HP 64146

7700 Series Emulator PC Interface

User's Guide

HEWLETT PACKARD

HP Part No. 64146-97004
Printed in U.S.A.
February 1994

Edition 2

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	**64146-97001, August 1992**
Edition 2	**64146-97004, February 1994**

Using This Manual

This manual introduces you to the HP 64146A/B 7700 Series emulator as used with the PC Interface.

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 does not:

- Show you how to use every PC Interface command and option; the PC Interface is described in the *HP 64700 Emulators PC Interface: User's Reference.*

For the most part, the HP 64146A and HP 64146B emulators all operate the same way. Differences of between the emulators are described where they exist. Both the HP 64146A and HP 64146B emulators will be referred to as the "HP 64146A/B 7700 Series emulator" or "7700 Series emulator". In the specific instances where HP 64146B emulator differs from HP 64146A emulator, it will be described as "HP 64146B emulator".

Organization

Chapter 1	**Introduction to the 7700 Series Emulator.** This chapter lists the 7700 Series emulator features and describes how they can help you in developing new hardware and software.
Chapter 2	**Getting Started.** This chapter shows you how to use emulation commands by executing them on a sample program. This chapter describes the sample program and how to: load programs into the emulator, map memory, display and modify memory, display registers, step through programs, run programs, set software breakpoints, search memory for data, and use the analyzer.
Chapter 3	**"In-Circuit" Emulation.** This chapter shows you how to plug the emulator into a target system, and how to use the "in-circuit" emulation features.
Chapter 4	**Configuring the Emulator.** You can configure the emulator to adapt it to your specific development needs. This chapter describes the options available when configuring the emulator and how to save and restore particular configurations.
Chapter 5	**Using the Emulator.** This chapter describes emulation topics which are not covered in the "Getting Started" chapter (for example, coordinated measurements and storing memory).
Appendix A	**File Format Readers.** This chapter shows you what the "Reader" program accomplishes, and how to use it.

Contents

1 **Introduction to the 7700 Series Emulator**

Introduction 1-1
Purpose of the 7700 Series Emulator 1-1
  Supported Microprocessors 1-3
Features of the 7700 Series Emulator 1-5
  Clock Speed 1-5
  Emulation memory 1-5
  Analysis 1-6
  Foreground or Background Emulation Monitor 1-6
  Register Display and Modification 1-7
  Single-Step 1-7
  Breakpoints 1-7
  Real Time Operation 1-7
  Coverage Measurements 1-8
  Reset Support 1-8
  Watch Dog Timer 1-8
Limitations, Restrictions 1-9
  Access to Internal RAM 1-9
  Trace Internal RAM 1-9
  DMA Support 1-9
  Watch Dog Timer in Background 1-9
  Step Command with Foreground Monitor 1-9
  Step Command and Interrupts 1-9
  Emulation Commands in Stop/Wait Mode 1-10
  Stack Address 1-10

2 **Getting Started**

Introduction 2-1
Before You Begin 2-2
  Prerequisites 2-2
  A Look at the Sample Program 2-3
  Sample Program Assembly 2-7
  Linking the Sample Program 2-7
Starting Up the PC Interface 2-8

Selecting PC Interface Commands 2-9
Emulator Status 2-9
Modifying Configuration 2-9
Selecting Processor Type 2-9
Defining the Reset Value for Stack Pointer 2-9
Mapping Memory 2-11
Which Memory Locations Should Be Mapped? 2-12
Loading Programs into Memory 2-14
File Format 2-14
Memory Type 2-14
Force Absolute File Read 2-15
Absolute File Name 2-15
Using Symbols 2-15
Displaying Global Symbols 2-15
Displaying Local Symbols 2-16
Transfer Symbols to the Emulator 2-18
Displaying Memory in Mnemonic Format 2-19
Stepping Through the Program 2-20
Specifying a Step Count 2-22
Modifying Memory 2-23
Running the Program 2-24
Searching Memory for Data 2-24
Breaking into the Monitor 2-25
Using Software Breakpoints 2-25
Defining a Software Breakpoint 2-26
Displaying Software Breakpoints 2-27
Setting a Software Breakpoint 2-27
Clearing a Software Breakpoint 2-27
Using the Analyzer 2-28
Resetting the Analysis Specification 2-28
Specifying a Simple Trigger 2-28
Starting the Trace 2-30
Displaying the Trace 2-30
Using the Storage Qualifier 2-32
Restriction of the Analyzer 2-35
Using a Command File 2-37
Resetting the Emulator 2-39
Exiting the PC Interface 2-39

3 "In-Circuit" Emulation

Installing the Target System Probe 3-2

Installing the Target System Probe 3-3

4 Configuring the Emulator

Introduction 4-1
Accessing the Emulator Configuration Options 4-2
Internal Clock? 4-3
Real-Time Mode? 4-4
Breaks on ROM Writes? 4-5
Software Breakpoints? 4-5
CMB Interaction? 4-6
Target interrupts? 4-7
High speed access mode? 4-7
Insert auto RDY? 4-9
Watchdog timer ? 4-9
Trace HOLD tags? 4-9
Trace refresh? 4-10
Trace DMA cycles? 4-10
Processor type 4-10
Processor Mode 4-12
Memory data access width 4-12
Reset stack pointer 4-13
16-bit address display 4-13
Monitor Type 4-14
Foreground Monitor Address 4-15
Storing an Emulator Configuration 4-15
Loading an Emulator Configuration 4-16

5 Using the Emulator

Introduction 5-1
Internal RAM and SFR 5-2
Making Coordinated Measurements 5-3
Running the Emulator at /EXECUTE 5-4
Breaking on the Analyzer Trigger 5-5
Storing Memory Contents to an Absolute File 5-6

A File Format Readers

Using the HP 64146 Format Reader A-1
What the Reader Accomplishes A-1
Location of the HP 64146 Reader Program A-3
Using the HP 64146 Format Reader from MS-DOS A-3

Using the HP 64146 Format Reader from the PC Interface . . . A-4
If the Reader Won’t Run A-5
Including RD7700 in a Make File A-5
Using the IEEE-695 Reader A-6
What the Reader Accomplishes A-6
Location of the IEEE-695 Reader Program A-8
Using the IEEE-695 Reader from MS-DOS A-8
Using the IEEE-695 Reader from the PC Interface A-9
If the IEEE-695 Reader Won’t Run A-10
Including RIEEE695 in a Make File A-10

Illustrations

Figure 1-1. HP 64146 Emulator for MELPS 7700 Series 1-2
Figure 2-1. Connecting the Emulation Pod 2-3
Figure 2-2. Sample Program Listing 2-4
Figure 2-3. Linkage Editor Command File 2-7
Figure 2-4. PC Interface Display 2-8
Figure 2-5. Sample Program Load Map Listing 2-12
Figure 2-6. Memory Configuration Display 2-13
Figure 2-7. Modifying the Trace Specification 2-29
Figure 2-8. Modifying the Pattern Specification 2-29
Figure 3-1. Installing the Probe to LCC80 Socket 3-3
Figure 3-2. Installing the Probe to SDIP64 Socket 3-4
Figure 4-1. General Configuration Display 4-2
Figure 5-1. Cross Trigger Condition 5-5

Tables

Table 1-1. Supported Microprocessors 1-3
Table 2-1. Processor Name for Emulator Configuration 2-10
Table 4-1. <chip_name> for Emulator Configuration 4-11

Introduction to the 7700 Series Emulator

Introduction

The topics in this chapter include:

- Purpose of the 7700 Series Emulator
- Features of the 7700 Series Emulator

Purpose of the 7700 Series Emulator

The HP 64146A/B 7700 Series Emulator is designed to replace the MELPS 7700 Series microprocessor in your target system so you can control operation of the processor in your application hardware (usually referred to as the *target system*). The emulator performs just like the MELPS 7700 Series microprocessor, but is a device that allows you to control the MELPS 7700 Series directly. These features allow you to easily debug software before any hardware is available, and ease the task of integrating hardware and software.

Note

In this manual, MELPS 7700 Series is referred to as 7700 Series.

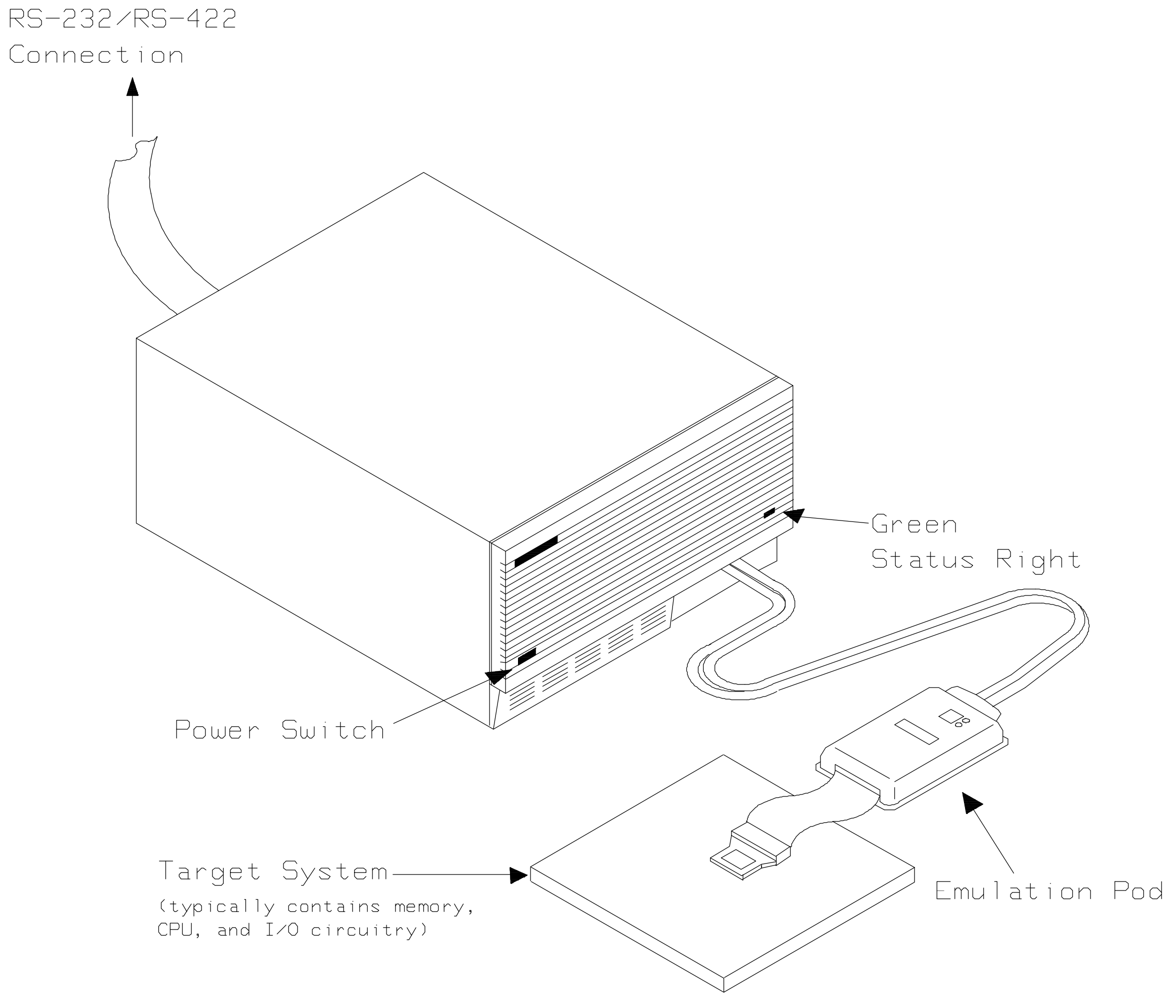


Figure 1-1. HP 64146 Emulator for MELPS 7700 Series

Supported Microprocessors

A list of the supported 7700 Series microprocessors is shown in Table 1-1. You need to purchase appropriate emulation pod and emulation processor.

Processor		Clock	Emulation Processor	Emulation Pod
M37700/1	M2-xxxFP/SP	8	M37700SAFP	M37700T-HPD
	M2AxxxFP/SP	16		
	SFP/SP	8		
	SAFP/SP	16		
M37700/1	M4-xxxFP/SP	8	M37700S4AFP	
	M4AxxxFP/SP	16		
	S4FP/SP	8		
	S4AFP/SP	16		
M37702/3	M2-xxxFP/SP	8	M37702S1AFP	M37702T-HPD
	M2AxxxFP/SP	16		
	S1FP/SP	8		
	S1AFP/SP	16		
M37702/3	M4-xxxFP/SP	8	M37702S4AFP	
	M4AxxxFP/SP	16		
	S4FP/SP	8		
	S4AFP/SP	16		
M37702	M6LxxxFP	8	M37702S1BFP	M37702TL-HPD HP 641466-61002 (64146B)
M37702/3	M2BxxxFP/SP	25	M37702S1BFP	M37702TB-HPD
	S1BFP/SP	25		HP 64146-61001 (64146A)
M37702/3	M4BxxxFP/SP	25	M37702S4BFP	HP 64146-61002
	S4BFP/SP	25		(64146B)
	M6BxxxFP	25		
M37704/5	M2-xxxFP/SP	8	M37704S1AFP	M37704T-HPD
	M2AxxxFP/SP	16		
	S1FP/SP	8		
	S1AFP/SP	16		
M37704	M3BxxxFP	25	M37704M4BFP	M37704TB-HPD
	M3BxxxFP	25		
M37710	M4BxxxFP	25	M37710M4BFP	M37710TL-HPD
	S4BFP	25		
M37720	S1FP	8	M37720S1AFP	M37720T-HPD
	S1AFP	16		
M37730	S2FP/SP	8	M37730S2AFP	M37730T-HPD
	S2AFP/SP	16		

Table 1-1. Supported Microprocessors

M37732	S4FP/SP	8	M37732S4AFP	M37732T-HPD
	S4AFP/SP	16		
M37780	STJ/FP	16	M37780STJ	M37780T-HPD
M37781	M4TxxxJ/FP	16	M37781M4TJ	M37781T-HPD
	E4TxxxJ/FP	16		
M37795	SJ	8	M37795SJ	M37795T-HPD
	STJ	8		
M37796	E4-xxxJ	8	M37796E4J	
	E4TxxxJ	8		

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

The HP 64146A emulator is provided with the following items.

- HP 64146-61001 emulation pod with M37702S1BFP emulation processor
- Adaptor for M37703 processor

The HP 64146B emulator is provided with the following items.

- HP 64146-61002 emulation pod with M37702S1BFP emulation processor
- Adaptor for M37703 processor

As you can see from Table 1-1, the HP 64146A/B emulator can emulate M37702/3M2 and M37702/3S1 processor by default. These emulation pods can be used with clock up to 25 MHz. Also, HP 64146B emulator can emulate M37702 M6L processor using default emulation pod, HP 64146-61002.

To emulate other processors of 7700 Series, you need to purchase appropriate emulation pod and/or emulation processor.

The HP 64146A/B #001 emulator is provided with no emulation pod. You need to purchase appropriate emulation pod and emulation processor listed in Table 1-1.

To purchase emulation pod or emulation processor, contact the address listed in the manual provided with your emulation pod.

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

Features of the 7700 Series Emulator

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

Clock Speed

The HP 64146-61001 and HP 64146-61002 emulation pods generate internal clock of 1 MHz. These emulation pods can be used with target system clock up to 25 MHz.

The emulator can run with no wait state up to 25 MHz. When clock is faster than 16 MHz, you can use the emulator with one of the following methods.

- Insert one wait state by the RDY signal. The emulator can be configured to generate the RDY signal. Also, the emulator accepts RDY signal from the target system.

- Use the high speed access mode of the emulator. The emulator can run with no wait state. However, there is a limitation in the mapping of the emulation memory in this mode. Refer to Chapter 4 of this manual for more detail.

Emulation memory

The HP 64146A/B 7700 Series emulator is used with one of the following Emulation Memory Cards.

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

The emulation memory can be configured into 256 byte blocks. A maximum of 16 ranges can be configured as emulation RAM (eram), emulation ROM (erom), target system RAM (tram), target system ROM (trom), or guarded memory (grd). The HP 64146A/B 7700

Series emulator will attempt to break to the emulation monitor upon accessing guarded memory; additionally, you can configure the emulator to break to the emulation monitor upon performing a write to ROM (which will stop a runaway program).

Analysis

The HP 64146A/B 7700 Series emulator is used with one of the following analyzers which allows you to trace code execution and processor activity.

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

Foreground or Background Emulation Monitor

When you power up the emulator, or when you initialize it, the background monitor is used by default. You can also configure the emulator to use a foreground monitor. Before the background and foreground monitors are described, you should understand the function of the emulation monitor program.

The Function of the Monitor Program

The monitor program is the interface between the emulation system controller and the target system. The emulation system controller uses its own microprocessor to accept and execute emulation, system, and analysis commands. The monitor program is executed by the emulation processor.

The monitor program makes possible emulation commands which access target system resources. (The only way to access target system resource is through the emulation processor.) For example, when you enter a command to modify target system memory, it is the execution of monitor program instructions that cause the new values to be written to target system memory.

The Background Monitor

On emulator power-up, or after initialization, the emulator uses the background monitor program. The background monitor does not occupy processor address space.

The Foreground Monitor

You can configure the emulator to use a foreground monitor program. When a foreground monitor is selected it executes in the foreground emulator mode. The foreground monitor occupies processor memory space and executes as if it were part of your program.

Register Display and Modification

You can display or modify the 7700 Series internal register contents. This includes the ability to modify the program counter (PC) and the program bank register (PG) values so you can control where the emulator starts a program run.

Single-Step

When you are using the background monitor, you can direct the emulation processor to execute a single instruction or a specified number of instructions.

Breakpoints

You can set the emulator/analyzer interaction so the emulator will break to the monitor program when the analyzer finds a specific state or states, allowing you to perform post-mortem analysis of the program execution. You can also set software breakpoints in your program. This feature is realized by inserting BRK instructions into user program. Refer to the "Using Software Breakpoints" section of "Getting Started" chapter for more information.

Real Time Operation

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

Coverage Measurements

Coverage memory is provided for the processor’s external program memory space. This memory allows you to perform coverage measurements on programs in emulation memory.

Reset Support

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

Watch Dog Timer

You can configure the emulator to disable the watch dog timer.

Limitations, Restrictions

Access to Internal RAM

Modifying internal RAM or SFR suspends user program execution.

Trace Internal RAM

Read data from the internal RAM or SFR is not traced correctly by the emulation analyzer.

Note

Write data is also not traced correctly, when the following conditions are met:

- The emulator is used with the M37795 emulation pod.
- The processor is operating in the memory expansion or microprocessor mode with 8 bit external bus.

DMA Support

Direct memory access to emulation memory is not allowed.

Watch Dog Timer in Background

Watch dog timer suspends count down while the emulator is running in background monitor.

Step Command with Foreground Monitor

Step command is not available when the emulator is used with a foreground monitor.

Step Command and Interrupts

When an interrupt occurs while the emulator is running in monitor, the emulator fails to do the first step operation. The emulator will display the mnemonic of the instruction which should be stepped, but the instruction is not actually executed. The second step operation will step the first instruction of the interrupt routine.

Emulation Commands in Stop/Wait Mode

When the 7700 microprocessor is in the stop or wait mode, emulation commands which access memory or registers will fail. You need to break the emulator into the monitor to use these commands. Once you break the emulator into the monitor, the stop or wait mode will be released.

Stack Address

In some versions of 7700 microprocessor, the stack can be located in Bank FF. However, the HP 64146A/B 7700 Series emulator doesn't support the feature. The stack must be located in Bank 0.

Getting Started

Introduction

This chapter leads you through a basic, step by step tutorial that shows how to use the HP 64146A/B emulator with the PC Interface.

This chapter will:

- Tell you what must be done before you can use the emulator as shown in the tutorial examples.
- Describe the sample program used for this chapter's examples.
- Briefly describe how PC Interface commands are entered 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 execution of the sample program.

Before You Begin

Prerequisites

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

1. Connected the emulator to your computer. The HP 64700 Series Installation/Service manual 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.

Connected the emulation pod to the emulator as shown on Figure 2-1.

Caution

Turn off power of the emulator before inserting the cables into the emulation pod.

You should read the *HP 64700 Emulators PC Interface: User's Reference* manual to learn how to use the PC Interface in general. For the most part, this manual contains information specific to the 7700 Series emulator.

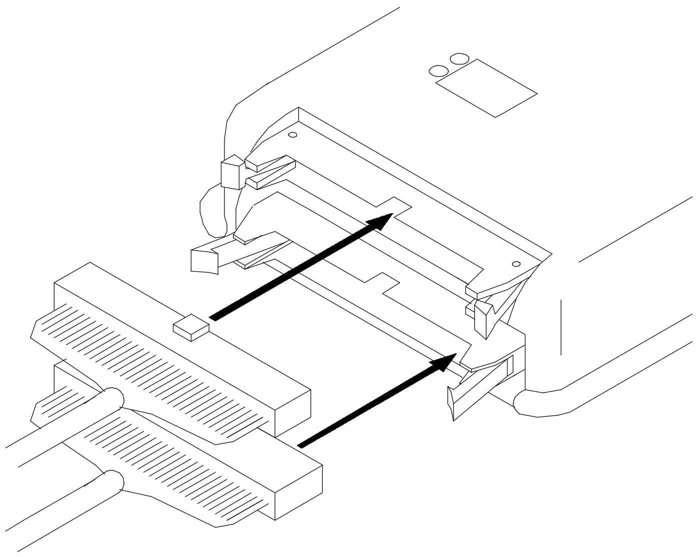


Figure 2-1. Connecting the Emulation Pod

A Look at the Sample Program

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

Using the various features of the emulator, we will show you how to load this program into emulation memory, execute it, monitor the program's operation with the analyzer, and simulate entry of different commands by using the "**M**emory **M**odify" emulation command.

Data Declarations

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

```
                .DP             0
                .DT             0
                .PUB            Init
                .PUB            Msgs
                .PUB            Cmd_Input
                .PUB            Msg_Dest

                .SECTION        BUFFER
;**********************************************************
; Command input byte.
;**********************************************************
Cmd_Input:      .BLKB           1
;**********************************************************
; Destination of the command messages.
;**********************************************************
Msg_Dest:       .BLKB           20H
                .BLKB           100H
Stack:

                .SECTION        TABLE
Msgs:
Msg_A:          .BYTE           'THIS IS MESSAGE A'
Msg_B:          .BYTE           'THIS IS MESSAGE B'
Msg_I:          .BYTE           'INVALID COMMAND'

                .SECTION        SAMPPROG
                .DATA           8
                .INDEX          16
;**********************************************************
; Set up the Stack Pointer.
;**********************************************************
Init:           LDX             #Stack
                TXS
                SEM
;**********************************************************
; Clear Previous command.
;**********************************************************
Clear_Input:    LDA             B,#00H
                STA             B,DT:Cmd_Input
;**********************************************************
; Read command input byte.  If no command has been entered,
; continue to scan for it.
;**********************************************************
Scan:           LDA             A,DT:Cmd_Input
                CMP             A,#00H
                BEQ             Scan

                .INDEX          8
;**********************************************************
; A command has been entered.  The destination area is
; cleared.
;**********************************************************
                SEP             X
Clear_Output:   LDX             #00H
                LDY             #20H
```

Figure 2-2. Sample Program Listing

```
Clear_Loop:     STA             B,DT:Msg_Dest,X
                INX
                DEY
                BNE             Clear_Loop
                .INDEX          16
;*********************************************************
; Check if the command entered is command A, command B,
; or invalid command.
;*********************************************************
                CLP             X
Process_Cmd:    CMP             A,#41H
                BEQ             Cmd_A
                CMP             A,#42H
                BEQ             Cmd_B
                BRA             Cmd_I
;*********************************************************
; Command A is entered.  A = the number of bytes in
; message A.  X = location of the message.  Jump to the
; routine which writes the message.
;*********************************************************
Cmd_A:          LDA             A,#11H
                LDX             #Msg_A
                BRA             Output
;*********************************************************
; Command B is entered.
;*********************************************************
Cmd_B:          LDA             A,#11H
                LDX             #Msg_B
                BRA             Output
;*********************************************************
; An invalid command is entered.
;*********************************************************
Cmd_I:          LDA             A,#0FH
                LDX             #Msg_I
;*********************************************************
; Message is written to the destination.  Y = location of
; the destination area.
;*********************************************************
Output:         LDY             #Msg_Dest
                MVN             0,0
;*********************************************************
; Go back and scan for next command.
;*********************************************************
                BRA             Clear_Input

                .END
```

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

Initialization

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

Reading Input

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

Processing Commands

When a command is entered, the **Clear_Output** routine clears the destination area. Then, the instructions from **Process_Cmd** to **Cmd_A** determine whether the command was "A", "B", or an invalid command.

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

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

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

The instructions at **Cmd_A, Cmd_B,** and **Cmd_I** each load register Accumulator A with the length of the message to be displayed and register index register X with the starting location of the appropriate message. Then, execution transfers to **Output** which writes the appropriate message to the destination location, **Msg_Dest**.

After the message is written, the program branches back to read the next command.

The Destination Area

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

Sample Program Assembly

The sample program is written for and assembled/linked with Mitsubishi RASM77 Assembler and LINK77 Linkage Editor.

The sample program was assembled with the following command:

```
C>rasm77 -s cmd_rds.a77 <RETURN>
```

Linking the Sample Program

The sample program can be linked with following command and generates the absolute file. The contents of "cmd_rds.lnk" linkage editor subcommand file is shown in Figure 2-3.

```
C>link77 @cmd_rds.lnk <RETURN>
```

```
cmd_rds
,
,SAMPPROG=C000 TABLE=C100 BUFFER=100
,-s -ms
```

Figure 2-3. Linkage Editor Command File

Note

To load a program into the emulator, both .hex and .sym file are needed. To generate .sym file, you need to specify **-s** option when assmble and link your program.

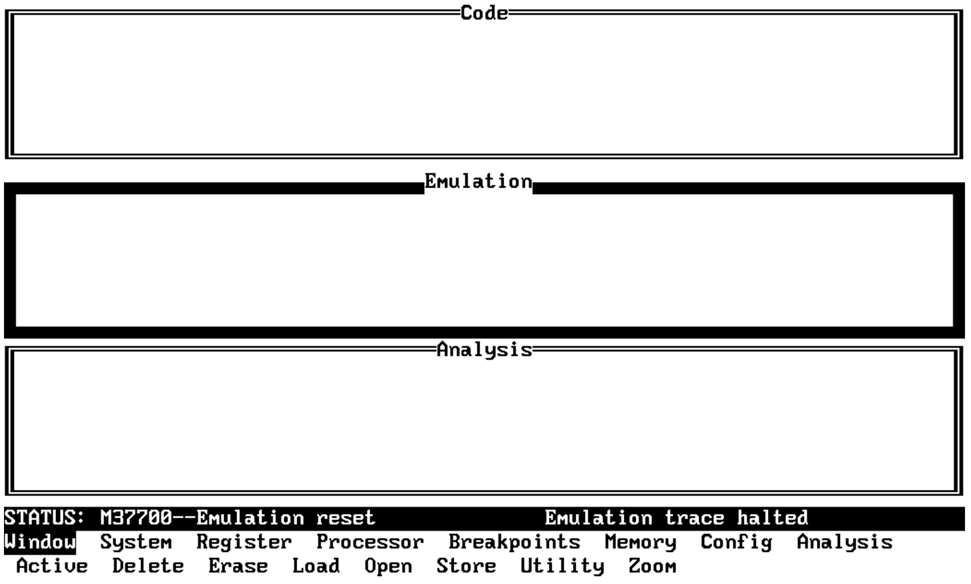
Starting Up the PC Interface

If you have set up the emulator device table and the **HPTABLES** shell environment variable as shown in the *HP 64700 Emulators PC Interface: Reference*, you can start up the 7700 Series PC Interface by entering the following command from the MS-DOS prompt:

```
    pcm377b <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, **pcm377b** 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 **ptm377b** for **pcm377b** in this command.) If this command is successful, you will see the display shown in Figure 2-4. Otherwise, you will be given an error message and returned to the MS-DOS prompt.



```
STATUS: M37700--Emulation reset            Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Active  Delete  Erase  Load  Open  Store  Utility  Zoom
```

Figure 2-4. PC Interface Display

Selecting PC Interface Commands

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

When a command or command option is highlighted, a short message describing that option is shown on the bottom line of the display.

Emulator Status

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

Modifying Configuration

Selecting Processor Type

The 7700 Series emulator can emulate various microprocessors of 7700 Series. Before starting emulation, you need to tell the emulator the name of microprocessor you are going to emulate. To do this, you must modify the pod configuration. Select:

```
Config, General
```

Use the arrow keys to move the cursor to the "Processor type?" field. Use the TAB key to select the processor type you are going to emulate. A list of valid processor types and corresponding processor names are listed in Table 2-1.

When your processor is not listed in Table 2-1, refer to chapter 4 of this manual for information on configuring the emulator.

Defining the Reset Value for Stack Pointer

Even though the 7700 Series emulator has a background monitor, it requires that you define a stack pointer. If no stack is defined, the background monitor has no place to store run address and other information required to respond to commands.

<chip_name>	Processor
7700M2	M37700M2-xxxFP
	M2AxxxFP
	M37701M2-xxxSP
	M2AxxxSP
7700M4	M37700M4-xxxFP
	M4AxxxFP
	M37701M4-xxxSP
	M4AxxxSP
7700S	M37700SFP
	SAFP
	M37701SSP
	SASP
7700S4	M37700S4FP
	S4AFP
	M37701S4SP
	S4ASP
7702M2	M37702M2-xxxFP
	M2AxxxFP
	M2BxxxFP
	M37703M2-xxxSP
	M2AxxxSP
	M2BxxxSP
7702M4	M37702M4-xxxFP
	M4AxxxFP
	M4BxxxFP
	M37703M4-xxxSP
	M4AxxxSP
	M4BxxxSP
7702M6	M37702M6BxxxFP
	M6LxxxFP
7702S1	M37702S1FP
	S1AFP
	S1BFP
	M37703S1SP
	S1ASP
	S1BSP
7702S4	M37702S4FP
	S4AFP
	S4BFP
	M37703S4SP
	S4ASP
	S4BSP
7704M2	M37704M2-xxxFP
	M2AxxxFP
	M37705M2-xxxSP
	M2AxxxSP
7704M3	M37704M3BxxxFP
7704M4	M37704M4BxxxFP
7704S1	M37704S1FP
	S1AFP
	M37705S1SP
	S1ASP
7710M4	M37710M4BxxxFP
7710S4	M37710S4BFP
7720S1	M37720S1FP
	S1AFP
7730S2	M37730S2FP
	S2AFP
	S2SP
	S2ASP
7732S4	M37732S4FP
	S4AFP
7780S	M37780STJ
	STFP
7781M4	M37781M4TxxxJ
	M4TxxxFP
7781E4	M37781E4TxxxJ
	E4TxxxFP
7795S	M37795SJ
	STJ
7796E4	M37796E4-xxxJ
	E4TxxxJ
	E4TxxxFP

Table 2-1. Processor Name for Emulator Configuration

Use the arrow keys to move the cursor to the "Reset value for Stack Pointer" field, type "**27f**", and press **Enter**.

The stack pointer value will be set to the stack pointer (SP) on entrance to the emulation monitor initiated RESET state (the "Emulation reset" status). To save the configuration, use the **Enter** key to exit the field in the last field. (The **End** key on Vectra keyboards moves the cursor directly to the last field.)

Mapping Memory

The 7700 Series emulator contains 128K/512K/1M/2M bytes of high-speed emulation memory (no wait states required) that can be mapped at a resolution of 256 bytes.

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

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 "Breaks on ROM writes?" configuration item is enabled (see the "Configuring the Emulator" chapter).

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

Caution

When you use the 7700 Series internal ROM, you must map memory space where internal ROM is located as emulation ROM.

You don’t have to map internal RAM as emulation RAM, since the emulator uses internal RAM of the emulation processor.

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. A part of linker load map listing for the sample program (cmd_rds.map) is shown in Figure 2-5.

```
MELPS 7700 LINKER V.2.00.00H   MAP FILE         Thu May 17 10:35:50 1990


SECTION           FILENAME          ATR.  TYPE  START    LENGTH   ALIGNMENT

BUFFER            cmd_rds.r77       REL   RAM   000100   000121
SAMPPROG          cmd_rds.r77       REL   ROM   00C000   000048
TABLE             cmd_rds.r77       REL   ROM   00C100   000031


GLOBAL LABEL INFORMATION

Cmd_Input       000100   Init            00C000   Msg_Dest        000101
Msgs            00C100

GLOBAL SYMBOL INFORMATION



TOTAL ROM SIZE 121 (79H) BYTES
TOTAL RAM SIZE 289 (121H) BYTES
```

Figure 2-5. Sample Program Load Map Listing

From the load map listing, you can see that the sample program occupies locations in three address ranges. The code area, which contains the opcodes and operands which make up the sample program, occupies locations C000 hex through C047 hex. The data area, which contains the ASCII values of the messages the program displays, is occupies locations C100 hex through C132 hex. The destination area, which contains the command input byte and the locations of the message destination and the stack, occupies locations 100 hex through 221 hex.

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:

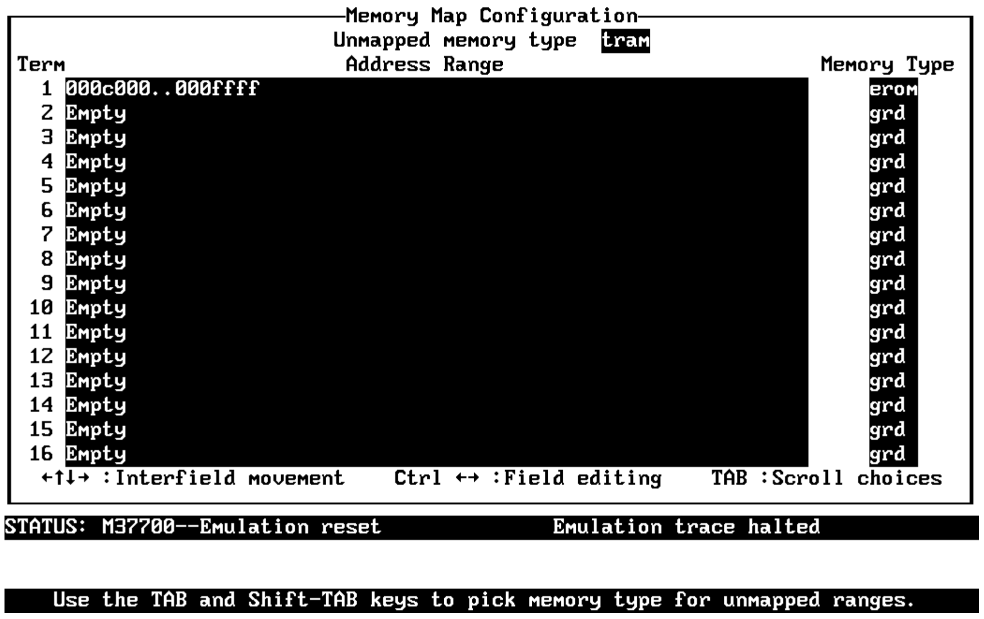
```
0c000..0ffff
```

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

Note

When you are going to emulate a processor which have no internal RAM, you need to map 100 hex through fff hex as emulation RAM for this tutorial.

To save your memory map, use 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 2-6.



```
                            Memory Map Configuration
                          Unmapped memory type  tram
Term                          Address Range                       Memory Type
  1 000c000..000ffff                                                  erom
  2 Empty                                                             grd
  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: M37700--Emulation reset              Emulation trace halted

    Use the TAB and Shift-TAB keys to pick memory type for unmapped ranges.
```

Figure 2-6. Memory Configuration Display

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

Note

The memory mapper re-assigns blocks of emulation memory after the insertion or deletion of mapper terms.

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** key to select the format of your absolute file. The emulator accepts absolute files in the following formats:

- MELPS 7700 Hex format.
- HP-MRI IEEE 695 absolute.
- HP64000 absolute.
- Raw HP64000 absoulte
- Intel hexadecimal.
- Tektronix hexadecimal.
- Motorola S-records.

For this tutorial, choose the MELPS 7700 Hex format.

Memory Type

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

Since emulation memory is mapped for sample program locations, you can enter either "emulation" or "both".

Force Absolute File Read

This option is available for MELPS 7700 Hex, HP-MRI IEEE 695, and HP64000 formats. It forces the file format readers to regenarate the emulator absolute file (.hpa) and symbol data base (.hps) before loading the code. Normally, these files are only regenarated whenever the file you specify (the output of your language tools) is newer than the emulator absolute file and symbol data base.

For more information, refer to the "Using the HP 64146 Format Reader" section in Appendix A.

Absolute File Name

For most formats, you enter the name of your absolute file in the last field. Type **cmd_rds.hex**, and press **Enter** to start the memory load.

Using Symbols

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 Refernce*.

Displaying Global Symbols

When you load MELPS 7700 Hex, HP-MRI IEEE 695, or HP64000 format absolute files into the emulator, the corresponding symbol database is also loaded.

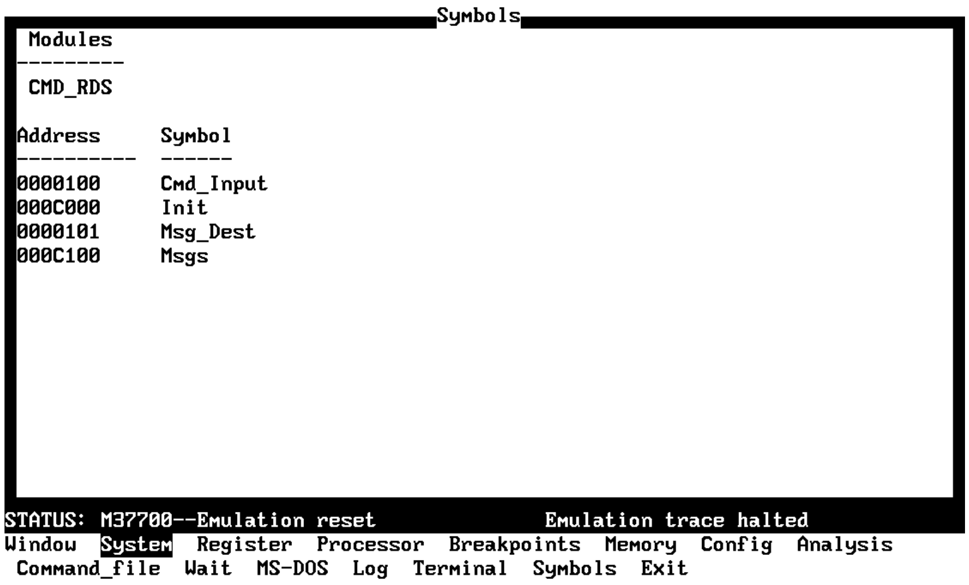
The symbols database can also be loaded with the "**S**ystem **S**ymbols **G**lobal **L**oad" command. This command is provided for situations where multiple absolute files are loaded into the emulator; it allows you to load the various sets of global symbols corresponding to the various absolute files. When global symbols are loaded into the emulator, information about previous global symbols is lost (that is, only one set of global symbols can be loaded at a time).

After global symbols are loaded, both global and local symbols can be used when entering expressions. Global symbols are entered as they appear in the source file or in the global symbols display.

To display global symbols, select:

```
System, Symbols, Global, Display
```

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



```
                                  Symbols
 Modules
---------

 CMD_RDS

Address     Symbol
----------  ------
0000100     Cmd_Input
000C000     Init
0000101     Msg_Dest
000C100     Msgs

STATUS: M37700--Emulation reset              Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Command_file  Wait  MS-DOS  Log  Terminal  Symbols  Exit
```

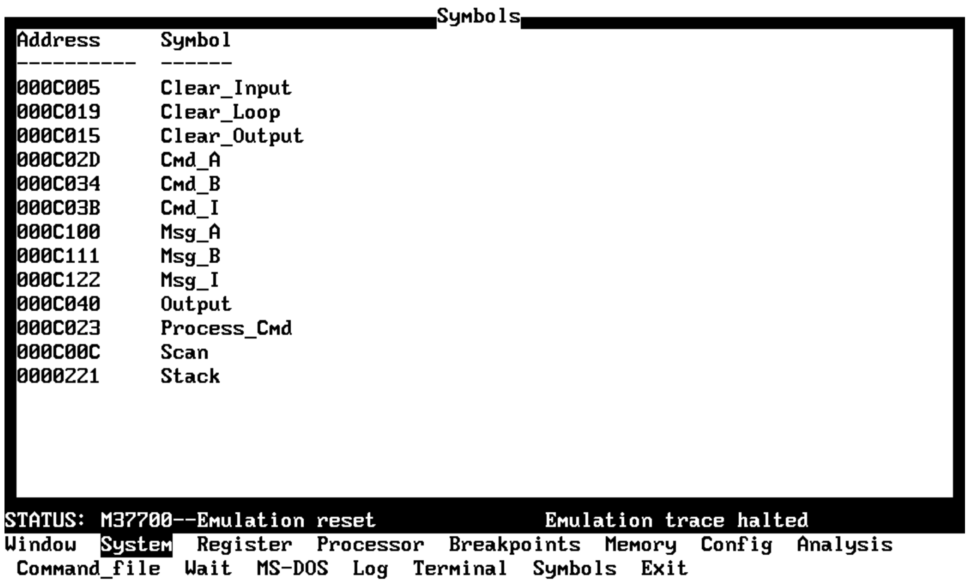
The global symbols display has two parts. The first parts lists all the modules that were linked to produce this object file. These module names are used by you when you want to refer to a local symbol, and are case-sensitive. THe second part of the display lists all global symbols in this module. These names can be used in measurement specifications, and are case-snesitive. For example, if you wish to make a measurement using the symbol **Cmd_Input**, you must specify **Cmd_Input**. The strings **cmd_input** or **CMD_INPUT** are not valid symbol names here.

Displaying Local Symbols

To display local symbols, select:

System, **S**ymbols, **L**ocal, **D**isplay

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



```
                                    Symbols
Address      Symbol
----------   ------
000C005      Clear_Input
000C019      Clear_Loop
000C015      Clear_Output
000C02D      Cmd_A
000C034      Cmd_B
000C03B      Cmd_I
000C100      Msg_A
000C111      Msg_B
000C122      Msg_I
000C040      Output
000C023      Process_Cmd
000C00C      Scan
0000221      Stack

STATUS: M37700--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
```

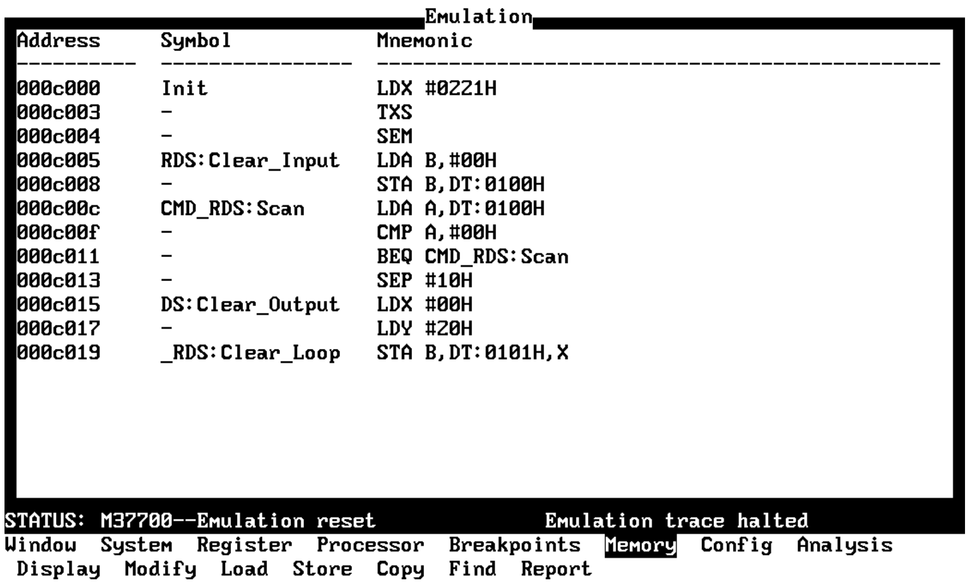
You can find more information on emulator symbol handling commands in the *Emulator PC Interface Reference*.

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
```

Now, you need to specify the values of M flag and X flag. Use the TAB key and select "**m0x0**". Move the cursor to the "address range" field, and enter the address range "**0c000..0c019**". (You could also specify this address range using symbols, for example, **"Init..CMD_RDS:Clear_Loop"** or **"Init..Init+19"**.) The emulation window automatically becomes the active window as a result of this command. You can press <CTRL>**z** to zoom the memory window. The resulting display follows.



```
                                    Emulation
Address     Symbol            Mnemonic
----------  ----------------  ---------------------------------------------
000c000     Init              LDX #0221H
000c003     -                 TXS
000c004     -                 SEM
000c005     RDS:Clear_Input   LDA B,#00H
000c008     -                 STA B,DT:0100H
000c00c     CMD_RDS:Scan      LDA A,DT:0100H
000c00f     -                 CMP A,#00H
000c011     -                 BEQ CMD_RDS:Scan
000c013     -                 SEP #10H
000c015     DS:Clear_Output   LDX #00H
000c017     -                 LDY #20H
000c019     _RDS:Clear_Loop   STA B,DT:0101H,X

STATUS: M37700--Emulation reset             Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Display  Modify  Load  Store  Copy  Find  Report
```

Notice that you need to specify the values of M flag and X flag. This is needed because the length of operands is variable according to M flag and X flag. The values of M flag and X flag you specified are used when the inverse-assembler start disassembling. Every time the inverse-assembler encounters an instruction which changes M flag and/or X flag (SEM, CLM, SEP X, etc...), the value set by the instruction is used to disassemble memory contents.

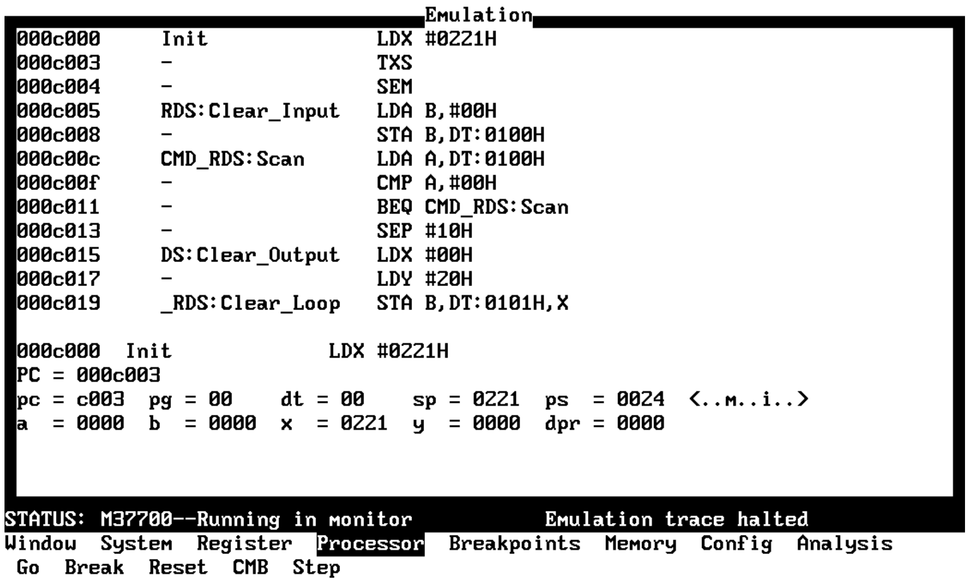
You can specify "**continue**" option to direct the emulator to use the current setting of M flag and X flag.

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, C000 hex. The executed instruction, the program counter address, and the resulting register contents are displayed as shown in the following listing.



```
                                   Emulation
000c000      Init              LDX #0221H
000c003      -                 TXS
000c004      -                 SEM
000c005      RDS:Clear_Input   LDA B,#00H
000c008      -                 STA B,DT:0100H
000c00c      CMD_RDS:Scan      LDA A,DT:0100H
000c00f      -                 CMP A,#00H
000c011      -                 BEQ CMD_RDS:Scan
000c013      -                 SEP #10H
000c015      DS:Clear_Output   LDX #00H
000c017      -                 LDY #20H
000c019      _RDS:Clear_Loop   STA B,DT:0101H,X

000c000  Init             LDX #0221H
PC = 000c003
pc = c003  pg = 00    dt = 00    sp = 0221  ps  = 0024  <..m..i..>
a  = 0000  b  = 0000  x  = 0221  y  = 0000  dpr = 0000



STATUS: M37700--Running in monitor           Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Go  Break  Reset  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.

To continue stepping through the program, you can select:

```
    Processor, Step, Pc
```

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

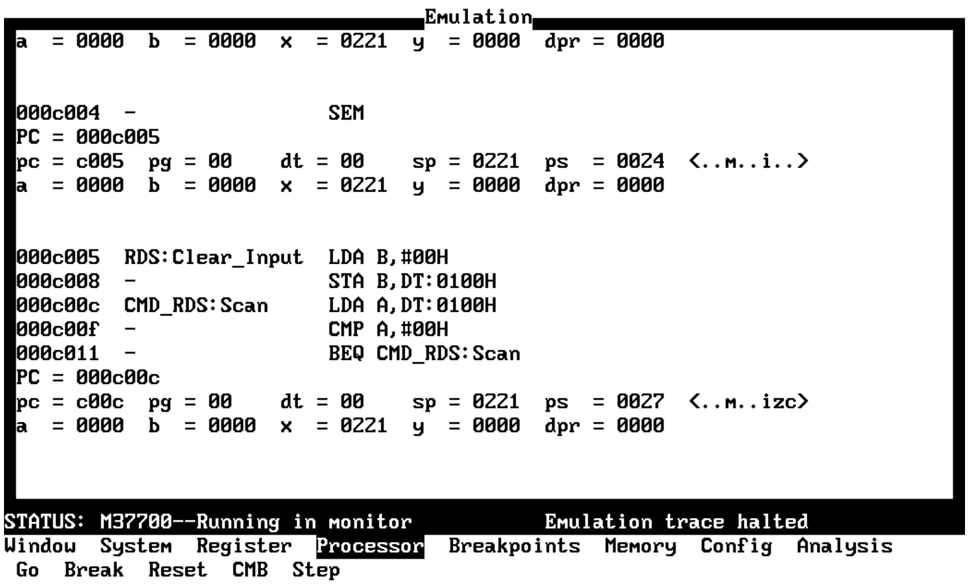
To repeat the previous command, you can press <CTRL>**r**.

Specifying a Step Count

If you wish to continue to step a number of 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 of times to step. Type **5** into the field, and press **Enter**. The resulting display follows.



```
                                    Emulation
a  = 0000  b  = 0000  x  = 0221  y  = 0000  dpr = 0000


000c004  -                 SEM
PC = 000c005
pc = c005  pg = 00    dt = 00    sp = 0221  ps  = 0024  <..m..i..>
a  = 0000  b  = 0000  x  = 0221  y  = 0000  dpr = 0000


000c005  RDS:Clear_Input  LDA B,#00H
000c008  -                STA B,DT:0100H
000c00c  CMD_RDS:Scan     LDA A,DT:0100H
000c00f  -                CMP A,#00H
000c011  -                BEQ CMD_RDS:Scan
PC = 000c00c
pc = c00c  pg = 00    dt = 00    sp = 0221  ps  = 0027  <..m..izc>
a  = 0000  b  = 0000  x  = 0221  y  = 0000  dpr = 0000



STATUS: M37700--Running in monitor           Emulation trace halted
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Go  Break  Reset  CMB  Step
```

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

Note

When the emulator performs step execution, all memory access is performed by byte access.

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 check if a command has been entered. To simulate the entry of a sample program command, you can modify the command input byte by selecting:

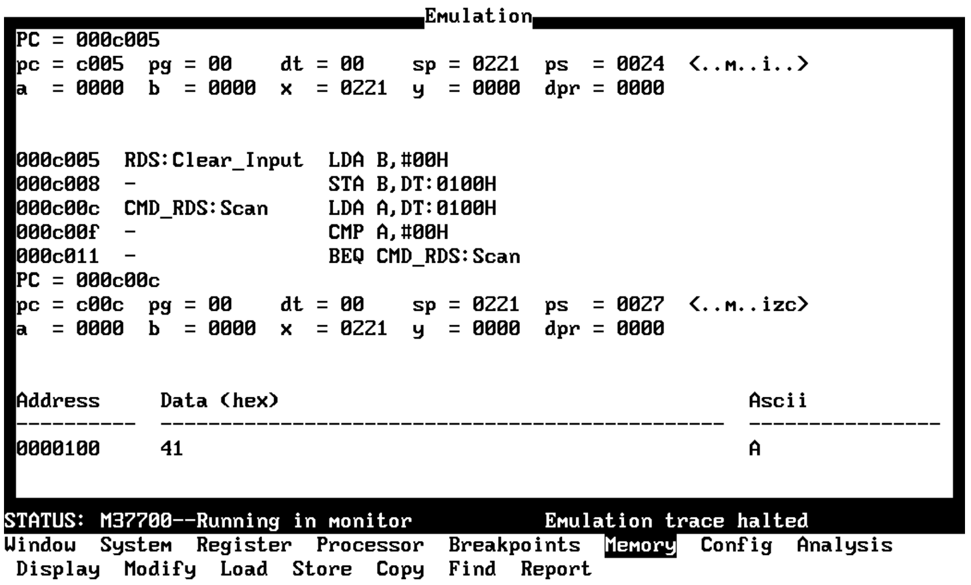
```
    Memory, Modify, Bytes
```

Now enter the address of the memory location to be modified, an equal sign, and new value of that location, for example, "**Cmd_Input=41**". (The **Cmd_Input** label was defined as a global symbol in the source file.)

To verify that 41 hex was indeed written to **Cmd_Input** (100 hex), select:

```
    Memory, Display, Bytes
```

Type the address "**100**" or the symbol **Cmd_Input**, and press **Enter**. This command will automatically activate the memory window. The resulting display is shown below.



```
                                   Emulation
PC = 000c005
pc = c005  pg = 00    dt = 00    sp = 0221  ps  = 0024  <..m..i..>
a  = 0000  b  = 0000  x  = 0221  y  = 0000  dpr = 0000


000c005  RDS:Clear_Input  LDA B,#00H
000c008  -                STA B,DT:0100H
000c00c  CMD_RDS:Scan     LDA A,DT:0100H
000c00f  -                CMP A,#00H
000c011  -                BEQ CMD_RDS:Scan
PC = 000c00c
pc = c00c  pg = 00    dt = 00    sp = 0221  ps  = 0027  <..m..izc>
a  = 0000  b  = 0000  x  = 0221  y  = 0000  dpr = 0000


Address     Data (hex)                                              Ascii
----------  ----------------------------------------------------    ----------------
0000100     41                                                      A

STATUS: M37700--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 emulator executing 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, and 41 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, 101 hex through 121 hex, and enter the data 20 hex, and 41 hex. The resulting information in the memory window shows you that the message was indeed written as it was supposed to have been.

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, and 42 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 (dependent on the type of instruction being executed and whether the processor is in a hold state).

Using Software Breakpoints

Software breakpoints are implemented in the 7700 Series emulator by replacing opcodes with BRK instruction as software breakpoint instruction. When you set a software breakpoint, the emulator replaces the opcode at the address specified with a BRK instruction. When the emulator executes this instruction in the user program, execution breaks to the monitor.

If the BRK instruction was not inserted as the result of a "**B**reakpoints" command (in other words, it is part of the user program), the "Undefined software breakpoint" message is displayed above the status line. Up to 32 software breakpoints may be defined.

Note

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

Note

Because software breakpoints are implemented by replacing opcodes with BRK instruction, you cannot define software breakpoints in target ROM. You can, however, use the Terminal Interface **cim** command to copy target ROM into emulation memory (see the *Terminal Interface: Reference* manual for information on the **cim** command).

Note

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.

Defining a Software Breakpoint

To define a breakpoint at the address of the **Cmd_I** label of the sample program (C03B hex), select:

```
    Breakpoints, Add
```

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

You can add multiple breakpoints in a single command by separating each one with a semicolon. For example, you could type "Cmd_A;Cmd_B" 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, Bytes
```

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

```
ALERT:  Software breakpoint: 000c03b
STATUS: M37700--Running in monitor
```

To continue program execution, select:

Processor, **G**o, **P**c

Displaying Software Breakpoints

To view the status of the breakpoint, select:

Breakpoints, **D**isplay

The resulting display shows that the breakpoint has been cleared.

Setting a Software Breakpoint

When a breakpoint is hit, it becomes disabled. To re-enable the software breakpoint, you can select:

Breakpoints, **S**et, **S**ingle

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 breakpoint 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, **C**lear, **S**ingle

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

Using the Analyzer

The 7700 Series emulation analyzer has 48 trace signals which monitor internal emulation lines (address, data, and status lines). Optionally, you may have an additional 16 trace signals which monitor external input lines. The analyzer collects data at each pulse of a clock signal, and saves the data (a trace state) if it meets a "storage qualification" condition.

Note

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

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 execution of **Cmd_A** routine. To do this, you must modify the default analysis specification by selecting:

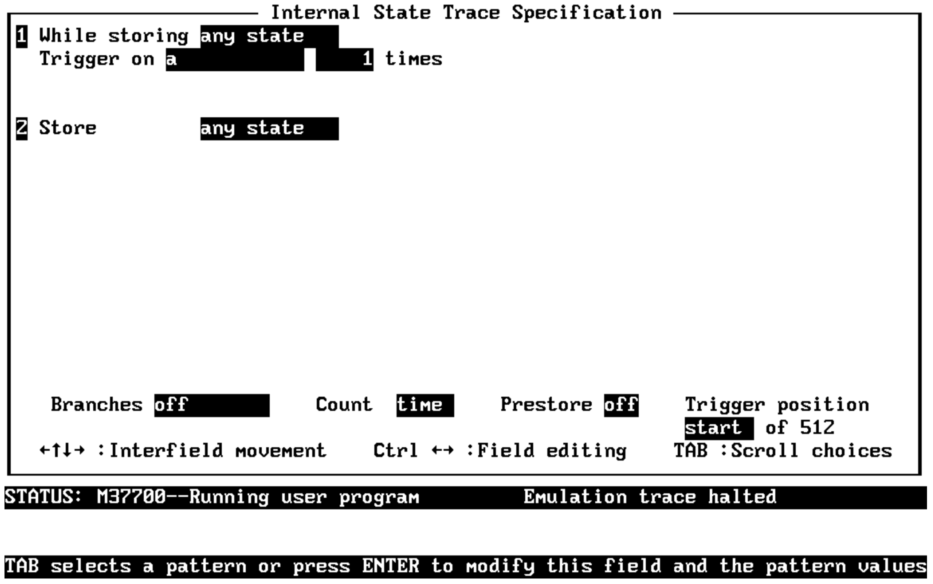
```
Analysis, Trace, Modify
```

The emulation analysis specification is shown. Use the right arrow key to move the cursor to 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 in the local symbol **Cmd_A** or address C02D hex.

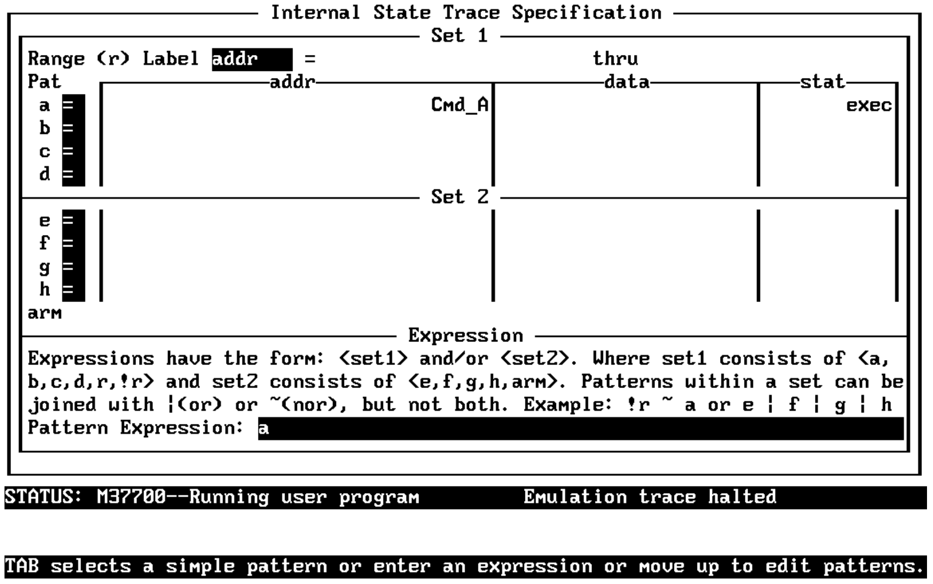
Move the cursor to the **stat** field of pattern **a**. Use the <TAB> key to select **exec** in the field.

The resulting analysis specification is shown in Figure 2-7 and 2-8. To save the new specification, use the **End Enter** to exit out of the field in the lower right corner. You’ll return to the trace specification. Press **End** to move to the trigger spec field. Press **Enter** to exit the trace specification.



```
                  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
                                                    start  of 512
 ←↑↓→ :Interfield movement   Ctrl ←→ :Field editing   TAB :Scroll choices
STATUS: M37700--Running user program        Emulation trace halted


TAB selects a pattern or press ENTER to modify this field and the pattern values
```

Figure 2-7. Modifying the Trace Specification



```
                  Internal State Trace Specification
                              Set 1
Range (r) Label addr    =                    thru
Pat  ─────────────addr─────────────────────────data─────────────stat──
 a =                          Cmd_A                               exec
 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: M37700--Running user program        Emulation trace halted


TAB selects a simple pattern or enter an expression or move up to edit patterns.
```

Figure 2-8. Modifying the Pattern 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 "A" by selecting:

```
Memory, Modify, Bytes
```

Enter "**Cmd_Input=41**". The status line now shows that the trace is complete.

Displaying the Trace

To display the trace, select:

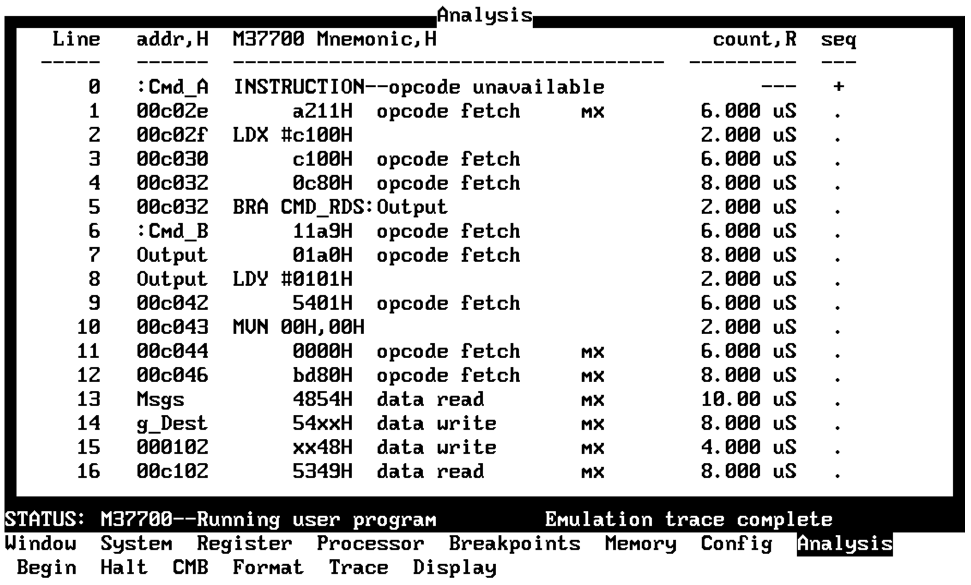
```
Analysis, Display
```

You are now given two fields in which to specify the states to display. Use the right arrow key to move the cursor to the "Ending state to display" field. Type "**60**" into the ending state field, press **Enter**, and use <CTRL>**z** to zoom the trace window.

Note

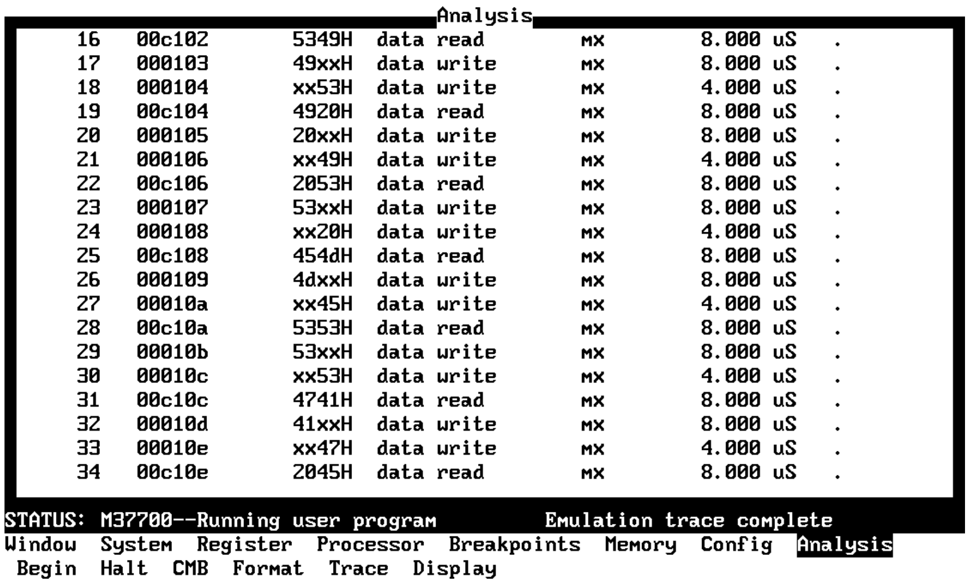
If you choose to dump a complete trace into the trace buffer, it will take a few minutes to display the trace.

Use the **Home** key to get to the top of the trace. The resulting trace is similar to the trace shown in the following display.



```
                                  Analysis
  Line    addr,H  M37700 Mnemonic,H                          count,R  seq
 -----    ------  ---------------------------------   ---------   ---
     0    :Cmd_A  INSTRUCTION--opcode unavailable                ---   +
     1    00c02e        a211H  opcode fetch     mx       6.000 uS   .
     2    00c02f  LDX #c100H                             2.000 uS   .
     3    00c030        c100H  opcode fetch              6.000 uS   .
     4    00c032        0c80H  opcode fetch              8.000 uS   .
     5    00c032  BRA CMD_RDS:Output                     2.000 uS   .
     6    :Cmd_B        11a9H  opcode fetch              6.000 uS   .
     7    Output        01a0H  opcode fetch              8.000 uS   .
     8    Output  LDY #0101H                             2.000 uS   .
     9    00c042        5401H  opcode fetch              6.000 uS   .
    10    00c043  MVN 00H,00H                            2.000 uS   .
    11    00c044        0000H  opcode fetch     mx       6.000 uS   .
    12    00c046        bd80H  opcode fetch     mx       8.000 uS   .
    13    Msgs          4854H  data read        mx       10.00 uS   .
    14    g_Dest        54xxH  data write       mx       8.000 uS   .
    15    000102        xx48H  data write       mx       4.000 uS   .
    16    00c102        5349H  data read        mx       8.000 uS   .

STATUS:  M37700--Running user program         Emulation trace complete
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Begin  Halt  CMB  Format  Trace  Display
```

Line 0 in the trace list above shows the state which triggered the analyzer. The trigger state is always on line 0. To list the next lines of the trace, press the **PgDn** or **Next** key.



```
                                  Analysis
    16    00c102        5349H  data read        mx       8.000 uS   .
    17    000103        49xxH  data write       mx       8.000 uS   .
    18    000104        xx53H  data write       mx       4.000 uS   .
    19    00c104        4920H  data read        mx       8.000 uS   .
    20    000105        20xxH  data write       mx       8.000 uS   .
    21    000106        xx49H  data write       mx       4.000 uS   .
    22    00c106        2053H  data read        mx       8.000 uS   .
    23    000107        53xxH  data write       mx       8.000 uS   .
    24    000108        xx20H  data write       mx       4.000 uS   .
    25    00c108        454dH  data read        mx       8.000 uS   .
    26    000109        4dxxH  data write       mx       8.000 uS   .
    27    00010a        xx45H  data write       mx       4.000 uS   .
    28    00c10a        5353H  data read        mx       8.000 uS   .
    29    00010b        53xxH  data write       mx       8.000 uS   .
    30    00010c        xx53H  data write       mx       4.000 uS   .
    31    00c10c        4741H  data read        mx       8.000 uS   .
    32    00010d        41xxH  data write       mx       8.000 uS   .
    33    00010e        xx47H  data write       mx       4.000 uS   .
    34    00c10e        2045H  data read        mx       8.000 uS   .

STATUS:  M37700--Running user program         Emulation trace complete
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Begin  Halt  CMB  Format  Trace  Display
```

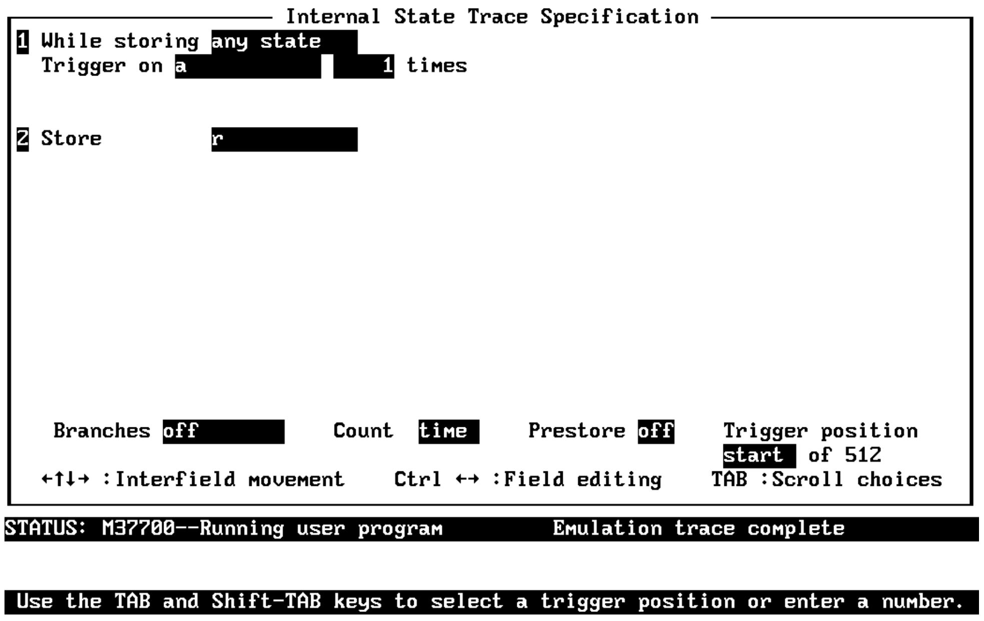
The resulting display shows the MVN instruction moves the "THIS IS MESSAGE A" message to the destination locations.

Using the Storage Qualifier

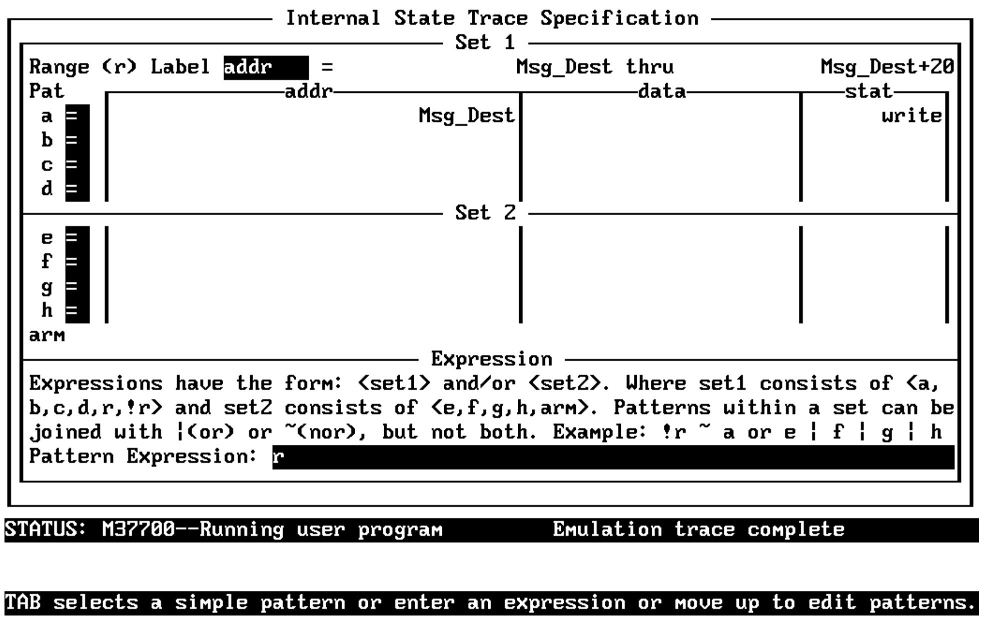
You can use storage qualifier to trace only specific conditions. Suppose that you would like to trace only states which write the messages to the **Msg_Dest** area. To accomplish this, select the following command, and modify the analysis specification as shown in the following displays.

```
    Analysis, Trace, Modify
```

Only write cycles to address **Msg_Dest** through **Msg_Dest+20h** will be stored in the trace buffer.



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


2 Store         r










   Branches off        Count  time     Prestore off    Trigger position
                                                       start  of 512
  ←↑↓→ :Interfield movement    Ctrl ←→ :Field editing    TAB :Scroll choices
STATUS: M37700--Running user program         Emulation trace complete


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



```
──────────────── Internal State Trace Specification ────────────────
──────────────────────────── Set 1 ────────────────────────────
Range (r) Label addr     =          Msg_Dest thru           Msg_Dest+20
Pat    ┌──────────────addr──────────────┬──────────data──────────┬───stat───┐
  a =  │                       Msg_Dest │                        │     write│
  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: r

STATUS: M37700--Running user program          Emulation trace complete

TAB selects a simple pattern or enter an expression or move up to edit patterns.
```

Start the trace, and modify the command input byte so that the program write a message to the destination area:

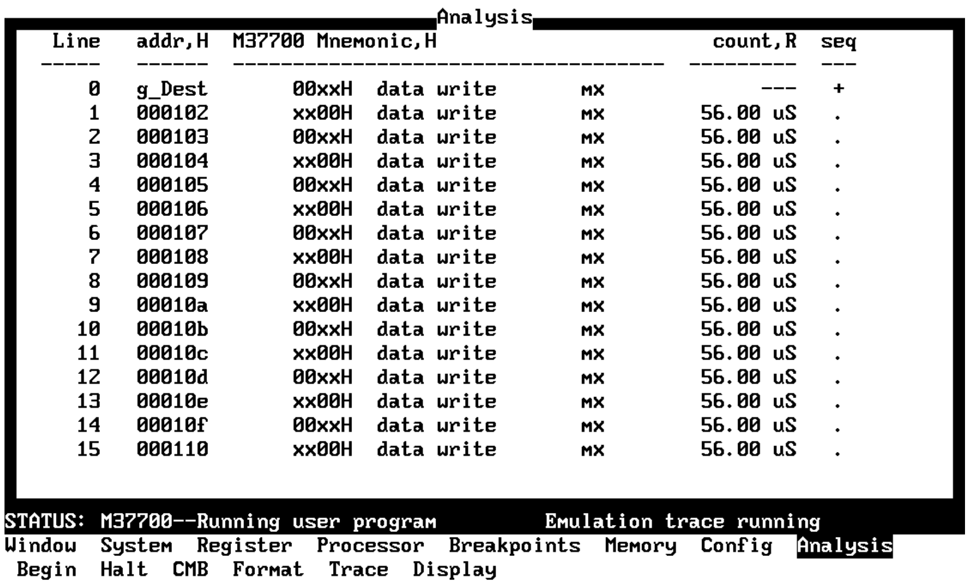
```
Analysis, Begin
Memory, Modify, Bytes
```

Enter "**Cmd_Input=41**".

Now display trace listing:

```
Analysis, Display
```

Type "**60**" into the "Ending state to display" field, and press **Enter** and **Home**. The resulting display shows the **Clear_Output** routine clears the destination area.



```
                                    Analysis
   Line    addr,H  M37700 Mnemonic,H                             count,R  seq
  -----    ------  -----------------------------------  ---------  ---
      0    g_Dest       00xxH  data write       mx            ---    +
      1    000102       xx00H  data write       mx        56.00 uS   .
      2    000103       00xxH  data write       mx        56.00 uS   .
      3    000104       xx00H  data write       mx        56.00 uS   .
      4    000105       00xxH  data write       mx        56.00 uS   .
      5    000106       xx00H  data write       mx        56.00 uS   .
      6    000107       00xxH  data write       mx        56.00 uS   .
      7    000108       xx00H  data write       mx        56.00 uS   .
      8    000109       00xxH  data write       mx        56.00 uS   .
      9    00010a       xx00H  data write       mx        56.00 uS   .
     10    00010b       00xxH  data write       mx        56.00 uS   .
     11    00010c       xx00H  data write       mx        56.00 uS   .
     12    00010d       00xxH  data write       mx        56.00 uS   .
     13    00010e       xx00H  data write       mx        56.00 uS   .
     14    00010f       00xxH  data write       mx        56.00 uS   .
     15    000110       xx00H  data write       mx        56.00 uS   .

STATUS: M37700--Running user program         Emulation trace running
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Begin  Halt  CMB  Format  Trace  Display
```

You may notice the status line shows that the trace is still running. You need to halt the trace to change the analysis specification. To halt the trace:

```
Analysis, Halt
```

7700 Series Analysis Status Qualifiers

The status qualifier "write" was used in the above example. The following analysis qualifiers may also be used with the 7700 Series emulator.

Qualifier	Status bits (40..47)	Description
backgrnd	x1xx xxxx	Background cycle
byte	xx1x 1x1x	Byte access
cpu	xx11 xxxx	CPU cycle
data	xx1x 10xx	Data access
dma	xx10 xxxx	DMA cycle
exec	xx11 01xx	Execution Cycle
fetch	xx11 11x1	Fetch cycle
foregrnd	x0xx xxxx	Foreground cycle
hold	xx01 xxxx	HOLD cycle
mx	1xxx xxxx	Value of MX signal
read	xx1x 1xx1	Read cycle
ref	xx00 xxxx	Refresh cycle
word	xx1x 1x0x	Word cycle
write	xx1x 1xx0	Write cycle

Restriction of the Analyzer

The following section describes the restrictions of the analyzer of 7700 Series emulator.

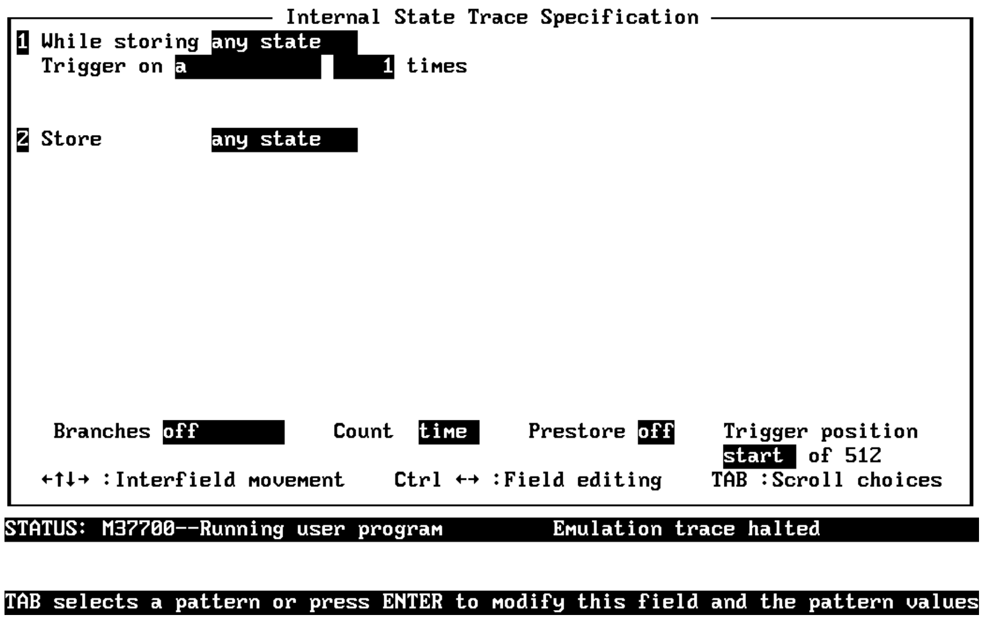
Trace of Internal RAM

The analyzer **cannot** trace data which is read from internal RAM or SFR. Such data always appears FF hex in the trace listing. This is because the emulator uses the internal RAM and SFR of the emulation processor to perform emulation. Data read from internal RAM or SFR does not appear on the data bus.

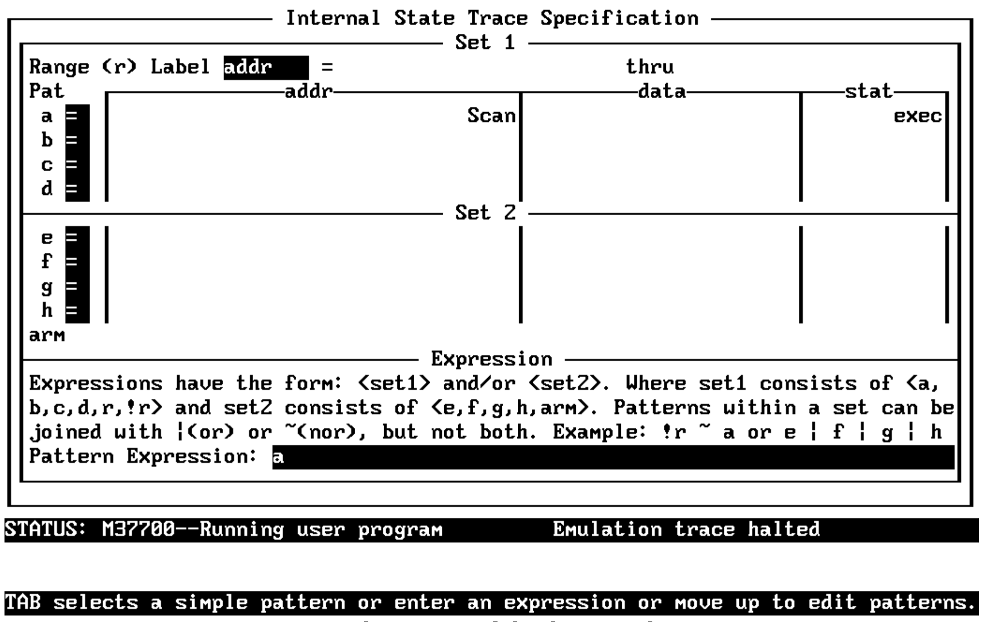
As an example, trace the accesses to the **Cmd_Input** area. To set up the analysis specifications, select:

```
Analysis, Trace, Reset
Analysis, Trace, Modify
```

Modify the analysis specifications as shown in the following displays.



```
───────────── 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
                                                        start  of 512
  ←↑↓→ :Interfield movement    Ctrl ←→ :Field editing    TAB :Scroll choices
STATUS: M37700--Running user program         Emulation trace halted

TAB selects a pattern or press ENTER to modify this field and the pattern values
```

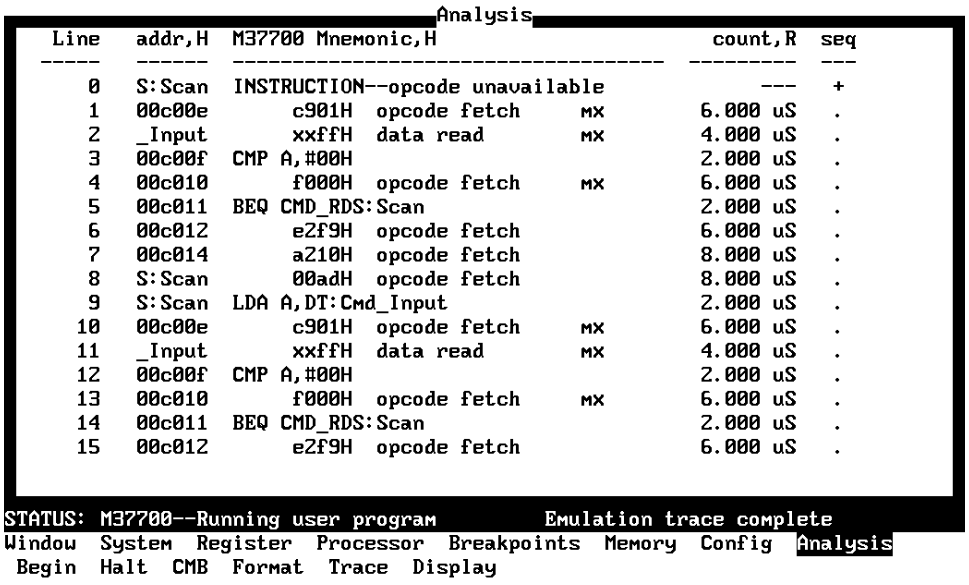


```
                    Internal State Trace Specification
                                  Set 1
 Range (r) Label addr   =                     thru
 Pat         ───────────addr──────────┬──────────data──────────┬──stat──
   a =                            Scan│                        │    exec
   b =                                │                        │
   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: M37700--Running user program         Emulation trace halted

TAB selects a simple pattern or enter an expression or move up to edit patterns.
```

Start the trace, and display trace listing:

Analysis, **B**egin
Analysis, **D**isplay

Press **Enter** three times. You will see the following display.



```
                                        Analysis
  Line   addr,H  M37700 Mnemonic,H                               count,R  seq
 -----   ------  ----------------------------------  ---------  ---
     0   S:Scan  INSTRUCTION--opcode unavailable               ---    +
     1   00c00e       c901H  opcode fetch     mx        6.000 uS   .
     2   _Input       xxffH  data read        mx        4.000 uS   .
     3   00c00f  CMP A,#00H                             2.000 uS   .
     4   00c010       f000H  opcode fetch     mx        6.000 uS   .
     5   00c011  BEQ CMD_RDS:Scan                       2.000 uS   .
     6   00c012       e2f9H  opcode fetch               6.000 uS   .
     7   00c014       a210H  opcode fetch               8.000 uS   .
     8   S:Scan       00adH  opcode fetch               8.000 uS   .
     9   S:Scan  LDA A,DT:Cmd_Input                     2.000 uS   .
    10   00c00e       c901H  opcode fetch     mx        6.000 uS   .
    11   _Input       xxffH  data read        mx        4.000 uS   .
    12   00c00f  CMP A,#00H                             2.000 uS   .
    13   00c010       f000H  opcode fetch     mx        6.000 uS   .
    14   00c011  BEQ CMD_RDS:Scan                       2.000 uS   .
    15   00c012       e2f9H  opcode fetch               6.000 uS   .

STATUS: M37700--Running user program         Emulation trace complete
Window  System  Register  Processor  Breakpoints  Memory  Config  Analysis
 Begin  Halt  CMB  Format  Trace  Display
```

As you can see in line 11 of the trace listing, data read from internal RAM (which should be 00 hex) appears FF hex.

Using a Command File

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

```
System, Log, Input, Enable
```

Enter command file name "cmd_rds.cmd", and press **Enter**. This sets up a file to record all commands you execute. The commands will be logged to the file cmd_rds.cmd in the current directory. You can then use this file as a command file to execute these commands automatically.

First, to load the sample program:

```
Memory, Load
```

Enter file format, memory type, and absolute file name, and press **Enter**.

To display global symbols:

```
System, Symbols, Global, Display
```

To display memory in mnemonic format:

```
Memory, Display, Mnemonic
```

Select "**m0x0**" as the MX value, enter address range "**Init..Init+19**", and press **Enter**.

Now we’re finished logging commands to the file. To disable logging:

```
System, Log, Input, Disable
```

The command file cmd_rds.cmd will no longer accept command input. The file looks like this:

```
ml
@7700
@Both
@no
@cmd_rds.hex
ssgd
mdm
@m0x0
@Init..Init+19
```

You can see a @ symbol in front of some lines in the file. These represents data values, as opposed to commands.

Let’s execute the command file "cmd_rds.cmd".

```
System, Command_file
```

Enter "cmd_rds.cmd", press **Enter**. Watch the command file commands execute. As you can see, the sequence of commands you entered is automatically executed.

Resetting the Emulator

To reset the emulator, select:

```
Processor, Reset, Hold
```

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

You can also 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 two different ways to exit the PC Interface. You can exit the PC Interface using the "locked" option which specifies that the current configuration will be present next time you start up the PC Interface. You can select this option as follows.

```
System, Exit, Locked
```

Symbols are lost when you use the "System Exit Locked" command; however, you can reload them (after you reenter the PC Interface) with the "System Symbols Global Load" command.

The other way to exit the PC Interface is with the "unlocked" option which specifies that the default configuration will be present the next time you start up the PC Interface. You can select this option with the following command.

```
System, Exit, Unlocked
```

Notes

3

"In-Circuit" Emulation

When you are ready to use the 7700 Series emulator in conjunction with actual target system hardware, there are some special considerations you should keep in mind.

- Installing the emulator probe.
- Properly configure the emulator.

We will cover the first topic in this chapter. For complete details on in-circuit emulation configuration, refer to the "Configuring the Emulator" chapter.

Installing the Target System Probe

Caution

DAMAGE TO THE EMULATOR CIRCUITRY MAY RESULT IF THESE PRECAUTIONS ARE NOT OBSERVED. The following precautions should be taken while using the 7700 Series emulator.

Power Down Target System. Turn off power to the user target system and to the 7700 Series 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 7700 Series emulator contains devices which are susceptible to damage by static discharge. Therefore, operators should take precautionary measures before handling the user plug to avoid emulator damage.

Note

When you use the emulator in-circuit, turn ON the target system first, then turn ON the emulator. Likewise, turn OFF the target system first, then turn OFF the emulator.

Note

Your target system **must** have a clock generation circuit. The HP 64146-61001 and HP 64146-61002 emulation pods cannot generate clock signal using a ceramic (or quartz crystal) resonator.

Installing the Target System Probe

1. Set up the switches inside the emulation pod. Refer to the manual provided with your emulation pod.
2. Remove the 7700 Series microprocessor from the target system socket. Note the location of pin 1 on the processor and on the target system socket.
3. Store the microprocessor in a protected environment (such as antistatic foam).
4. Install the target system probe into the target system microprocessor socket.
5. Turn on your target system, and then, turn on the emulator.

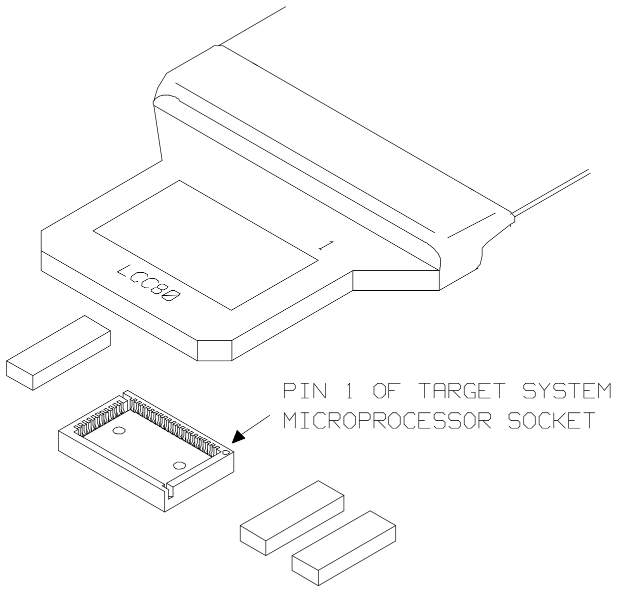


Figure 3-1. Installing the Probe to LCC80 Socket

When your target system uses 64 pin shrink DIP socket, use the adaptor as hown in Figure 3-2.

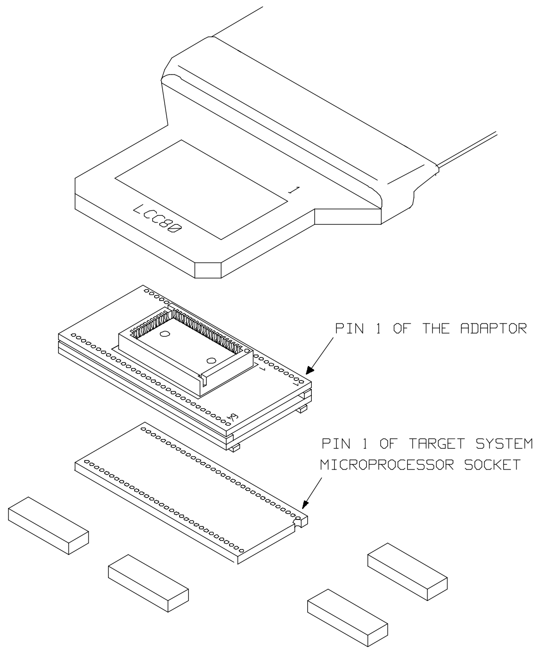


Figure 3-2. Installing the Probe to SDIP64 Socket

Configuring the Emulator

Introduction

The HP 64146A/B 7700 Series emulator is designed to help you in all stages of target system development. For instance, you can run the emulator out-of-circuit when developing and debugging your target system software and in-circuit when integrating your target system software with hardware. You can use the emulator's internal clock or your target system clock. Emulation memory can be used along with your target system memory, and it can be mapped as RAM or ROM. And, there are many more options available.

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

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.

Accessing the Emulator Configuration Options

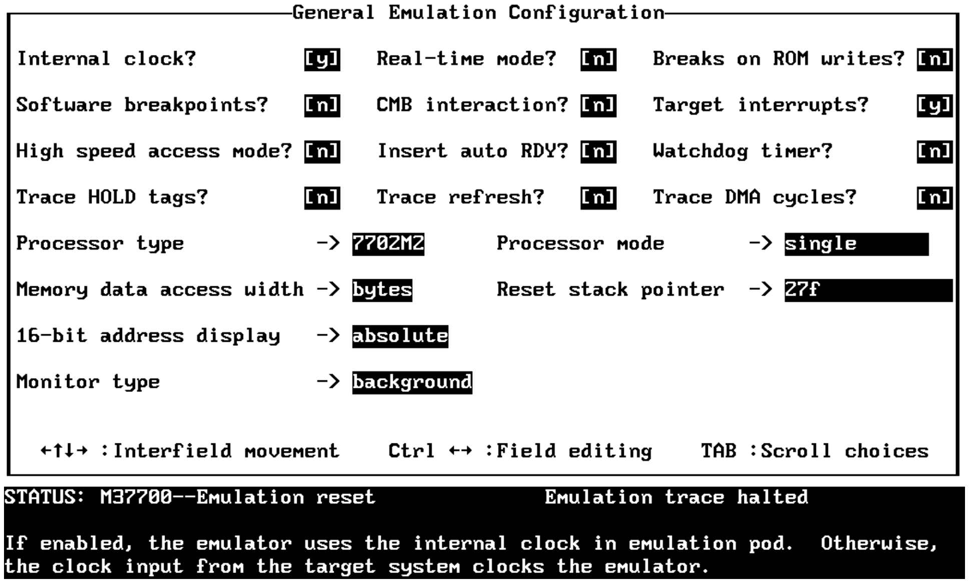
There are two configuration options for the 7700 Series emulator. You can select either:

```
    Config, General
```

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

Note

It is possible to use the System Terminal window to modify the emulator configuration. However, if you do this, some PC Interface features may no longer work properly. We recommend that you only modify the emulator configuration by using the options presented in the PC Interface.



```
                        General Emulation Configuration
Internal clock?          [y]   Real-time mode? [n]   Breaks on ROM writes? [n]

Software breakpoints?    [n]   CMB interaction? [n]  Target interrupts?    [y]

High speed access mode?  [n]   Insert auto RDY? [n]  Watchdog timer?       [n]

Trace HOLD tags?         [n]   Trace refresh?  [n]   Trace DMA cycles?     [n]

Processor type           -> 7702M2       Processor mode      -> single

Memory data access width -> bytes        Reset stack pointer -> 27f

16-bit address display   -> absolute

Monitor type             -> background

  ←↑↓→ :Interfield movement    Ctrl ←→ :Field editing    TAB :Scroll choices
STATUS: M37700--Emulation reset              Emulation trace halted

If enabled, the emulator uses the internal clock in emulation pod.  Otherwise,
the clock input from the target system clocks the emulator.
```

Figure 4-1. General Configuration Display

Internal Clock?

This configuration question allows you to select the emulator's clock source; you can choose either the internal clock oscillator or the target system clock.

yes	Selects the internal clock oscillator as the emulator clock source. The internal clock is provided from the emulation pod. In the case of HP 64146-61001 or HP 64146-61002 emulation pod, the clock speed is 1 MHz. When you use an emulation pod with clock faster than 16 MHz, you need to select the high speed access mode to run the emulator with no wait state. If the high speed access mode is not selected, the emulator requires one wait state.
no	Selects the clock input from the target system. You must use a clock input conforming to the specifications for the 7700 Series microprocessor. The HP 64146A/B emulator runs with no wait state with target system clock up to 16 MHz. When clock is faster than 16 MHz, you need to select the high speed access mode to run the emulator with no wait state. If the high speed access mode is not selected, the emulator requires one wait state.

Note

You can insert a wait state with one of the following method.

- Providing the /RDY from the target system.
- Configuring the emulator to generate the /RDY by answering "y" to the "**Insert auto RDY?**" question.

Note

When the external clock is selected, your target system must have a clock generation circuit. The HP 64146-61001 and HP 64146-61002 emulation pods cannot generate clock using a ceramic (or quartz crystal) resonator.

Real-Time Mode?

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

no	All commands, regardless of whether or not they require a break to the emulation monitor, are accepted by the emulator.
yes	When runs are restricted to real-time and the emulator is running the user program, all commands that cause a break (except "Processor Reset", "Processor Break", "Processor Go", and "Processor Step") are refused. For example, the following commands are not allowed when runs are restricted to real-time:

- Display/modify registers.
- Display/modify target system memory.
- Display/modify internal RAM or SFR.
- Load/store target system memory.

Caution

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

Breaks on ROM Writes?

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

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

Software Breakpoints?

This question allows you to enable or disable the software breakpoints feature. The 7700 Series emulator uses BRK instruction as software breakpoint.

no	The software breakpoints feature is disabled. This is specified by the default emulator configuration, so you must change this configuration item before you can use software breakpoints.
yes	Allows you to use the software breakpoints feature. When you set a software breakpoint, a BRK instruction will be placed at the address specified. When BRK instruction is executed, program execution will break into the monitor.

When you define (add) a breakpoint, software breakpoints are automatically enabled.

CMB Interaction?

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).

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 the `Processor, CMB, Go, ...` command is entered, 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 will also be enabled when the

```
Processor, CMB, Execute
```

command is entered.

Target interrupts?

This configuration allows you to specify whether or not the emulator responds to interrupt signals from the target system.

yes	The emulator will respond to interrupt signals from the target system.
no	The emulator will not respond to interrupt signals from the target system.

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

Note

You may need to set up switches inside the emulation pod to accept interrupts from the target system. Refer to the manual provided with your emulation pod.

High speed access mode?

When clock speed is faster than 16 MHz, the emulator can run with no wait state by selecting the "**high speed access mode**." If you don't select the high speed access mode, the emulator requires one wait state.

yes	Enables the high speed access mode of the emulator. In the high speed access mode:

- The emulator can run with no wait state up to 25 MHz.
- you can map the emulation memory only to the following address ranges.

Memory	Monitor	Available location
128K	Background	000000H-01F7FFH
128K	Foreground	000000H-01FFFFH
512K	Background	000000H-07F7FFH
512K	Foreground	000000H-07FFFFH
1M	Background	000000H-0FF7FFH
1M	Foreground	000000H-0FFFFFH
2M	Background	000000H-1FF7FFH
2M	Foreground	000000H-1FFFFFH

no Select the normal mode. In the normal mode:

- The emulator can run with no wait state up to 16 MHz.
- The emulator requires one wait state when clock is faster than 16 MHz.
- You can define up to 16 different map terms which can be placed wherever you like.

Note

You can insert a wait state with one of the following method.

- Providing the /RDY from the target system.
- Configuring the emulator to generate the /RDY by answering "y" to the "**Insert auto RDY?**" question.

Note

Changing this configuration will reset the memory map,

Insert auto RDY?

This configuration allows you to select whether or not the emulator introduces a wait state.

no	Disables RDY signal generation by the emulator. When clock is equal or slower than 16 MHz, always select this answer.
yes	Enables RDY signal generation by the emulator. When you enables this feature, the emulator inserts a wait state to all memory access.

Watchdog timer ?

This configuration allows you to enable/disable the watchdog timer interrupt.

no	Disable the watchdog timer interrupt. This may useful in early stage of your program development.
yes	Enables the watchdog timer interrupt.

Trace HOLD tags?

This configuration allows you to select whether or not the emulator traces HOLD/HLDA cycles.

no	Disables tracing HOLD/HLDA cycles.
yes	Enables tracing HOLD/HLDA cycles. HOLD/HLDA cycles will appear as one analysis trace line.

Trace refresh?

This configuration allows you to select whether or not the emulator traces refresh cycles by 7720 microprocessor.

no	Disables tracing refresh cycles by 7720 microprocessor.
yes	Enables tracing refresh cycles by 7720 microprocessor.

Trace DMA cycles?

This configuration allows you to select whether or not the emulator traces DMA cycles by 7720 microprocessor.

no	Disables tracing DMA cycles by 7720 microprocessor.
yes	Enables tracing refresh cycles by 7720 microprocessor. DMA cycles will appear as one analysis trace line.

Processor type

This configuration allows you to select the processor you are going to emulate. This item sets up the internal memory address, reset value for the stack pointer.

Select appropriate <chip name> listed in Table 4-1.

When your processor is not listed in Table 4-1, you need to set up the following configuration items from the System Terminal Window:

- cf irom
- cf iram
- cf isfr
- cf ipmr

<chip_name>	Processor
7700M2	M37700M2-xxxFP
	M2AxxxFP
	M37701M2-xxxSP
	M2AxxxSP
7700M4	M37700M4-xxxFP
	M4AxxxFP
	M37701M4-xxxSP
	M4AxxxSP
7700S	M37700SFP
	SAFP
	M37701SSP
	SASP
7700S4	M37700S4FP
	S4AFP
	M37701S4SP
	S4ASP
7702M2	M37702M2-xxxFP
	M2AxxxFP
	M2BxxxFP
	M37703M2-xxxSP
	M2AxxxSP
	M2BxxxSP
7702M4	M37702M4-xxxFP
	M4AxxxFP
	M4BxxxFP
	M37703M4-xxxSP
	M4AxxxSP
	M4BxxxSP
7702M6	M37702M6BxxxFP
	M6LxxxFP
7702S1	M37702S1FP
	S1AFP
	S1BFP
	M37703S1SP
	S1ASP
	S1BSP
7702S4	M37702S4FP
	S4AFP
	S4BFP
	M37703S4SP
	S4ASP
	S4BSP
7704M2	M37704M2-xxxFP
	M2AxxxFP
	M37705M2-xxxSP
	M2AxxxSP
7704M3	M37704M3BxxxFP
7704M4	M37704M4BxxxFP
7704S1	M37704S1FP
	S1AFP
	M37705S1SP
	S1ASP
7710M4	M37710M4BxxxFP
7710S4	M37710S4BFP
7720S1	M37720S1FP
	S1AFP
7730S2	M37730S2FP
	S2AFP
	S2SP
	S2ASP
7732S4	M37732S4FP
	S4AFP
7780S	M37780STJ
	STFP
7781M4	M37781M4TxxxJ
	M4TxxxFP
7781E4	M37781E4TxxxJ
	E4TxxxFP
7795S	M37795SJ
	STJ
7796E4	M37796E4-xxxJ
	E4TxxxJ
	E4TxxxFP

Table 4-1. <chip_name> for Emulator Configuration

You can get help messages for these commands by typing the following command in the System Terminal Window:

```
R> ? cf irom
R> ? cf iram
R> ? cf isfr
R> ? cf ipmr
```

Processor Mode

This configuration defines operation mode in which the emulator works.

single	The emulator will operate in single-chip mode.
expand_8bit	The emulator will operate in memory expansion mode with 8 bit data width.
expand_16bit	The emulator will operate in memory expansion mode with 16 bit data width.
proc_8bit	The emulator will operate in microprocessor mode with 8 bit data width.
proc_16bit	The emulator will operate in microprocessor mode with 16 bit data width.

Note

You may need to set up a switch inside the emulation pod in addition to this configuration. Refer to the manual provided with your emulation pod.

Memory data access width

This question allows you to specify the types of cycles that the emulation monitor use when accessing target system memory. When an emulation command requests the monitor to read or write target system memory locations, the monitor will either use byte or word instructions to accomplish the read/write.

bytes	Specifies that the emulator will access target system memory by byte accesses.
words	Specifies that the emulator will access target system memory by word accesses.

Reset stack pointer

This question allows you to specify the value to which the 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 16-bit address.

This configuration is automatically set up by the "Processor type ?" question. Reset value for Stack Pointer is set to the end of internal RAM. When the processor you select has no internal RAM, it is set to FFF hex.

This address should be defined in RAM area (internal RAM, target RAM or emulation RAM). When the emulator breaks to the monitor, the background monitor uses 5 bytes of stack area.

Caution

Without a stack pointer, the emulator is unable to make the transition to run state, step, or perform many other emulation functions.

16-bit address display

This configuration allows you to enable/disable symbol display in 16 bit addresses in mnemonic field.

absolute	16 bit addresses are displayed in absolute value. Symbols are displayed only in 24 bit addresses of mnemonic field.
symbol	Symbols are displayed both in 16 and 24 bit addresses of mnemonic filed. You need to answer the following question.

DT value for symbols

Since symbols have 24 bit value, you need to specify the value of the upper 8 bit which will be used to display symbols in 16 bit addresses. The value specified in this question will be combined with the 16 bit value in mnemonic field, and symbols are displayed using the value.

Monitor Type

background	Specify monitor type as **background monitor**. When you select background monitor, configuration question of foreground monitor address have no effect to emulator's operation.
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 "Memory Load" feature. The foreground monitor program must already assembled and linked with appropriate monitor start address specification.

To use the foreground monitor, follow below steps.

1. Decide which location the foreground monitor should be loaded.

2. Assemble and link the foreground monitor program along with the location you decided.

3. Configure the emulator as described in this chapter. (monitor type selection and monitor location).

4. Load the foreground monitor program into memory by "Memory Load" feature.

Foreground Monitor Address

You must specify foreground monitor start address when you select **"fg"** by above configuration question "Monitor type". This address specification must be same with the address specification when you assemble the foreground monitor program.

The address must be specified in a 16-bit hexadecimal address, and must be located on a 2K byte boundary other than internal RAM and Special Function Register area.

Note

You cannot use the step command when you are using a foreground monitor.

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.
- Memory map.
- Break conditions.

- Trigger configuration.
- Window specifications.

To store the current emulator configuration, select:

```
Config, Store
```

Enter the name of file to which the emulator configuration will be saved.

Loading an Emulator Configuration

If you have previously stored an emulator configuration and wish to re-load it into the emulator, select:

```
Config, Load
```

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

5

Using the Emulator

Introduction

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

This chapter shows you how to:

- Make Coordinated Measurements.
- Store the contents of memory into absolute files.

This chapter also discusses:

- Internal RAM and SFR of 7700 Series.
- Display or Modify the 7700 Series Special Function Registers.

Internal RAM and SFR

The emulator uses internal RAM of emulation processor to emulate user program. When you direct the emulator to display the contents of internal RAM (or SFR) area, the emulator breaks to the monitor and the monitor program reads the contents of memory. Therefore, execution of user program is suspended to perform your direction. However, you can configure the emulator so that write cycles are performed to both internal RAM (or SFR) and emulation memory. In this case, you can see the data written to emulation memory without suspending program execution.

To use this feature, you need to map these area to emulation RAM (eram). When you do this, you can display the contents of emulation memory without suspending user program execution. You still can display the contents of internal RAM by appending "@i" to address specification.

For example, to see the content of address 100 hex in internal RAM,

```
Memory, Display, Bytes
```

You can enter either of the following as address specification.

```
100    (This specifies emulation memory)
100@i  (This specifies internal RAM)
```

When you don't map the internal RAM and SFR area to emulation RAM, you can access the internal RAM and SFR without appending "@i".

Note

The contents of emulation memory is updated only when user program writes data to internal RAM or SFR. Therefore, the contents of emulation memory may be different from the actual value of internal RAM or SFR. Especially, you should pay a close attention when seeing flags of SFR.

Note

When you modify memory, the emulator breaks to the monitor, and writes data to internal RAM or SFR. Therefore, user program is suspended when modifying internal RAM or SFR.

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 some other instrument connected to the BNC connector.

This section 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.

Note

You must use the background emulation monitor to perform coordinated measurements.

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

/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 CMB run control interaction is enabled, all emulators are required to break to background upon reception of a false READY signal and will not return to foreground until this line is known to be in a true state.
/EXECUTE	Active low. This line serves as a global interrupt signal. Upon reception of an enabled /EXECUTE signal, each emulator is to interrupt whatever it is doing and execute a previously defined process, typically, run the emulator or start a trace measurement.

Running the Emulator at /EXECUTE

Before you can specify that the emulator run upon 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 "Enable 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 specify that the emulator begin executing a program upon reception of the /EXECUTE signal, select:

```
Processor, CMB, Go
```

At this point you may either select the current program counter, or you may select a specific address.

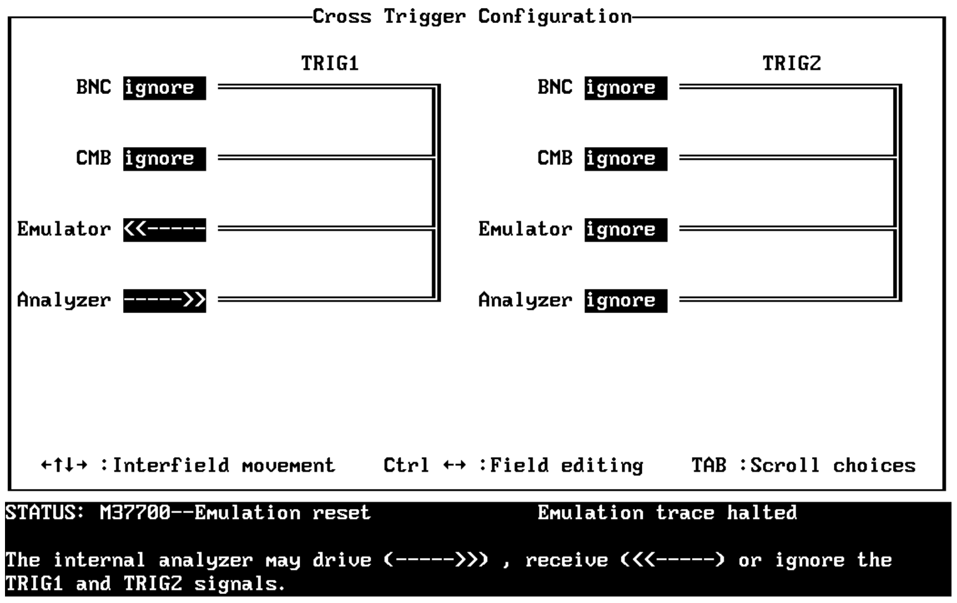
The command you enter is saved and is executed when the /EXECUTE signal becomes active. Also, you will see the message "ALERT: CMB execute; run started".

Breaking on the Analyzer Trigger

To cause emulator execution to break into the monitor when the analyzer trigger condition is found, you must modify the trigger configuration. To access the trigger configuration, select:

```
Config, Trigger
```

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



```
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: M37700--Emulation reset              Emulation trace halted

The internal analyzer may drive (----->>) , receive (<<-----) or ignore the
TRIG1 and TRIG2 signals.
```

Figure 5-1. Cross Trigger Condition

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, and use the **Tab** key to select the "----->>" arrow which shows that the analyzer is driving TRIG1. Next, move the cursor to the highlighted "Emulator" field and use the **Tab** key to select the arrow pointing towards the emulator (<<-----); this specifies that emulator execution will break into the monitor when the TRIG1 signal is driven. The trigger configuration display is shown in figure 5-1.

Note ☝

If your emulator is configured with external analyzer, a "Timing" cross trigger option will be displayed.

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
```

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, and optionally, you can include an extension of up to 3 alphanumeric characters.

Caution

The "Memory Store" 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

Using the HP 64146 Format Reader

A HP 64146 Format "reader" is provided with the PC Interface. The HP 64146 Format Reader converts a MELPS 7700 Hex format file into two files that are usable with the HP 64146A/B emulator. This means you can use available language tools to create MELPS 7700 Hex format absolute files, then load those files into the emulator using the 7700 Series PC Interface.

The HP 64146 Format Reader can operate from within the PC Interface or as a separate process. Operation from within the PC Interface is available if there is enough memory on your personal computer to run the PC Interface and the HP 64146 Format Reader simultaneously.

You can also run the reader as part of a "make file."

What the Reader Accomplishes

Using any MELPS 7700 Hex format absolute file and its symbol file (.hex file and .sym file), the HP 64146 Format Reader will produce two new files, an "absolute" file and an ASCII symbol file, that will be used by the 7700 Series PC Interface.

The Absolute File

During execution of the HP 64146 Format 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 64146 Format Reader contains global symbols, module names, local symbols, and, when using applicable development tools like 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 you specific language tools to generate symbol file (.sym file).

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

```
 module_name1
 module_name2
 ...
 module_nameN
global_symbol1                    address
global_symbol2                    address
...
global_symbolN                    address
|module_name|local_symbol1        address
|module_name|local_symbol2        address
...
|module_name|local_symbolN        address
|module_name|# 1234               address
```

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

The local symbols are scoped. When accessing the variable named "count" in the source file named "main.c", you would enter "MAIN:count". Notice that the module name of the source file "main.c" is "MAIN". See the following table.

Module Name	Variable Name	You Enter:
MAIN	count	MAIN:count
MAIN	line number 23	MAIN: line 23

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. Line numbers should appear in ascending order in both the line number itself and its associated address.

Note When the line number symbol is displayed in the emulator, it appears as a bracketed number. Therefore, the symbol "modname:# 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.

Location of the HP 64146 Reader Program

The HP 64146 Format 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 64146 Format Reader and PC Interface to operate properly. This is usually defined in "\autoexec.bat" file.

Using the HP 64146 Format Reader from MS-DOS

The command name for the HP 64146 Format Reader is **RD7700.EXE**. You can execute the HP 64146 Format Reader from the command line with the command:

```
C:\HP64700\BIN\RD7700 [-q] <filename>
<RETURN>
```

where:

[-q]	specifies the "quiet" mode. This option suppresses the display of messages.
<filename>	is the name of the file containing the MELPS 7700 Hex format absolute program.

The command

```
C:\HP64700\BIN\RD7700 TESTPROG.HEX
```

will therefore create the files "TESTPROG.HPA" and "TESTPROG.HPS".

Using the HP 64146 Format Reader from the PC Interface

The 7700 Series PC Interface has a file format option under the "**M**emory, **L**oad" command. After you select this option, the HP 64146 Format Reader will operate on the file you specify. After this completes successfully, the 7700 Series PC Interface will accept the absolute and symbol files produced by the Reader.

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

1. Start up the 7700 Series PC Interface.

2. Select "**M**emory, **L**oad". The memory load menu will appear.

3. Specify the file format as "7700". This will appear as the default file format.

4. Specify the name of a file in MELPS 7700 Hex format ("TESTFILE.HEX", for example,).

Note

The "<filename>.HPT" file is a temporary file used by the HP 64146 Format Reader to process the symbols.

Using a MELPS 7700 Hex format file and symbol file with the base name that you specify (TESTFILE, for example), the PC Interface performs the following:

- 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 7700 Series Format 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 MELPS 7700 Hex format file creation date/time, the HP 64146 Format 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/time for the MELPS 7700 Hex format file, 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 64146 Format Reader at the MS-DOS command line prompt, the HP 64146 Format Reader will always update the absolute and symbol files.

When the HP 64146 Format Reader operates on a file, a status message will be displayed indicating that it is reading a MELPS 7700 Hex format file. When the HP 64146 Format 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, then the PC Interface may run out of memory while attempting to create the database file used. If this condition occurs, you will need to exit the PC Interface and execute the program at the command prompt to create the files that are downloaded to the emulator.

Including RD7700 in a Make File

You may wish to incorporate the "RD7700" 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 "-.HPA" and "-.HPS" files are not current, the process of loading an MELPS 7700 Hex format file will automatically create them.

Specifications of HP 64146 Format Reader

The following are specifications of HP 64146 Format Reader.

- Label names and symbol names must be 15 and less characters in length.
- Up to 300 sections can be handled.
- If a label name contains "?", ".", or ":", they will be replaced with "_".

Note

The Format Reader uses both .hex file and .sym file. If .sym file is not found, the Format Reader will give you an error message. To generate .sym file, you need to specify **-s** option when you assemble and link your program.

Using the IEEE-695 Reader

An IEEE-695 MUFOM (Microprocessor Universal Format for Object Modules) "reader" comes with the PC Interface. The IEEE-695 reader converts an IEEE-695 format file into two files that are usable with the emulator. This means you can use available language tools to create IEEE-695 absolute files, then load those files into the emulator from the PC Interface.

The IEEE-695 reader can operate from within the PC Interface or as a separate process. You may need to execute the reader as a separate process if there is not enough memory on your personal computer to run the PC Interface and the reader simultaneously.

You can also run the reader as part of a "make file."

What the Reader Accomplishes

The IEEE-695 reader accepts as input an IEEE-695 format 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 IEEE-695 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 IEEE-695 reader contains global symbols, module names, local symbols, and, when using applicable development tools like 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 IEEE-695 absolute file.

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

```
 module_name1
 module_name2
 ...
 module_nameN
global_symbol1  address
global_symbol2  address
...
global_symbolN  address
|module_name|local_symbol1        address
|module_name|local_symbol2        address
...
|module_name|local_symbolN      address
|module_name|# 1234     address
```

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 scoped. 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 the following table.

Module Name	Function Name	Variable Name	You Enter
main.c	foo	count	main.c:foo.count
main.c	bar	count	main.c:bar.count

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. Line numbers should appear in ascending order.

Note

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

Location of the IEEE-695 Reader Program

The IEEE-695 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 IEEE-695 reader and PC Interface to operate properly. This is usually defined in the “\autoexec.bat” file.

Using the IEEE-695 Reader from MS-DOS

The command name for the IEEE-695 reader is **RIEEE695.EXE**. You can execute the IEEE-695 reader from the command line with the following command syntax:

```
C:\HP64700\BIN\RIEEE695 [-u] [-q] <filename>
<RETURN>
```

[-u]	Specifies that the first leading underscore of a symbol is not removed.
[-q]	Specifies the “quiet” mode. This option suppresses the display of messages.

<filename>	Specifies the name of the file containing the IEEE-695 absolute program.

Using the IEEE-695 Reader from the PC Interface

The 7700 Series PC Interface has a file format option under the "**M**emory **L**oad" command. After you select this option, the IEEE-695 reader will operate on the file you specify. After the reader completes successfully, the 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 the PC Interface.
2. Select "**M**emory **L**oad." The memory load menu will appear.
3. Specify the file format as "IEEE-695." This will appear as the default file format.
4. Specify the memory to be loaded (emulation, target, or both).
5. Specify a file in IEEE-695 format ("TESTFILE.ABS," for example). The file extension can be something other than ".ABS," but cannot be ".HPA," ".HPT," or ".HPS."

Note

The "<filename>.HPT" file is a temporary file used by the IEEE-695 reader to process the symbols.

Using the IEEE-695 file that you specify (TESTFILE.ABS, for example), the PC Interface performs the following:

- 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 IEEE-695 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 IEEE-695 file creation date/time, the IEEE-695 reader re-creates 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/time for the IEEE-695 file, the current absolute file, TESTFILE.HPA, is then loaded into the emulator.

Note

Date/time checking is only done within the PC Interface. When you run the IEEE-695 reader at the MS-DOS command line prompt, the reader will always update the absolute and symbol files.

When the IEEE-695 reader operates on a file, a status message will be displayed indicating that it is reading an IEEE-695 file. When the reader completes its processing, another message will be displayed indicating the absolute file is being loaded.

If the IEEE-695 Reader Won’t Run

If your program is very large, then 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 RIEEE695 in a Make File

You may want to incorporate the “RIEEE695” 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 “-.HPA” and “-.HPS” files are not current, loading an IEEE-695 file will automatically create them.

Index

A

absolute files
- .HPA created by HP 64146 Format Reader **A-1**
- .HPA created by IEEE-695 reader **A-7**
- loading **2-14**
- storing **5-6**

analysis begin **2-30**
analysis display **2-30**
analysis specification
- resetting the **2-28**
- saving **2-28**
- trigger condition **2-28**

analyzer
- status qualifiers **2-34**
- storage qualifier **2-32**
- using the **2-28**

anlyzer
- restriction **2-35**

ASCII symbol files
- .HPS created by HP 64146 Format Reader **A-1**
- .HPS created by IEEE-695 reader **A-7**

assemblers **2-12**
assembling the getting started sample program **2-7**

B

BNC connector **5-3**
break command **2-21, 2-25, 2-39**
break conditions **4-15**
breakpoints
- software **2-25**

breaks
- guarded memory accesses **2-11**
- on analyzer trigger **5-5**
- write to ROM **4-5**
- writes to ROM **2-11**

C

cautions
- filenames in the memory store command **5-7**
- installing the target system probe **3-2**

internal memory must be assigned as emulation memory **2-11**
real-time dependent target system circuitry **4-4**
the emulator cannot run without a stack pointer **4-13**
turn off the emulator before connecting the emulation pod **2-2**
characterization of memory **2-11**
cim, Terminal Interface command **2-26**
clock source
external **4-3**
internal **4-3**
clock source selection, emulator configuration **4-3**
CMB (coordinated measurement bus) **5-3**
enabling interaction **4-6**
execute signal while emulator is reset **2-39**
signals **5-3**
command file **2-37**
commands (PC Interface), selecting **2-9**
configuration
16 bit symbol display **4-13**
break processor on write to ROM **4-5**
clock selection **4-3**
enable CMB interaction **4-6**
enable high speed access mode **4-7**
enable software breakpoints **4-5**
enable target interrupts **4-7**
enable watchdog timer **4-9**
for sample program **2-9**
insert RDY at memory operation **4-9**
memory data access width **4-12**
monitor type **4-14**
processor mode **4-12**
processor type **4-10**
restrict to real-time runs **4-4**
stack pointer **4-13**
trace DMA cycle **4-10**
trace HOLD cycle **4-9**
trace refresh cycle **4-10**
configuration (emulator)
loading **4-16**
storing **4-15**
configuration (general)
accessing **4-2**

configuration(hardware), installing the emulator **2-2**
coordinated measurements
  break on analyzer trigger **5-5**
  definition **5-3**
  multiple emulator start/stop **4-6**
  run at /EXECUTE **5-4**
count, step command **2-22**

D
device table, emulator **2-8**
displaying the trace **2-30**

E
emulation memory
  mapping internal RAM and SFR area **5-2**
  RAM and ROM **2-11**
  size of **2-11**
Emulation pod **1-4**
  ordering information **1-4**
Emulation processor **1-4**
  ordering information **1-4**
Emulator
  before using **2-2**
  device table **2-8**
  memory mapper resolution **2-11**
  prerequisites **2-2**
  purpose **1-1**
  reset **2-39**
  status **2-9**
emulator configuration
  processor type **2-9**
  stack pointer **2-9**
Emulator features **1-5**
  analyzer **1-6**
  breakpoints **1-7**
  clock speed **1-5**
  coverage measurements **1-8**
  emulation memory **1-5**
  foreground and background monitor **1-6**
  high speed access mode **1-5**
  processor reset control **1-8**
  register display/modify **1-7**
  restrict to real-time runs **1-7**
  single-step processor **1-7**

Emulator limitations **1-9**
  Access to Internal RAM **1-9**
  displaying memory **5-3**
  DMA support **1-9**
  emulation command fails in stop/wait mode **1-10**
  stack must be in bank 0 **1-10**
  step command with foreground monitor **1-9**
  step fails when an interrupt exists **1-9**
  trace internal RAM **1-9**
  watch dog timer **1-9**
eram, memory characterization **2-11**
erom, memory characterization **2-11**
EXECUTE
  CMB signal **5-4**
  run at **5-4**
executing programs **2-24**
exiting the PC Interface **2-39**
external clock source **4-3**

F
find data in memory **2-24**
flag
  to display memory in mnemonic format **2-20**
foreground monitor address **4-15**

G
getting started
  prerequisites **2-2**
global symbols **2-15, 2-20**
grd, memory characterization **2-11**
guarded memory accesses **2-11**

H
hardware installation **2-2**
high speed access mode **1-5**
HP 64146 Format Reader **A-1**
  using with PC Interface **A-4**
  specifications **A-6**
HP 64146 Format Reader command (RD7700.EXE) **A-3**
HPT (temporary) file used by IEEE-695 reader **A-9**
HPTABLES environment variable **2-8**

I
IEEE-695 reader **A-6**
  using with PC Interface **A-9**
IEEE-695 reader command (RIEEE695.EXE) **A-8**
in-circuit emulation **4-1**

installation
  emulator probe into target system **3-3**
  hardware **2-2**
  software **2-2**
installing target system probe
  See target system probe
internal clock source **4-3**
internal RAM
  display without suspending user program **5-2**

L

line numbers **2-30**
link the sample program **2-7**
linkers **2-12**
load map **2-12**
loading absolute files **2-14**
local symbols **2-16, 2-26, A-2, A-7**
location of foreground monitor **4-15**
locked, PC Interface exit option **2-39**
logging of commands **2-37**

M

make file **A-1, A-6**
mapping
  internal RAM and SFR area **5-2**
mapping memory **2-11**
memory
  displaying in mnemonic format **2-19**
  mapping **2-11**
  modifying **2-23**
  re-assignment of emulation memory blocks **2-14**
  searching for data **2-24**
memory characterization **2-11**
memory mapping
  ranges, maximum **2-11**
modify configuration **2-9**
monitor program **1-6**
  background **1-7**
  foreground **1-7**
monitor type **4-14**

N

notes
  "Timing" option only with external analyzer **5-6**
  absolute file names for stored memory **5-6**
  CMB interaction enabled on execute command **4-6**

date checking only in PC Interface **A-5, A-10**
displaying complete traces **2-30**
HPT (temporary) file used by IEEE-695 reader **A-9**
map internal RAM for the tutorial **2-13**
memory access with step execution **2-22**
re-assignment of emul. mem. blocks by mapper **2-14**
register command **2-21**
setting software bkpts. while running user code **2-26**
setting up the pod to accept target interrupts **4-7**
software breakpoint locations **2-25**
software breakpoints and ROM code **2-26**
specify -s option when assemble, link your program **2-7**
symbol file is required to use the Format Reader **A-6**
symbols in mnemonic memory and trace displays **A-8**
target system must have clock generation circuit **3-3**
terminal window to modify emul. config. **4-2**
turn on the target system before turn on the emulator **3-2**
use required options to include symbols **A-2, A-7**

O out-of-circuit emulation **4-1**

P PC Interface
exiting the **2-39**
HP 64146 Format Reader **A-4**
IEEE-695 reader **A-9**
selecting commands **2-9**
starting the **2-8**
predefining stack pointer **2-9, 4-13**
prerequisites for getting started **2-2**
processor operation mode **4-12**
Purpose of the Emulator **1-1**

R RAM, mapping emulation or target **2-11**
READY, CMB signal **5-4**
real-time runs
restricting the emulator to **4-4**
register display/modify **2-21**
relocatable files **2-12**
reset **2-39**
resetting the analyzer specifications **2-28**
restrict to real-time runs
emulator configuration **4-4**
permissible commands **4-4**

target system dependency **4-4**
ROM
mapping emulation or target **2-11**
write to **2-11**
run at /EXECUTE **5-4**
running programs **2-24**

S
sample program, linking **2-7**
sample programs
for getting started **2-3**
saving analysis specifications **2-28**
searching for data in memory **2-24**
selecting PC Interface commands **2-9**
SFR
display without suspending user program **5-2**
simple trigger, specifying **2-28**
software breakpoints **2-25**
clearing **2-27**
defining (adding) **2-26**
displaying **2-27**
enabling **4-5**
setting **2-27**
software installation **2-2**
specifications
See analysis specification
stack pointer **1-10**
stack pointer,defining **2-9, 4-13**
starting the trace **2-30**
status line **2-9**
status qualifiers, 7700 Series **2-34**
step **2-20**
count specification **2-22**
stop mode **1-10**
storage qualifier **2-32**
supported microprocessors **1-3**
symbols **2-15**
.HPS file format **A-2, A-7**
global **2-20**
local **2-26, A-2, A-7**

T
target system dependency on executing code **4-4**
target system probe

installation **3-2**
installation procedure **3-3**
target system RAM and ROM **2-11**
temporary file used by IEEE-695 reader **A-9**
trace
analyzer signals **2-28**
description of listing **2-30**
displaying the **2-30**
starting the **2-30**
tram, memory characterization **2-11**
TRIG1, TRIG2 internal signals **5-5**
trigger **2-28**
breaking into monitor on **5-5**
specifying a simple **2-28**
trigger state **2-31**
TRIGGER, CMB signal **5-4**
trom, memory characterization **2-11**

U
undefined software breakpoint **2-25**
unlocked, PC Interface exit option **2-39**

W
wait mode **1-10**
write to ROM break **4-5**

Z
zoom, window **2-16, 2-19**