y]
Reset value for stack pointer(SS:SP)?   0000:0009
Monitor type?           background
  ←↑↓→ :Interfield movement    Ctrl ←→ :Field editing    TAB :Scroll choices
STATUS: n70216--Emulation reset              Emulation trace halted
If "int" is selected, the emulator uses the internal clock.  Otherwise, the
emulator uses the external( input from the target system ) clock.
```

Figure 4-1. General Emulator Configuration (70216)

Note

You can use the System Terminal window to modify the emulator configuration. If you do this, some PC Interface features may no longer work properly. We recommend that you modify the emulator configuration using only the PC Interface.

Clock source

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

int	Selects the internal clock oscillator as the emulator clock source. In the 70208/70216 emulator, internal clock speed is 8 MHz (system clock). In the 70208H/70216H emulator, internal clock speed is 16 MHz (system clock). This is the default.
ext	An external target system clock is the emulator clock source. In the 70208/216 emulator, external oscillator clock sources must be within the range of 4-20 MHz. In the 70208H/70216H emulator, external oscillator clock sources must be within the range of 2-32 MHz.

Note

Changing the clock source drives the emulator into the reset state.

Enable Real-Time Mode

The "Enable real-time mode" question lets you configure the emulator to refuse commands that cause an emulator break to monitor during user program runs.

No	All commands, 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 "**P**rocessor **R**eset", "**P**rocessor **B**reak", "**P**rocessor **G**o", and "**P**rocessor **S**tep") 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

Restrict emulator to real-time runs with certain target systems. If your target system circuitry depends on constant program execution, you should restrict the emulator to real-time runs. This helps avoid target system damage. Remember that you still can execute the "**P**rocessor **R**eset", "**P**rocessor **B**reak", and "**P**rocessor **S**tep" commands. You should use caution when executing these commands.

Enable target READY

High-speed emulation memory provides no-wait-state operation. However, the emulator may optionally respond to the target system ready lines 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).

Enable target NMI

This configuration option specifies whether or not the emulation processor accepts to NMI signal generated by the target system.

Yes	The emulator accepts NMI signal generated by the target system. When the NMI signal is accepted, the emulator calls the NMI procedure as actual microprocessor.
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.

Enable target RESET

The 70216 emulator can respond or ignore target system reset while running in user program or waiting for target system reset (refer to "**P**rocessor **G**o **R**eset" command in "In-circuit Emulation" chapter). 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	If disabled, the emulator completely ignores the reset signal from target system. This is true if the emulator is in foreground (executing user program).

Enable target HOLD

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

No	The emulator ignores HOLD signal from target system completely.
Yes	The emulator accepts HOLD signal. When the HOLD is accepted, the emulator will respond as actual microprocessor.

Trace refresh cycles

This question allows you to specify whether or not the analyzer trace the 70216 emulation processor’s refresh cycles.

Yes	Specifies that the analyzer will trace the 70216 refresh cycles.
No	Specifies that the analyzer will not trace the 70216 refresh cycles.

Trace DMA cycles

This question allows you to specify whether or not the analyzer trace the 70216 emulation processor’s internal DMA cycles.

Yes	Specifies that the analyzer will trace the 70216 internal DMA cycles.
No	Specifies that the analyzer will not trace the 70216 internal DMA cycles.

Trace hold cycles

This question allows you to specify whether or not the analyzer trace the 70216 emulation processor’s hold cycles.

Yes	Specifies that the analyzer will trace the 70216 hold cycles.
No	Specifies that the analyzer will not trace the 70216 hold cycles.

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.

Enable ROM break

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 writing to memory mapped as emulation ROM. It cannot prevent writes to target system RAM locations mapped as ROM, 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.

Note

The **wrrom** analysis specification status option allows you to use “write to ROM” cycles as trigger and storage qualifiers.

Enable sw_breakpoints

This question allows you to enable or disable the software breakpoints feature.
When you define (add) a breakpoint, software breakpoints are automatically enabled.

No	The software breakpoints feature is disabled. This is the default emulator configuration, so you must change this item before you can use software breakpoints.
Yes	Allows you to use the software breakpoints feature. The emulator detects the breakpoint interrupt instruction (CC hex), it generates a break to background request which as with the "processor break" command.

When you define or enable a software breakpoint, the emulator will set the trap bit at the software breakpoint address.When software breakpoints are enabled and emulator detects the breakpoint trap bit, emulator execute the instruction at the breakpoint address and it generates a break to background request which as with the "processor break" command.
Since the system controller knows the locations of defined software breakpoints, it can determine whether the breakpoint trap interrupt is a software breakpoint or opcode in your target program.

If it is a software breakpoint, execution breaks to the monitor,and the breakpoint trap bit is cleared. A subsequent run or step command will execute from next address.

When software breakpoints are disabled, the emulator clears the trap bit. Up to 32 software breakpoints may be defined.

Enable CMB Interaction

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

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

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

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

Yes Multiple emulator start/stop is enabled. If you enter the

```
Processor, CMB, Go, ...
```

command, the emulator will start executing code when a pulse on the /EXECUTE line is received. The READY line is driven false while the emulator is running in the monitor. It goes true whenever execution switches to the user program.

Note

CMB interaction also will be enabled when you enter the

```
Processor, CMB, Execute
```

command.

Enable DMA in background

This configuration allows you to specify whether or not the emulator accepts DMARQ0-3 (DMA Request 0-3) signals 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 forground operation.

Enable support FPP

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.

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.
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.

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 normarlly 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 to 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.

Disassembler mode

This configuration specifies the mode of dis-assembler that are used by the emulator to display memory, trace, and register in mnemonic format.

native	Selecting the native mode specifies that the emulator will display dis-assembler with NEC 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.

Enable word access

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.

Yes	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.

No	Selecting the byte access mode specifies that the emulator will access target memory using upper and lower byte cycles (one byte at a time).

The 70208/70208H Emulator is the byte access mode and the 70216/70216H Emulator is the word access mode 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.

Reset value for 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 <SS>:<SP> address.

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

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.

Monitor Type

This configuration option allows you to select and use a foreground emulation monitor program. The default monitor is background monitor.

background	Specify monitor type as background monitor. When you select background monitor, you can specify the background monitor location.

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

foreground	Specify monitor type as foreground monitor. When you select foreground monitor, you must specify correct foreground monitor start address with next configuration question (foreground monitor address). After you completed the configuration setting, you need to load foreground monitor program to the emulator with "**M**emory, **L**oad" feature. The foreground monitor program must already assembled and linked with appropriate location specification. Refer to the *HP 64791/2 70208H/70216H Emulator Terminal Interface User's Guide* for more information.

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

Note If you select a foreground monitor, a 4 kilobyte block is automatically mapped at the address specified by the next question.

Foreground Monitor Address?

The location of the foreground monitor is important because it will occupy part of the processor address space. Foreground monitor location must not overlap the location of target system programs. The default foreground monitor location is "0F0000H".

When entering monitor block addresses, you must only specify addresses on 4K byte boundaries; otherwise, an invalid syntax message is displayed.

Note

Relocating the monitor causes all memory mapper terms to be removed.

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.

Storing an Emulator Configuration

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

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

To store the current emulator configuration, select:

```
Config, Store
```

Enter the name of a file in which to save the emulator configuration.

Loading an Emulator Configuration

If you want to reload a previously stored emulator configuration, select:

```
Config, Load
```

Enter the configuration file name and press **Enter**. The emulator will be reconfigured with the values specified in the configuration file

5

Using the Emulator

Introduction

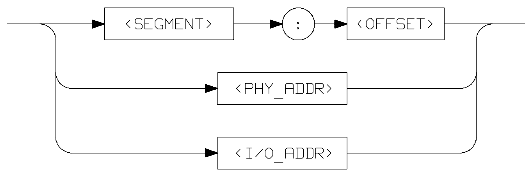
In the "Getting Started" chapter, you learned how to use the basic features of the 70216 emulator. This chapter describes the more in-depth features of the emulator.

This chapter shows you how to:

- Address syntax in emulation commands.
- Register names and classes.
- Make coordinated measurements.
- Store the contents of memory into absolute files.

Address Syntax

Syntax



The address used in emulation commands may be specified as a logical address or as a physical address (though a physical address in run or step command is coverted to logical address by the emulator system).

Expressions are defined in the *HP 64700 Emulators Terminal Interface: User’s Reference* manual.

Parameters

<SEGMENT>	This expression (0-0FFFF hex) is the segment portion of the logical address. The value specified is placed in the 70216 PS register.
<OFFSET>	This expression (0-0FFFF hex) is the offset portion of the logical address. The value specified is placed in the 70216 PC register.
<PHY_ADDR>	This expression (0-0FFFFF hex) is a physical address in the 70216 address range. In run commands , the emulation system converts this physical address to a <segment>:<offset> address as specified by the "segment algorithm" configuration option in "Configuring the Emulator" chapter.
<I/O_ADDR>	This expression (0-0FFFF hex) with no function code is a 70216 I/O address. This expression should be used in I/O command .

REGISTER NAMES and CLASSES

The following register names and classes are used with the "**R**egister **D**isplay/**M**odify" 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.

NOCLASS

Register name	**Description**
al, ah, bl, bh **cl, ch, dl, dh**	

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 (70208H/70216H 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
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, interruptis 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 (70208H/70216H Emulator only)

(DMA Control Unit register (for uPD71037 mode)

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)

Making Coordinated Measurements

Coordinated measurements are measurements synchronously made in multiple emulators or analyzers. Coordinated measurements can be made between HP 64700 Series emulators, which communicate over the Coordinated Measurement Bus (CMB). Coordinated measurements can also be made between an emulator and another instrument connected to the BNC connector.

This chapter will describe coordinated measurements made from the PC Interface which involve the emulator. These types of coordinated measurements are:

- Running the emulator on reception of the CMB /EXECUTE signal.
- Using the analyzer trigger to break emulator execution into the monitor.

Three signal lines on the CMB are active and serve the following functions:

/TRIGGER	Active low. The analyzer trigger line on the CMB and on the BNC serve the same logical purpose. They provide a means for the analyzer to drive its trigger signal out of the system, or for external trigger signals to arm the analyzer or break the emulator into its monitor.
READY	Active high. This line is for synchronized, multi-emulator start and stop. When you enable CMB run control interaction, all emulators must break to background on receipt of a false READY signal and will not return to foreground until this line is true.
/EXECUTE	Active low. This line serves as a global interrupt signal. On receipt of an enabled /EXECUTE signal, each emulator is to interrupt whatever it is doing and execute a previously defined process, such as run the emulator or start a trace measurement.

Running the Emulator at /EXECUTE

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

```
    Config, General
```

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

To begin executing a program on receipt of the /EXECUTE signal, select:

```
    Processor, CMB, Go
```

Now you may select either the current program counter ("Pc", in other words, the current PS:PC), or a specific address.

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

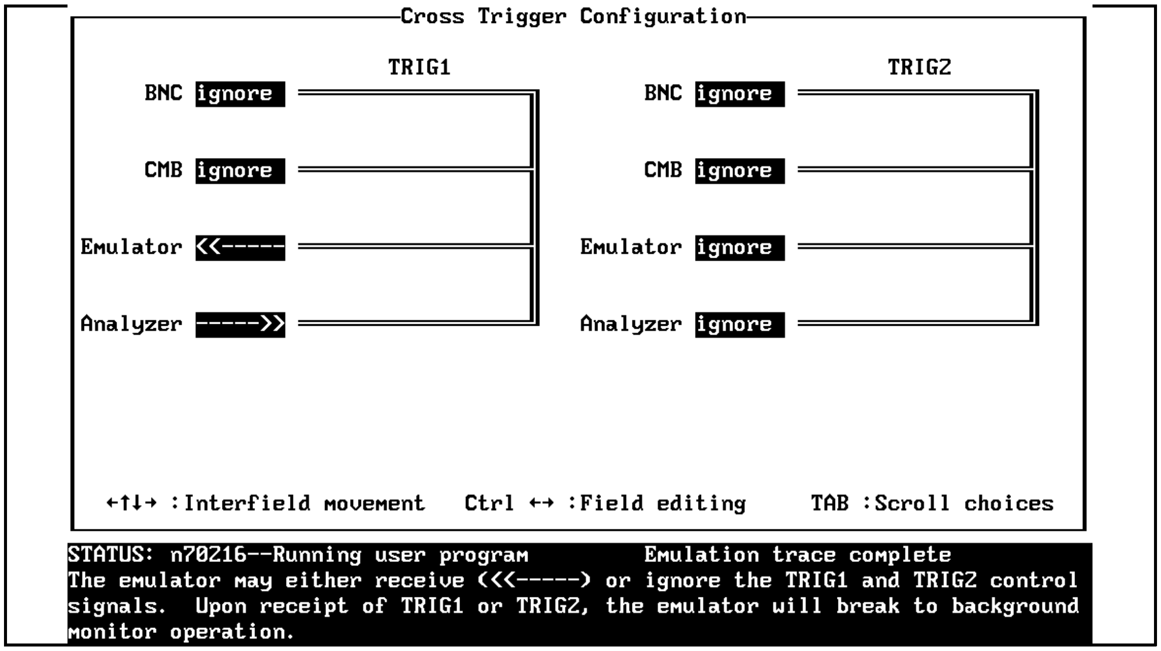
Breaking on the Analyzer Trigger

To break emulator execution into the monitor when the analyzer trigger condition occurs, you modify the trigger configuration. To access the trigger configuration, select:

```
    Config, Trigger
```

The trigger configuration display contains two diagrams, one for each internal TRIG1 and TRIG2 signal.

To use the internal TRIG1 signal to connect the analyzer trigger to the emulator break line, move the cursor to the highlighted "Analyzer" field in the TRIG1 portion of the display. Use the **TAB** key to select the "----->>" arrow pointing from the analyzer to TRIG1. Next, move the cursor to the highlighted "Emulator" field and use the **TAB** key to select the arrow pointing toward the emulator (<<-----). This specifies that emulator execution will break into the monitor when the TRIG1 signal is driven. The trigger configuration display appears as follows:



```
Cross Trigger Configuration

                    TRIG1                                   TRIG2
      BNC ignore                             BNC ignore
      CMB ignore                             CMB ignore
 Emulator <<-----                       Emulator ignore
 Analyzer ----->>                       Analyzer ignore

  ←↑↓→ :Interfield movement   Ctrl ←→ :Field editing     TAB :Scroll choices
STATUS: n70216--Running user program         Emulation trace complete
The emulator may either receive (<<-----) or ignore the TRIG1 and TRIG2 control
signals.  Upon receipt of TRIG1 or TRIG2, the emulator will break to background
monitor operation.
```

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.

```
   Memory, Store
```

When you store memory using "**M**emory, **S**tore" command, the address information saved to an absolute file is defined from the address expression used in the "**M**emory **S**tore" command. refer to "Address Expression in Emulation Commands" section in this chapter.

Note

The first character of the absolute file name must be a letter. You can name the absolute file with a total of 8 alphanumeric characters. You also can include an extension of up to 3 alphanumeric characters. If the file is stored in HP 64000 format, its extension must be ".X".

Caution

The "**M**emory **S**tore" command writes over an existing file if it has the same name that is specified with the command. You may wish to verify beforehand that the specified filename does not already exist.

Notes

File Format Readers

Introduction

The 70216 PC Interface is provided with the following "reader".

- Intel Object Module Format (OMF86) Reader
  - (This Reader is for the Intel OMF86 absolute file)
- NEC30 Reader
  - (This Reader is for the load module format file which is generated by NEC LK70116 linker for uPD70208 and uPD70216)
- HP64000 Reader

The Reader converts the file(s) into two files that are usable with the HP 64792 emulator. This means that you can use available language tools to create absolute files, then load those files into the emulator using the 70216 PC Interface.

The Reader can operate from within the PC Interface or as a separate process. When operating the Reader, it may be necessary to execute it as a separate process if there is not enough memory on your personal computer to operate the PC Interface and Reader simultaneously. You can also operate the reader as part of a "make file".

Using the OMF86, NEC30 Reader

What the Reader Accomplishes

The Reader accepts as input an absolute file in the form "<file>.<ext>", and creates two new files that are used by the PC Interface: an "absolute" file, and an ASCII symbol file.

The Absolute File

During execution of the Reader, an absolute file (<file>.HPA) is created. This absolute file is a binary memory image which is optimized for efficient downloading into the emulator.

The ASCII Symbol File

The ASCII symbol file (<file>.HPS) produced by the Reader contains global symbols, module names, local symbols, and, when using applicable development tools such as a "C" Compiler, program line number. Local symbols evaluate to a fixed (static, not stack relative) address.

Note

You must use the required options for your specific language tools to include symbolic ("debug") information in the absolute file. The Reader will only convert symbol information that is present in the input absolute file.

The symbol file contains symbol and address information in the following form:

```
 module_name1
 module_name2
 ...
module_nameN
global_symbol1  0100:1234
global_symbol2  0100:5678
...
global_symbolN  0100:ABCD
|module_name|# 1234         0200:0872
|module_name|local_symbol1  0200:0653
|module_name|local_symbol2  0200:0872
...
|module name|local_symbolN  0200:0986
```

The space preceding module names is required. A single tab separates symbol and address.

Each of the symbols is sorted alphabetically in the order: module names, global symbols, and local symbols.

The local symbols are scooped. This means that to access a variable named "count" in a function named "foo" in a source file module named "main.c", you would enter "main.c:foo.count". See table A-1.

Module Name	**Function Name**	**Variable Name**	**You Enter:**
MAIN.C	FOO	COUNT	MAIN.C:FOO.COUNT
MAIN.C	BAR	COUNT	MAIN.C:BAR.COUNT
MAIN.C	line number 23		MAIN.C: line 23

Table A-1. How to Access Variables

Line numbers will appear similar to a local symbol except that "local_symbolX" will be replaced by "#NNNNN" where NNNNN is a five digit decimal number. Line numbers should appear in ascending order.

Note

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

Line number symbols are accessed by entering the following on one line in the order shown:

module name
colon (:)
space
the word "line"
space
the decimal line number

For example:

```
MAIN.C: line 23
```

Location of the Reader Program

The Reader is located in the directory named **\hp64700\bin** by default, along with the PC Interface. This directory must be in the environment variable PATH for the Reader and PC Interface to operate properly. This is usually defined in the "\autoexec.bat" file. **The following examples assume that you have "\hp64700\bin" include in your PATH variable. If not, you must supply the directory name when executing the Reader program.**

Using the Reader from MS-DOS

The command names for the Reader are shown below.

Intel OMF86 Reader	RDOMF86.EXE
NEC30 Reader	RDNEC30.EXE

You can execute the Reader from the command line with the following command syntax:

```
C:\HP64700\BIN\<READER> [-q] [-u] [-m]
<filename> <RETURN>
```

<READER>	is the name of the command name for the Reader
[-q]	Specifies the "quiet" mode. This option suppress the display of messages.
[-u]	Specifies that the first leading underscore ("_") of a symbol is not removed.
[-m]	(RDOMF86.EXE only) Specifies that the OMF86 Reader removes duplicate module names generated by some construction tools. Some tools enclose all of the functions and variables in a module within a block (or function) whose name is the same as that of the module (or source file). When this option is used, the Intel OMF86 Reader will ignore the first enclosing block in a module is its name matches the module name.
<filename>	Specifies the same of the file containing the absolute program. You can include an extension in the file name.

The following commands will create the files "TESTPROG.HPA" and "TESTPROG.HPS".

```
ENTER: RDOMF86 TESTPROG.ABS
ENTER: RDNEC30 TESTPROG.LNK
```

Using the Reader from the PC Interface

The 70216 PC Interface has a file format option under the "**M**emory **L**oad" command.

After you select OMF86 as the file format, the Intel OMF86 Reader will operate on the file you specify. After the Reader completes successfully, the 70216 PC Interface will load the absolute and symbol files produced by the Reader.

To use the Reader from the PC Interface, follow these steps:

1. Start up the 70216 PC Interface.
2. Select "**M**emory, **L**oad". The memory load menu will appear.
3. Specify the file format as "OMF86". This will appear as the default file format.
4. Specify the memory to be loaded (emulation, target, or both).
5. Specify to force the file format reader to regenerate the emulator absolute file (.HPA) and symbol database (.HPS) before loading the code. Normally, these files are only regenerated whenever the file you specify (the output of your language tools) is never than the emulator absolute file and symbol database.
6. Specify that the OMF86 Reader removes duplicate module names generated by some construction tools. Some tools enclose all of the functions and variables in a module within a block (or function) whose name is the same as that of the module (or source file). When this option is used, the Intel OMF86 Reader will ignore the first enclosing block in a module is its name matches the module name.
7. Specify that the first leading underscore ("_") of a symbol is not removed.

8. Specify a file in Intel OMF86 format ("TESTFILE.OMF", for example). **The file extension can be something other than ".OMF", but ".HPA" or ".HPS" cannot be used.**

Using the Intel OMF86 file that you specify (TESTFILE.OMF, for example), the PC Interface performs the following:

- It checks to see if two files with the same base name and extensions .HPS and .HPA already exist (for example, TESTFILE.HPS and TESTFILE.HPA).

- If TESTFILE.HPS and TESTFILE.HPA don't exist, the Intel OMF86 Reader produces them. The new absolute file, TESTFILE.HPA, is then loaded into the emulator.

- If TESTFILE.HPS and TESTFILE.HPA already exist but the create dates and times are earlier than the Intel OMF86 file creation date/time, the Intel OMF86 Reader recreates them. The new absolute file, TESTFILE.HPA, is then loaded into emulator.

- If TESTFILE.HPS and TESTFILE.HPA already exist but the dates and times are later than the creation date/time for the Intel OMF86 file, the current absolute file, TESTFILE.HPA, is then loaded into the emulator.

Note

Date/time checking only done within the PC Interface. When you run the Reader at the MS-DOS command line prompt, the Reader will always update the absolute and symbol files.

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

If the Reader Won't Run

If your program is very large, the PC Interface may run out of memory while attempting to create the database file. If this occurs, exit the PC Interface and execute the Reader program at the MS-DOS command prompt.

Including Reader in a Make File

You may want to incorporate the "RDOMF86" or "RDNEC30" process as the last step in your "make" file, or as a step in your construction process, so as to eliminate the possibility of having to exit the PC Interface due to space limitations describe above. If the file with "-.HPA" and "-.HPS" extensions are not current, loading an absolute file will automatically create them.

Using the HP 64000 Reader

An HP 64000 “reader” is provided with the PC Interface. The HP 64000 Reader converts the files into two files that are usable with your emulator. This means that you can use available language tools to create HP 64000 absolute files, then load those files into the emulator using the PC Interface.

The HP 64000 Reader can operate from within the PC Interface or as a separate process. When operating the HP 64000 Reader, it may be necessary to execute it as a separate process if there is not enough memory on your personal computer to operate the PC Interface and HP 64000 Reader simultaneously. You can also operate the reader as part of a “make file.”

What the Reader Accomplishes

Using the HP 64000 files (<file.X>, <file.L>, <scr1.A>, <scr2.A>, ...) the HP 64000 Reader will produce two new files, an “absolute” file and an ASCII symbol file, that will be used by the PC Interface. These new files are named: “<file>.hpa” and “<file>.hps.”

The Absolute File

During execution of the HP 64000 Reader, an absolute file (<file>.hpa) is created. This absolute file is a binary memory image which is optimized for efficient downloading into the emulator.

The ASCII Symbol File

The ASCII symbol file (<file>.hps) produced by the HP 64000 Reader contains global symbols, module names, local symbols, and, when using applicable development tools such as a “C” compiler, program line numbers. Local symbols evaluate to a fixed (static, not stack relative) address.

Note

You must use the required options for your specific language tools to include symbolic (“debug”) information in the HP 64000 symbol files. The HP 64000 Reader will only convert symbol information present in the HP 64000 symbol files (<file.L>, <src1.A>, <src2.A>, ...).

The symbol file contains symbol and address information in the following form:

```
 module_name1
 module_name2
 ...
 module_nameN
global_symbol1  0100:1234
global_symbol2  0100:5678
...
global_symbolN  0100:ABCD
|module_name1|# 1234         0200:0872
|module_name1|local_symbol1  0200:0653
|module_name1|local_symbol2  0200:0872
...
|module_name1|local_symbolN  0200:0986
```

Each of the symbols is sorted alphabetically in the order: module names, global symbols, and local symbols.

Line numbers will appear similar to a local symbol except that “local_symbolX” will be replaced by “#NNNNN” where NNNNN is a five digit decimal line number. The addresses associated with global and local symbols are specific to the processor for which the HP 64000 files were generated.

Note

If your emulator can store symbols internally, symbols will appear in disassembly. When the line number symbol is displayed in the emulator, it appears in brackets. Therefore, the symbol “MODNAME: line 345” will be displayed as “MODNAME:[345]” in mnemonic memory and trace list displays.

The space preceding module names is required. Although formatted for readability here, a single tab separates symbol and address.

The local symbols are scooped. This means that to access a variable named “count” in a source file module named “main.c,” you would enter “MAIN.C:COUNT” as shown below.

Module Name	Variable Name	You Enter:
MAIN.C	COUNT	MAIN.C:COUNT
MAIN.C	line number 23	MAIN.C: line 23

Table A-2. How to Access Variables

You access line number symbols by entering the following on one line in the order shown:

module name
colon (:)
space
the word “line”
space
the decimal line number

For example:

```
MAIN.C: line 23
```

Location of the HP 64000 Reader Program

The HP 64000 Reader is located in the directory named \hp64700\bin by default, along with the PC Interface. This directory must be in the environment variable PATH for the HP 64000 Reader and PC Interface to operate properly. The PATH is usually defined in the "\autoexec.bat" file.

The following examples assume that you have "\hp64000\bin" included in your PATH variable. If not, you must supply the directory name when executing the Reader program.

Using the Reader from MS-DOS

The command name for the HP 64000 Reader is **RHP64000.EXE**. To execute the Reader from the command line, for example, enter:

```
RHP64000 [-q] <filename>
```

[-q]	This option specifies the "quiet" mode, and suppresses the display of messages.
<filename>	This represents the name of the HP 64000 linker symbol file (file.L) for the absolute file to be loaded.

The following command will create the files "TESTPROG.HPA"and "TESTPROG.HPS"

```
RHP64000 TESTPROG.L
```

Using the Reader from the PC Interface

The PC Interface has a file format option under the "**M**emory **L**oad" command. After you select HP64000 as the file format, the HP 64000 Reader will operate on the file you specify. After this completes successfully, the PC Interface will accept the absolute and symbol files produced by the Reader.

To use the Reader from the PC Interface:

1. Start up the PC Interface.
2. Select "**M**emory **L**oad." The memory load menu will appear.
3. Specify the file format as "HP64000." This will appear as the default file format.
4. Specify the name of an HP 64000 linker symbol file (TESTFILE.L for example).

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

- It checks to see if two files with the same base name and extensions .HPS and .HPA already exist (for example, TESTFILE.HPS and TESTFILE.HPA).
- If TESTFILE.HPS and TESTFILE.HPA don't exist, the HP 64000 Reader produces them. The new absolute file, TESTFILE.HPA, is then loaded into the emulator.
- If TESTFILE.HPS and TESTFILE.HPA already exist but the create dates and times are earlier than the HP 64000 linker symbol file creation date/time, the HP 64000 Reader recreates them. The new absolute file, TESTFILE.HPA, is then loaded into the emulator.
- If TESTFILE.HPS and TESTFILE.HPA already exist but the dates and times are later than the creation date and time for the HP 64000 linker symbol file, the HP 64000 Reader will not recreate TESTFILE.HPA. The current absolute file, TESTFILE.HPA, is then loaded into the emulator.

Note Date/time checking is only done within the PC Interface. When running the HP 64000 Reader at the MS-DOS command line prompt, the HP 64000 Reader will always update the absolute and symbol files.

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

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

If the Reader Won't Run

If your program is very large, the PC Interface may run out of memory while attempting to create the database file. If this occurs, you will need to exit the PC Interface and execute the program at the MS-DOS command prompt to create the files that are downloaded to the emulator.

Including RHP64000 in a Make File

You may wish to incorporate the "RHP64000" process as the last step in your "make file," as a step in your construction process, to eliminate the possibility of having to exit the PC Interface due to space limitations describe above. If the files with ".HPA" and ".HPS" extensions are not current, loading an HP 64000 file will automatically create them.

Index

Notes