HP 64147

7750/51 Emulator Softkey Interface

User's Guide

HEWLETT PACKARD

HP Part No. 64147-97001
Printed in Japan
April 1995

Edition 1

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	**64147-97001, April 1995**

Using This Manual

This manual introduces you to the HP 64147A 7750/51 Series Emulator as used with the Softkey 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.

This manual does not:

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

Organization

Chapter 1	**Introduction to the 7750/51 Series Emulator.** This chapter briefly introduces you to the concept of emulation and lists the basic features of the 7750/51 Series emulator.
Chapter 2	**Getting Started.** This chapter shows you how to use emulation commands by executing them on a sample program. This chapter describes the sample program and how to: load programs into the emulator, map memory, display and modify memory, display registers, step through programs, run programs, set software breakpoints, and use the analyzer.
Chapter 3	**"In-Circuit" Emulation.** This chapter shows you how to install the emulation probe into a target system and how to use the "in-circuit" emulation features.
Chapter 4	**Configuring the Emulator.** This chapter shows you how to restrict the emulator to real-time execution, select a target system clock source, allow background cycles to be seen by the target system.
Appendix A	**Using the Foreground Monitor.** This appendix describes the advantages and disadvantages of foreground and background monitors and how to use foreground monitors.
Appendix B	**Using the Format Converter.** This appendix describes the usage of the file format converter.

Conventions

Example commands throughout the manual use the following conventions:

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

Notes

Contents

2 Getting Started

Illustrations

Notes

Introduction to the 7750/51 Series Emulator

Introduction

The topics in this chapter include:

- Purpose of the 7750/51 Series Emulator
- Features of the 7750/51 Series Emulator

Purpose of the 7750/51 Series Emulator

The HP 64147A 7750/51 Series Emulator is designed to replace the MELPS 7700/50/51 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 7700/50/51 Series microprocessor, but is a device that allows you to control the MELPS 7700/50/51 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 7700/50/51 Series is referred to as 7750/51 Series.

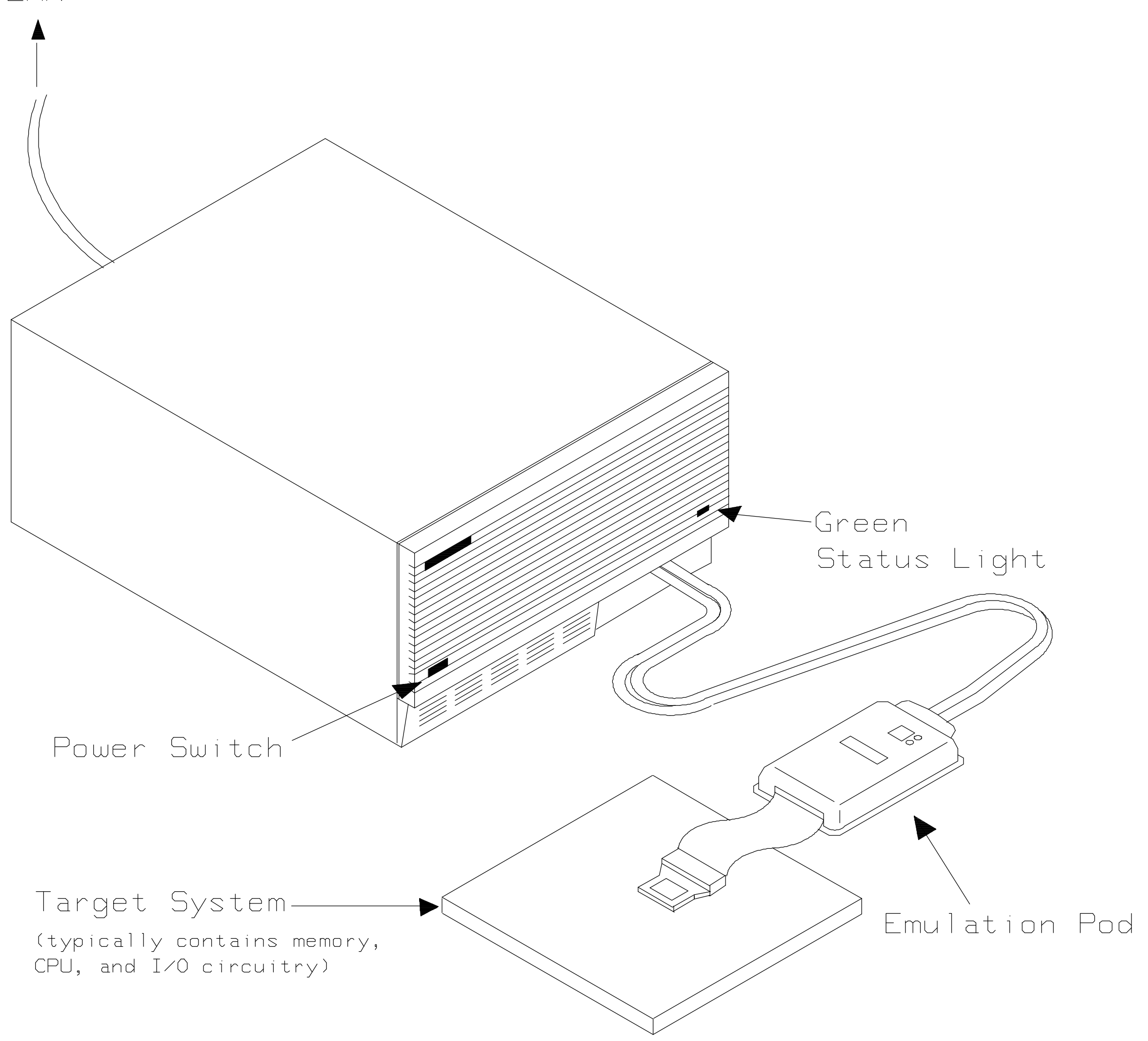


Figure 1-1. HP 64147 Emulator for MELPS 7750/51 Series

Supported Microprocessors

To emulate processors of 7750/51 Series, you need to purchase appropriate emulation pod and/or emulation processor. The HP 64147A 7750/51 Series emulator is provided with the following items.

- HP 64146-61002 emulation pod with M37702S1BFP emulation processor
- SDIP64 socket

The HP 64147A 7750/51 Series emulator can emulate M37702M2/4/6, M37703M2/4/6, M37702S1/4, M37703S1/4 and M37702M6L processors by using default emulation pod, HP 64146-61002. This emulation pod can be used with clock up to 25 MHz.

To emulate other processors by the HP 64147A 7750/51 emulator, you need to purchase appropriate emulation pod and/or emulation processor. Refer to the *Processor Support List for HP MELPS emulators* to determine if your microprocessor is supported or not.

The HP 64147A #001 emulator is provided with no emulation pod. You need to purchase appropriate emulation pod and emulation processor.

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

Features of the 7750/51 Series Emulator

This section introduces you to the features of the HP 64147A 7750/51 Series emulator. The chapters which follow show you how to use these features.

Clock Speed

The HP 64147A 7750/51 Series 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 up to 25MHz. 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.

The HP 64146-61002 emulation pod generate internal clock of 1/8/16/25 MHz. This emulation pods can be used with target system clock from 1 up to 25 MHz.

Emulation memory

The HP 64147A 7750/51 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 64147A 7750/51 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 64147A 7750/51 Series emulator is used with one of the following analyzers which allows you to trace code execution and processor activity.

- HP 64706 48-channel Emulation Bus Analyzer
- 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 7750/51 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.

Easy Products Upgrades

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

Limitations, Restrictions

Clock Speed

Maximum clock speed of HP 64147A 7750/51 emulator is 25MHz. This emulator does not support any operation with clock faster than 25MHz.

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 M37780/81/82/83/85/95/96 emulation pod.
- The processor is operating in the memory expansion or microprocessor mode with 8 bit external bus.

Step Command to Internal RAM

Step command to internal RAM area is not available.

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 microprocessor is in the stop or wait mode, emulation commands which access memory or registers will fail. In the case of using M37782/83/85 emulation pod, you need to reset the emulator to release stop or wait mode. And, in the case of using other emulation pod, you need to break the the emulator.

RDY/HOLD Input in Background Cycles

The 64147A M37750/51 emulator does not accept RDY/HOLD input while in background monitor. However, when you use M37780/81/82/83/85/95/96 emulation pod, M37750/51 emulator accepts RDY/HOLD input while in background monitor.

Accessing External Memory Area in SFR

When operation mode is memory expansion or microprocessor mode, there is external memory area in SFR. However, accessing to this area is not allowed.

High Speed Bus Mode

Always set bus mode as low speed bus mode, when you use M37751 emulation pod. HP 64147A 7750/51 emulator does not support high speed bus mode. Note that bus mode is automatically configured as high speed bus mode when you do **run from reset** command. Then, you need to re-configure bus mode as low speed bus mode before accessing SFR area.

RMPA Instruction

Disassembling in trace list may not be correct for next instruction of RMPA instruction. This failure will occur when RMPA instruction is repeated over about fifty times.

Stack Address

In some versions of 7720 microprocessor, the stack can be located in Bank FF. However, the HP 64147A 7750/51 Series emulator does not support the feature. The stack must be located in Bank 0.

Evaluation Chip

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

2

Getting Started

Introduction

This chapter will lead you through a basic, step by step tutorial that shows how to use the HP 64147A 7750/51 emulator with the Softkey Interface.

This chapter will:

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

This chapter will show you how to:

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

Before You Begin

Prerequisites

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

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

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

3. In addition, you should read and understand the concepts of emulation presented in the *Concepts of Emulation and Analysis* manual. The *Installation/Service* manual also covers HP 64700 system architecture. A brief understanding of these concepts may help avoid questions later.

4. Connected the emulator to the emulation probe as shown in Figure 2-1.

Caution

Turn off power of the 7750/51emulator before inserting the probe to the emulation pod avoid circuit damage.

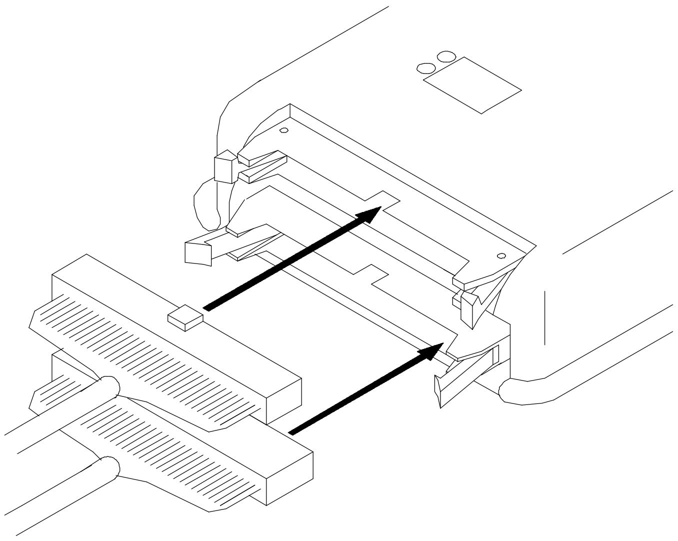


Figure 2-1. Connecting the Emulation Pod

A Look at the Demo Program

The demo program is *spmt_demo* consisting of source program *spmt_demo.c* and *scrt0.a77*.

Where is the spmt_demo Software?

The demo program is shipped with the Softkey Interface and may be copied from the following directory.

/usr/hp64000/demo/emul/hp64147

Compiling the Demo Program

The demo program is written for and compiled with the Mitsubishi NC77 Compiler Package. The demo program was compiled and assembled with the following commands.

```
$ nc77 -g -c spmt_demo.c <RETURN>
$ rasm77 -s scrt0.a77 <RETURN>
```

Linking the Demo Program

The following command was used to generate the absolute file. The contents of "spmt_demo.lnk" linkage editor subcommand file is shown in figure 2-2.

```
$ link77 @\spmt_demo.lnk<RETURN>
```

```
spmt_demo scrt0
,nc77lib.lib
,data=400 bss stack program=F000 init const
,-s -ms
```

Figure 2-2. Linkage Editor Subcommand File

Generate HP Absolute file

To generate HP Absolute file for the Softkey Interface, you need to use "**m77cnvhp**" absolute file format converter program. The m77cnvhp converter is provided with HP 64147 Softkey Interface. To generate HP Absolute file, enter following command:

```
$ m77cnvhp spmt_demo <RETURN>
```

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

Note

If you make the absolute file as IEEE-695 format, you don’t need to convert the file to HP Absolute file. HP 64147A Softkey Interface can load IEEE-695 format without any conversion.

Note

In the case of MELPS 7700 Hex format file, you must specify **-g** option to compiler and **-s** option to assembler/linker to generate local symbol information.

Entering the Softkey Interface

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

From the HP-UX Shell

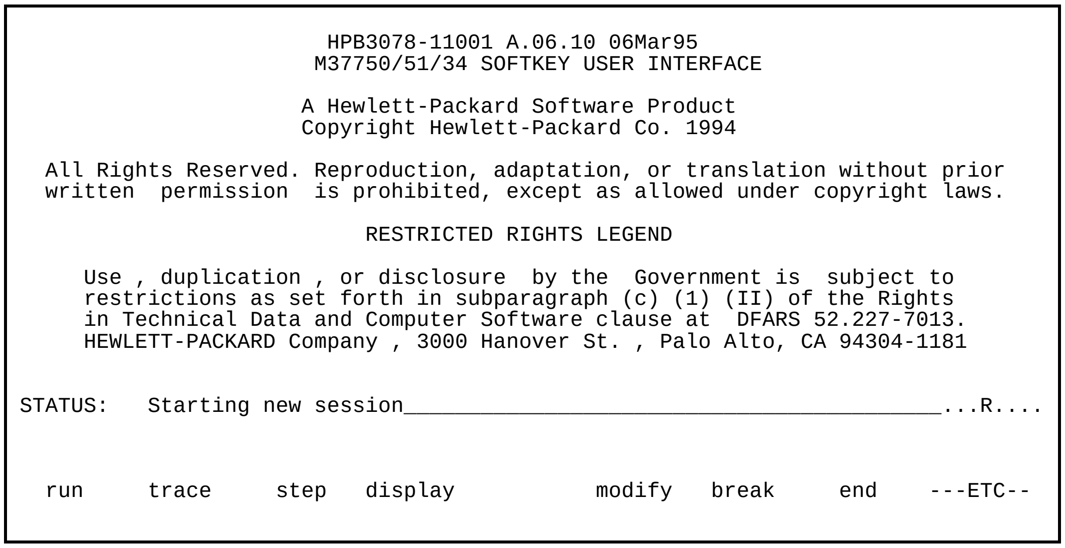
If **/usr/hp64000/bin** is specified in your PATH environment variable, you can also enter the Softkey Interface with the following command.

```
$ emul700 -u skemul <emul_name> <RETURN>
```

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

```
#--------+------------+-----------+------------------------------------------
# Channel|  Logical   | Processor | Remainder of Information for the Channel
#  Type  |   Name     |   Type    | (IP address for LAN connections)
#--------+------------+-----------+------------------------------------------
   lan:      7750        m37750     21.17.9.143
```

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



```
                        HPB3078-11001 A.06.10 06Mar95
                       M37750/51/34 SOFTKEY USER INTERFACE

                      A Hewlett-Packard Software Product
                      Copyright Hewlett-Packard Co. 1994

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

                           RESTRICTED RIGHTS LEGEND

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


STATUS:   Starting new session__________________________________________...R....



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

Figure 2-3. Softkey Interface Display

Configure the Emulator for Examples

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

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

```
    modify configuration <RETURN>
```

Answer to the series of questions as below.

```
Micro-processor clock source? internal <RETURN>
Enter monitor after configuration? yes <RETURN>
Restrict to real-time runs?  no <RETURN>
Reconfigure emulator?  yes <RETURN>
Processor type ?  <chip_name> <RETURN>
```

Select the <chip_name> you are going to emulate. Appropriate <chip_name> is listed in *Processor Support List for HP MELPS emulators*. If your processor is not listed in this guide, refer to chapter 4 of this manual for information on configuring the emulator.

```
Processor mode ?  single <RETURN>
Modify reset value for Stack Pointer?  no <RETURN>
Modify memory configuration?  yes <RETURN>
Is speed of input clock faster than 16MHz?  no <RETURN>
Monitor type?  backgnd <RETURN>
```

Now you should be facing memory mapping screen. One mapper term must be specified for the demo program. Enter the following line to map the program code.

```
    delete 1 <RETURN>
    0h thru 0ffffh emulation  ram <RETURN>
    end <RETURN>
Modify emulator pod configuration?  no <RETURN>
Modify debug/trace options?  no <RETURN>
Modify simulated I/O configuration?  no <RETURN>
Modify interactive measurement specification?  no <RETURN>
```

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

```
Configuration file name?  spmt_demo <RETURN>
```

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

On-Line Help

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

Softkey Driven Help

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

```
? system_commands <RETURN>
```

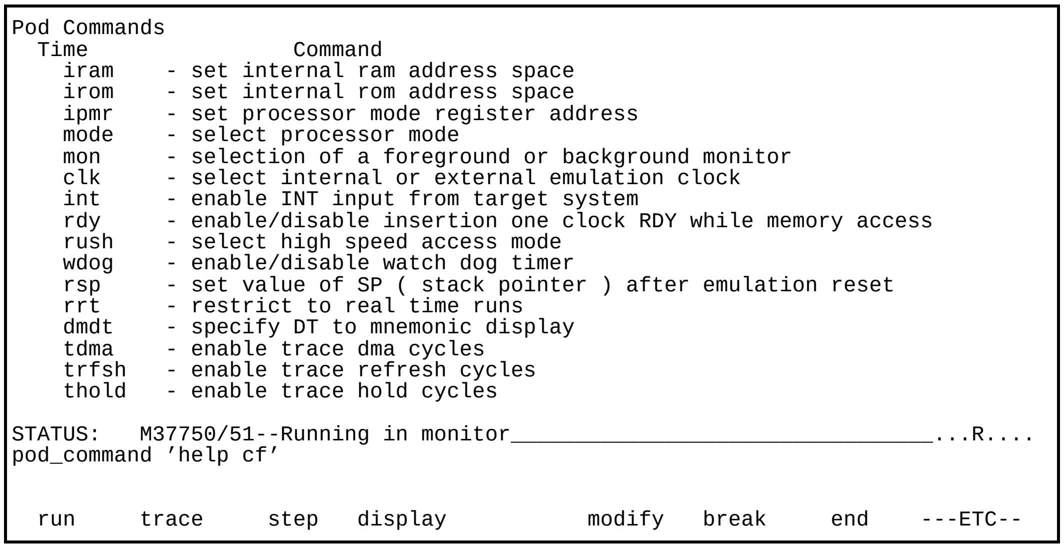
```
---SYSTEM COMMANDS & COMMAND FILES---

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

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

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

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


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

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

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

Pod Command Help

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

> ***display pod_command*** `<RETURN>`
> ***pod_command*** `’help cf’ <RETURN>`



```
Pod Commands
  Time                Command
    iram    - set internal ram address space
    irom    - set internal rom address space
    ipmr    - set processor mode register address
    mode    - select processor mode
    mon     - selection of a foreground or background monitor
    clk     - select internal or external emulation clock
    int     - enable INT input from target system
    rdy     - enable/disable insertion one clock RDY while memory access
    rush    - select high speed access mode
    wdog    - enable/disable watch dog timer
    rsp     - set value of SP ( stack pointer ) after emulation reset
    rrt     - restrict to real time runs
    dmdt    - specify DT to mnemonic display
    tdma    - enable trace dma cycles
    trfsh   - enable trace refresh cycles
    thold   - enable trace hold cycles

STATUS:   M37750/51--Running in monitor_________________________________...R....
pod_command ’help cf’


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

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

Note

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

> ***pod_command keyboard***

Loading Absolute Files

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

- HP absolute file format
- IEEE-695 file format

The "load" command has no special options for loading different absolute file formats; instead, the contents of the file are examined to determine the format being used. If you wish to load only that portion of the absolute file that resides in memory mapped as emulation RAM or ROM, use the "load emul_mem" syntax. If you wish to load only the portion of the absolute file that resides in memory mapped as target RAM, use the "load user_mem" syntax. If you want both emulation and target memory to be loaded, do not specify "emul_mem" or "user_mem". For example:

```
load spmt_demo <RETURN>
```

Note

When loading a program if the status line shows,

```
"ERROR:      No absolute file, No database:
spmt_demo
```

you may NOT be in the directory that your program is in. To find out what directory you are in, enter:

```
! pwd <RETURN>
```

The "!" allows you to use an HP-UX shell command. To move into the correct directory, enter:

```
cd <directory path>  <RETURN>
```

You can also specify the pathname where your program resides. For example, you could enter:

```
load
/usr/hp64000/demo/emul/hp64147/spmt_demo
<RETURN>
```

Displaying Symbols

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

Global

To display global symbols, enter the following command.

display global_symbols <RETURN>

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

```
Global symbols in spmt_demo.X
Procedure symbols
Procedure name _________________ Address range __ Segment _____________ Offset
_apply_controll                 00F773 - 00F847                           0773
_apply_producti                 00F4FF - 00F5C3                           04FF
_calculate_answ                 00F848 - 00F8F8                           0848
_clear_buffer                   00F2DB - 00F34F                           02DB
_endcommand                     00FA8B - 00FAA2                           0A8B
_format_result                  00F5C4 - 00F638                           05C4
_get_next_token                 00F6D6 - 00F772                           06D6
_initialze                      00F639 - 00F6D5                           0639
_input_line                     00F000 - 00F055                           0000
_lookup_token                   00F350 - 00F3EC                           0350
_main                           00FAA3 - 00FB3A                           0AA3
_math_library                   00F1F0 - 00F279                           01F0
_move_byte                      00F056 - 00F0A7                           0056
_outputline                     00F27A - 00F2DA                           027A
_parse_command                  00F96E - 00FA0A                           096E

STATUS:   Build successful; no warnings were issued_____________________...R....
display global_symbols


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

Local

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

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

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

To list the next symbols, press the <PGDN> or <Next> key. the source reference symbols in "spmt_demo.c" will be displayed.

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

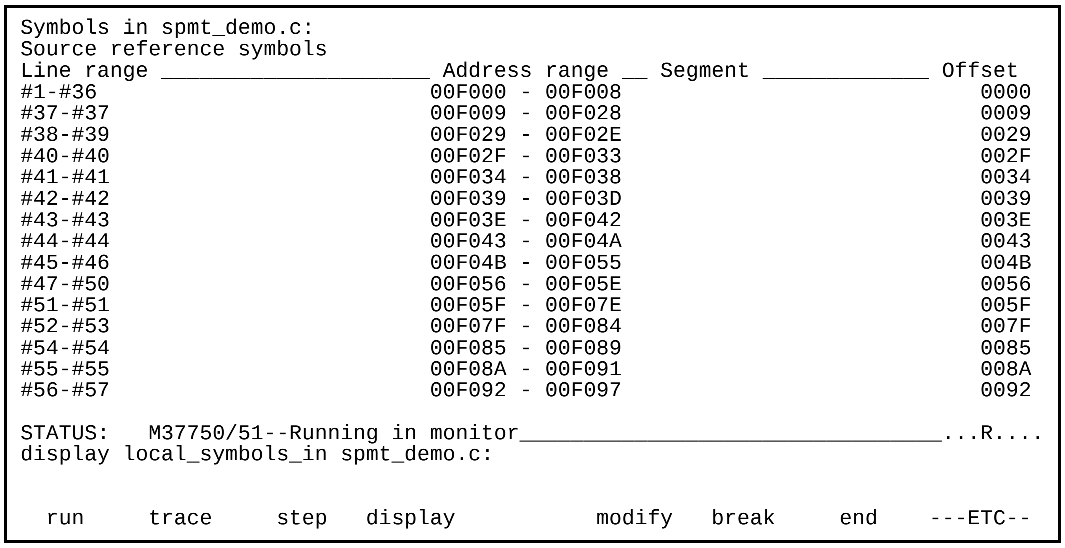
```
Symbols in spmt_demo.c:
Procedure symbols
Procedure name _________________ Address range __ Segment _____________ Offset
_apply_controll                 00F773 - 00F847                           0773
_apply_producti                 00F4FF - 00F5C3                           04FF
_calculate_answ                 00F848 - 00F8F8                           0848
_clear_buffer                   00F2DB - 00F34F                           02DB
_endcommand                     00FA8B - 00FAA2                           0A8B
_format_result                  00F5C4 - 00F638                           05C4
_get_next_token                 00F6D6 - 00F772                           06D6
_initialze                      00F639 - 00F6D5                           0639
_input_line                     00F000 - 00F055                           0000
_lookup_token                   00F350 - 00F3EC                           0350
_main                           00FAA3 - 00FB3A                           0AA3
_math_library                   00F1F0 - 00F279                           01F0
_move_byte                      00F056 - 00F0A7                           0056
_outputline                     00F27A - 00F2DA                           027A
_parse_command                  00F96E - 00FA0A                           096E

STATUS:   cws: spmt_demo.c:_____________________________________________...R....
display local_symbols_in spmt_demo.c:


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

Source Lines

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

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

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



```
Symbols in spmt_demo.c:
Source reference symbols
Line range _____________________ Address range __ Segment _____________ Offset
#1-#36                          00F000 - 00F008                              0000
#37-#37                         00F009 - 00F028                              0009
#38-#39                         00F029 - 00F02E                              0029
#40-#40                         00F02F - 00F033                              002F
#41-#41                         00F034 - 00F038                              0034
#42-#42                         00F039 - 00F03D                              0039
#43-#43                         00F03E - 00F042                              003E
#44-#44                         00F043 - 00F04A                              0043
#45-#46                         00F04B - 00F055                              004B
#47-#50                         00F056 - 00F05E                              0056
#51-#51                         00F05F - 00F07E                              005F
#52-#53                         00F07F - 00F084                              007F
#54-#54                         00F085 - 00F089                              0085
#55-#55                         00F08A - 00F091                              008A
#56-#57                         00F092 - 00F097                              0092

STATUS:   M37750/51--Running in monitor_________________________________...R....
display local_symbols_in spmt_demo.c:


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

Displaying Memory in Mnemonic Format

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

display memory main ***mnemonic options m0x0*** <RETURN>

```
Memory  :mnemonic :file = spmt_demo.c:
   address   data
    00FAA3  0B          PHD
    00FAA4  3B          TSA
    00FAA5  38          SEC
    00FAA6  E90200      SBC A,#0002H
    00FAA9  1B          TAS
    00FAAA  3A          INC A
    00FAAB  5B          TAD
    00FAAC  64000100    LDM #0001H,DP:00H
    00FAB0  9C94050000  LDM #0000H,DT:0594H
    00FAB5  D8          CLM
    00FAB6  A500        LDA A,DP:00H
    00FAB8  C90100      CMP A,#0001H
    00FABB  F003        BEQ 00FAC0H
    00FABD  4C30FB      JMP PG:FB30H
    00FAC0  20F9F8      JSR PG:F8F9H
    00FAC3  206EF9      JSR PG:F96EH

STATUS:   M37750/51--Running in monitor_________________________________...R....
display memory main mnemonic options  m0x0


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

You need to specify the values of M flag and X flag at the staring address of mnemonic memory display. When 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 continue disassembling memory contents.

Note

When you use <PGUP> or <PREV> key to see the previous lines of memory display, disassembled mnemonic may not be accurate.

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

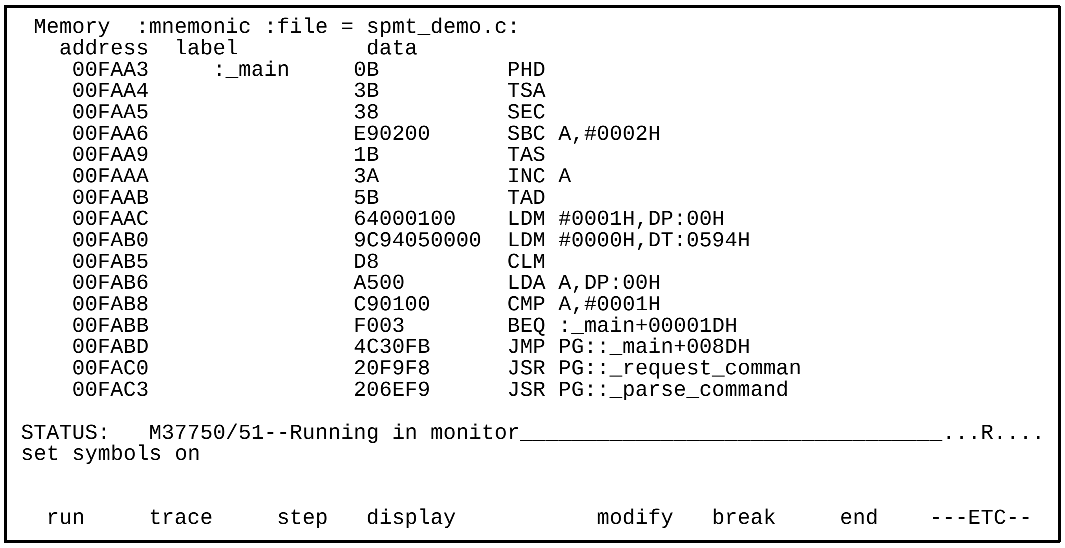
Display Memory with Symbols

If you want to see symbol information with displaying memory in mnemonic format, the emulator Softkey Interface provides "set symbols" command. To see symbol information, enter the following command.

set symbols on <RETURN>

Note

The "set" command is effective only to the window in which the command is invoked. You need to use this command at each window.

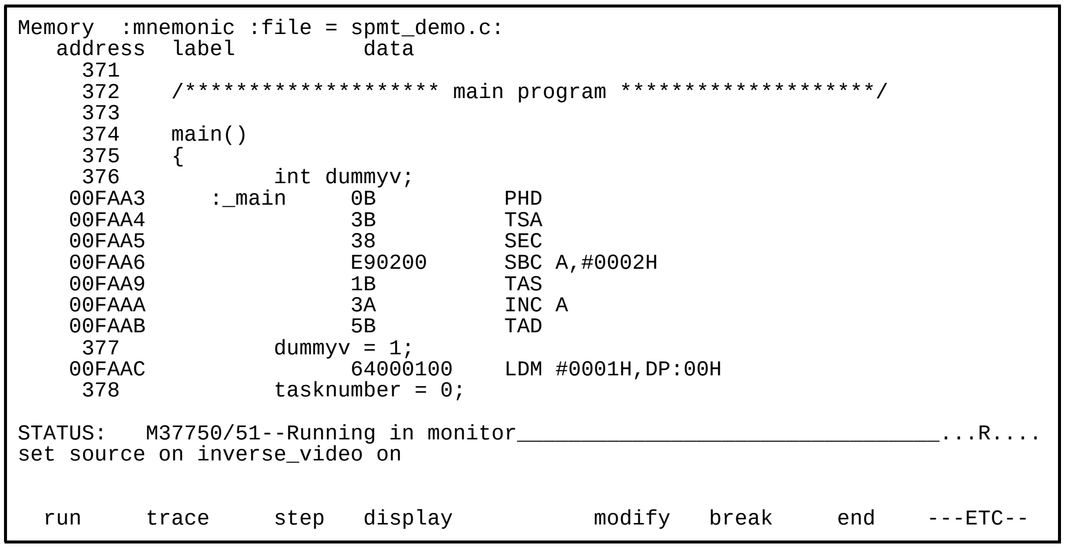


```
 Memory  :mnemonic :file = spmt_demo.c:
   address  label          data
    00FAA3     :_main     0B          PHD
    00FAA4                3B          TSA
    00FAA5                38          SEC
    00FAA6                E90200      SBC A,#0002H
    00FAA9                1B          TAS
    00FAAA                3A          INC A
    00FAAB                5B          TAD
    00FAAC                64000100    LDM #0001H,DP:00H
    00FAB0                9C94050000  LDM #0000H,DT:0594H
    00FAB5                D8          CLM
    00FAB6                A500        LDA A,DP:00H
    00FAB8                C90100      CMP A,#0001H
    00FABB                F003        BEQ :_main+00001DH
    00FABD                4C30FB      JMP PG::_main+008DH
    00FAC0                20F9F8      JSR PG::_request_comman
    00FAC3                206EF9      JSR PG::_parse_command

STATUS:   M37750/51--Running in monitor_________________________________...R....
set symbols on


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

Display Memory with Source Code

If you want to reference the source line information with displaying memory in mnemonic format, the emulator Softkey Interface provides "set source" command. To reference the source line information in inverse video, enter the following command:

set source on inverse_video on <RETURN>



```
Memory  :mnemonic :file = spmt_demo.c:
   address  label          data
     371
     372    /******************** main program ********************/
     373
     374    main()
     375    {
     376            int dummyv;
    00FAA3     :_main     0B          PHD
    00FAA4                3B          TSA
    00FAA5                38          SEC
    00FAA6                E90200      SBC A,#0002H
    00FAA9                1B          TAS
    00FAAA                3A          INC A
    00FAAB                5B          TAD
     377            dummyv = 1;
    00FAAC                64000100    LDM #0001H,DP:00H
     378            tasknumber = 0;

STATUS:   M37750/51--Running in monitor_________________________________...R....
set source on inverse_video on


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

To see the memory without source line referencing, enter the following command:

set source off <RETURN>

Running the Program

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

From Transfer Address

The "run from transfer_address" command specifies that the emulator start executing at a previously defined "start address".

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

Note

The **run from transfer_address** command is not available when you use MELPS 7700 Hex format.

From Reset

The "run from reset" command specifies that the emulator begin executing from reset vector as actual microprocessor does.

(See "Running the Emulation from Target Reset" section in the "In-Circuit Emulation" chapter).

Displaying Memory

The demo program "spmt_demo.c" alters memory.

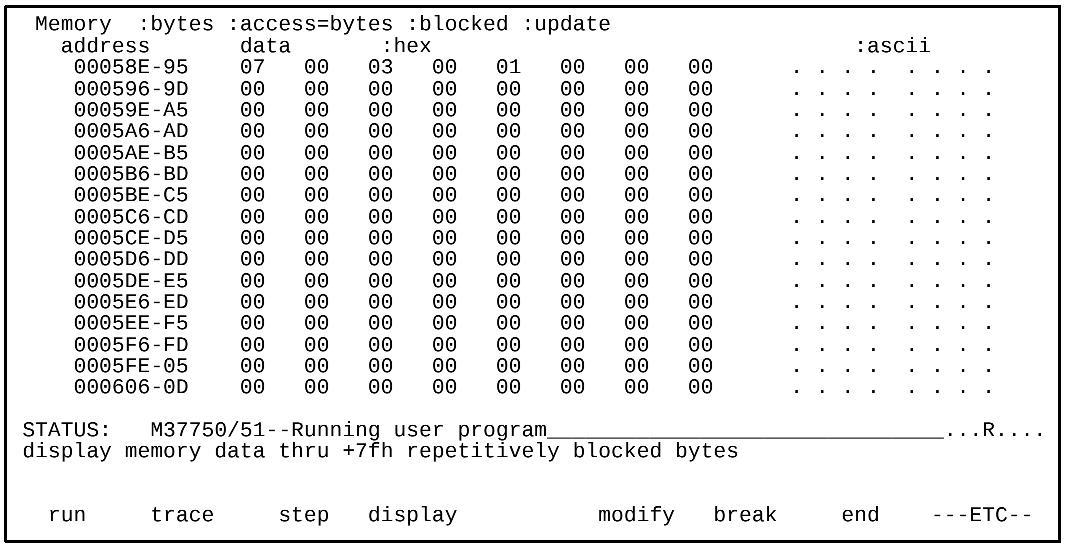
Using Symbolic Addresses

In the following display, the memory range is displayed using symbolic addresses **data**.

The memory display window is periodically updated. For example, enter the following command:

```
display memory data thru +7fh repetitively
blocked bytes <RETURN>
```

This command string is used to specify the range of memory from **data** to **data+7fh**.



```
 Memory  :bytes :access=bytes :blocked :update
   address       data       :hex                                 :ascii
    00058E-95    07   00   03   00   01   00   00   00      . . . .  . . . .
    000596-9D    00   00   00   00   00   00   00   00      . . . .  . . . .
    00059E-A5    00   00   00   00   00   00   00   00      . . . .  . . . .
    0005A6-AD    00   00   00   00   00   00   00   00      . . . .  . . . .
    0005AE-B5    00   00   00   00   00   00   00   00      . . . .  . . . .
    0005B6-BD    00   00   00   00   00   00   00   00      . . . .  . . . .
    0005BE-C5    00   00   00   00   00   00   00   00      . . . .  . . . .
    0005C6-CD    00   00   00   00   00   00   00   00      . . . .  . . . .
    0005CE-D5    00   00   00   00   00   00   00   00      . . . .  . . . .
    0005D6-DD    00   00   00   00   00   00   00   00      . . . .  . . . .
    0005DE-E5    00   00   00   00   00   00   00   00      . . . .  . . . .
    0005E6-ED    00   00   00   00   00   00   00   00      . . . .  . . . .
    0005EE-F5    00   00   00   00   00   00   00   00      . . . .  . . . .
    0005F6-FD    00   00   00   00   00   00   00   00      . . . .  . . . .
    0005FE-05    00   00   00   00   00   00   00   00      . . . .  . . . .
    000606-0D    00   00   00   00   00   00   00   00      . . . .  . . . .

STATUS:   M37750/51--Running user program_______________________________...R....
display memory data thru +7fh repetitively blocked bytes


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

Modifying Memory

You can use the modify memory command to send commands to the sample program. Memory locations **stackarea** and **stackarea+10h** correspond to memory address 402 hex and 412 hex respectivity. For example, to enter the ’10h’ at address 402h and enter ’A’ at address 412h: use the following commands.

display memory `stackarea <RETURN>`
modify memory `stackarea` ***to*** `10h <RETURN>`
modify memory `stackarea+10h` ***string to*** `’A’ <RETURN>`

After the memory location are modified, the memory display shows the following

```
Memory  :bytes :access=bytes :blocked :update
   address       data       :hex                                 :ascii
    000402-09    10   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
    00040A-11    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
    000412-19    41   FF   FF   FF   FF   FF   FF   FF      A . . .  . . . .
    00041A-21    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
    000422-29    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
    00042A-31    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
    000432-39    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
    00043A-41    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
    000442-49    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
    00044A-51    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
    000452-59    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
    00045A-61    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
    000462-69    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
    00046A-71    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
    000472-79    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .
    00047A-81    FF   FF   FF   FF   FF   FF   FF   FF      . . . .  . . . .

STATUS:   M37750/51--Running user program_______________________________...R....
modify memory stackarea+10h string to ’A’


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

Note

Modifying/displaying internal RAM or SFR suspends user program execution. This is because the emulator usesinternal RAM and SFR of emulation processor to perform emulation. However, you can configure the emulator so that write cycles are performed to both internal RAM (or SFR) and emulation memory. If you do this, you can display the data written to emulation memory without suspending user program execution. Refer to chapter 4 of this manual for more details.

Breaking into the Monitor

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

```
    break <RETURN>
```

Notice that the current address is pointed out with inverse video in displaying memory when the execution breaks to the monitor.

Using Software Breakpoints

Software breakpoints are handled by BRK instruction. When you define or enable a software breakpoint, the emulator will replace the opcode at the software breakpoint address with a BRK instruction.

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 instructions, you cannot define software breakpoints in target ROM.

Note

When you mapped interanl RAM as emulation memory, it is not allowed to set software breakpoints to this area.

Caution

Software breakpoints should not be set, enabled, disabled, or removed 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.

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

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

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

Enabling/Disabling Software Breakpoints

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

```
modify software_breakpoints enable <RETURN>
```

When software breakpoints are enabled and you set a software breakpoint, the BRK instruction will be placed at the address specified. When the breakpoint interrupt instruction is executed, program execution will break into the monitor.

Setting a Software Breakpoint

To set a software breakpoint at line 80 of "spmt_demo.c", enter the following command.

```
modify software_breakpoints set line 80
<RETURN>
```

To see the address where the software breakpoint has been set, enter the following command:

```
display memory line 80 mnemonic <RETURN>
set source on inverse_video on <RETURN>
```

```
 Memory  :mnemonic :file = spmt_demo.c:
   address   data
      80                     data = 1;
*   00F129  008E        BRK
    00F12B  0501        ORA A,DP:01H
    00F12D  009C        BRK
      81                     stack = 0;
    00F12E  9C90050000  LDM #0000H,DT:0590H
    00F133  4C1DF1      JMP PG:F11DH
      82             }
      83             pre_fetch = 0;
    00F136  D8          CLM
    00F137  9C92050000  LDM #0000H,DT:0592H
      84             pre_fetch = 1;
    00F13C  9C92050100  LDM #0001H,DT:0592H
      85    }
    00F141  D8          CLM
    00F142  AA          TAX

STATUS:   M37750/51--Running in monitor_________________________________...R....
set source on inverse_video on


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

The asterisk (*) in left side of the address lists points out that the software breakpoint has been set. The opcode at the software breakpoint address was replaced to the software breakpoint instruction.

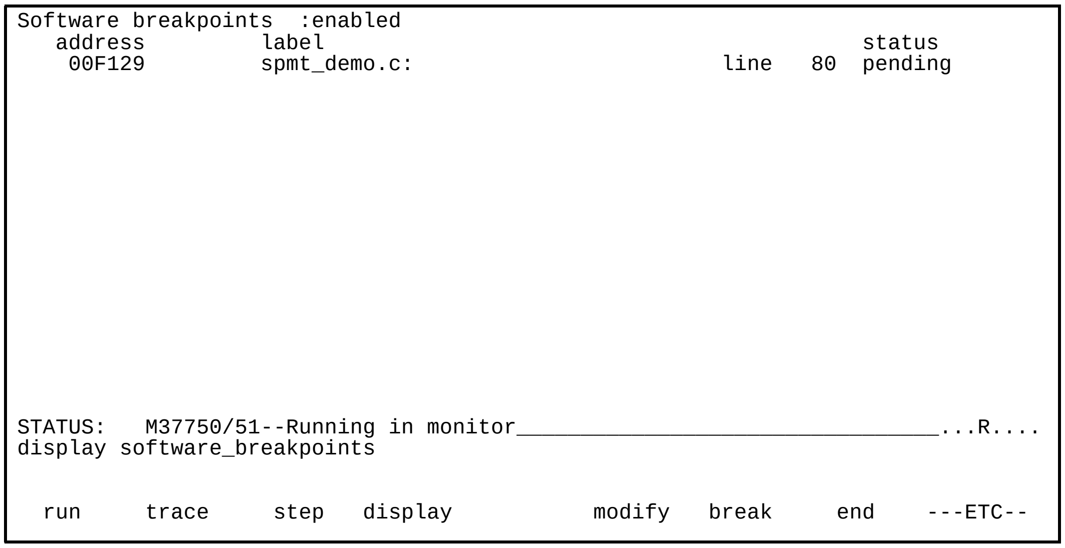
Note

When a software breakpoint is inserted, the mnemonic in memory display may not be accurate.

Displaying Software Breakpoints

To display software breakpoints, enter the following command.

display software_breakpoints <RETURN>



```
Software breakpoints  :enabled
   address         label                                                 status
    00F129         spmt_demo.c:                        line   80  pending











STATUS:   M37750/51--Running in monitor_________________________________...R....
display software_breakpoints


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

The software breakpoints display shows that the breakpoint is pending. When breakpoints are hit they become inactivated. To reactivate the breakpoint so that is "pending", you must re-enter the "modify software_breakpoints set" command.

After the software breakpoint has been set, enter the following command to cause the emulator to continue executing the demo program.

run <RETURN>

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

The software breakpoint address is pointed out with inverse video in displaying memory in mnemonic format. To see the software breakpoint with memory, enter the following command.

display memory line 80 ***mnemonic m0x0*** <RETURN>

Notice that the original opcode was replaced at the address that the software breakpoint has been set.

Clearing a Software Breakpoint

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

modify software_breakpoints clear `line 80 <RETURN>`

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

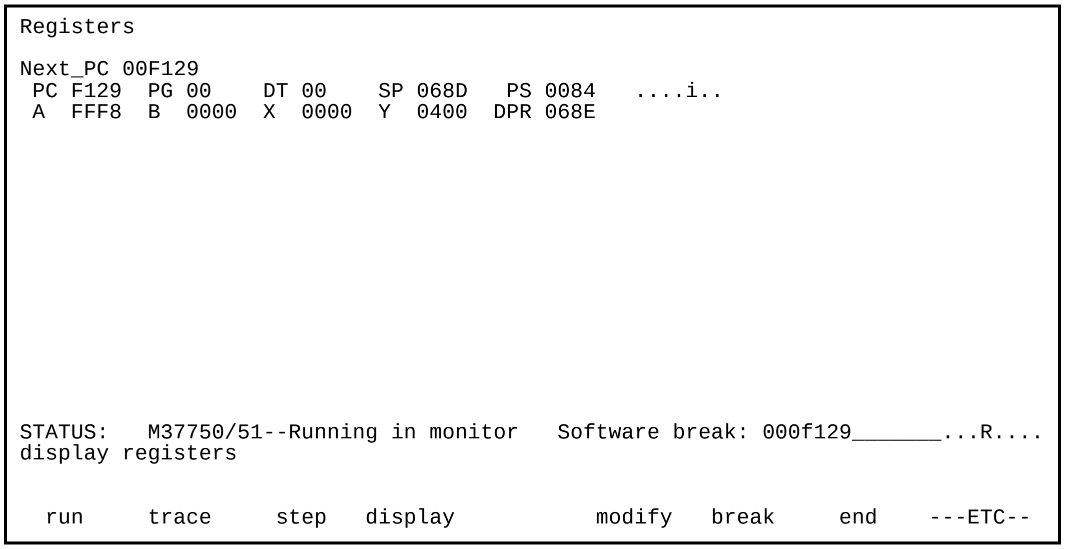
To clear all software breakpoints, you can enter the following command.

modify software_breakpoints clear `<RETURN>`

Displaying Registers

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

display registers `<RETURN>`



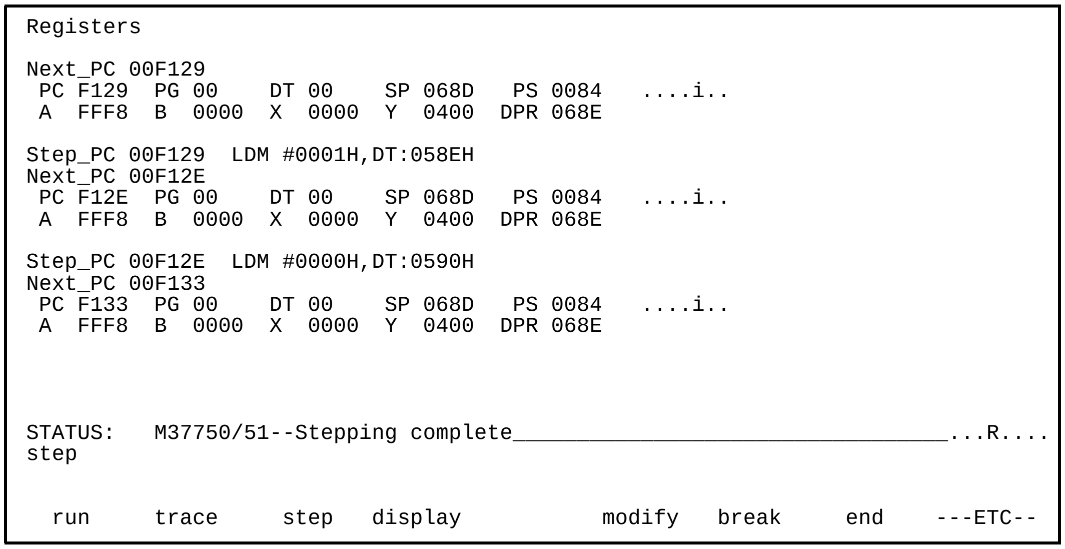
```
Registers

Next_PC 00F129
 PC F129  PG 00    DT 00    SP 068D   PS 0084   ....i..
 A  FFF8  B  0000  X  0000  Y  0400  DPR 068E
















STATUS:   M37750/51--Running in monitor   Software break: 000f129_______...R....
display registers


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

Stepping Through the Program

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

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

You will see the inverse-video moves according to the step execution. You can continue to step through the program just by pressing the <RETURN> key.



```
Registers

Next_PC 00F129
 PC F129  PG 00    DT 00    SP 068D   PS 0084   ....i..
 A  FFF8  B  0000  X  0000  Y  0400  DPR 068E

Step_PC 00F129  LDM #0001H,DT:058EH
Next_PC 00F12E
 PC F12E  PG 00    DT 00    SP 068D   PS 0084   ....i..
 A  FFF8  B  0000  X  0000  Y  0400  DPR 068E

Step_PC 00F12E  LDM #0000H,DT:0590H
Next_PC 00F133
 PC F133  PG 00    DT 00    SP 068D   PS 0084   ....i..
 A  FFF8  B  0000  X  0000  Y  0400  DPR 068E




STATUS:   M37750/51--Stepping complete__________________________________...R....
step


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

You can step program execution by source lines, enter:

step source <RETURN>

Source line stepping is implemented by single stepping assembly instructions until the next PC is outside of the address range of the current source line. When source line stepping is attempted on assembly code, stepping will complete when a source line is found. To terminate stepping type <Ctrl>-C.

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

Note Step command to internal RAM area is not allowed.

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

Using the Analyzer

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

Source Line Referencing

A trace may be taken and displayed using source line referencing. Also, lines of the source program can be displayed with the trace list where the trace occurred.

To display the trace with source code in inverse video, enter the following command:

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

Specifying a Simple Trigger

Suppose you want you trace program execution after the point at address **semantic_check**. The following command make this trace specification.

trace after semantic_check ***status exec***<RETURN>

The STATUS message shows "Emulation trace started.".

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

run <RETURN>

The STATUS message shows "Emulation trace complete.".

Display the Trace

The trace listings which following are of program execution on the 7750/51 emulator. To see the trace list, enter the following command:

display trace <RETURN>

```
Trace List    Depth=512     Offset=0
Label:      Address       Data    Opcode or Status w/ Source Lines   time count
Base:       symbols       hex            mnemonic w/symbols           relative
      ##########spmt_demo.c - line   201 thru   204 ############################

      semantic_check()
      {
              int i;
after  :_semantic_check     3B0B  PHD                                 120     nS
+001   :_semanti+000002     E938       E938H  opcode fetch            240     nS
+002   :_semanti+000001     0694  TSA                                 240     nS
+003   :scrt0.a:+0000EA     0694       0694H  data write              120     nS
+004   :_semanti+000002     0694  SEC                                 120     nS
+005   :_semanti+000003     0002  SBC A,#0002H                        280     nS
+006   :_semanti+000004     0002       0002H  opcode fetch            120     nS
+007   :_semanti+000006     3A1B       3A1BH  opcode fetch            360     nS
+008   :_semanti+000006     3A1B  TAS                                 120     nS
+009   :_semanti+000007     645B  INC A                               120     nS

STATUS:   M37750/51--Running user program Emulation trace complete______...R....
run


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

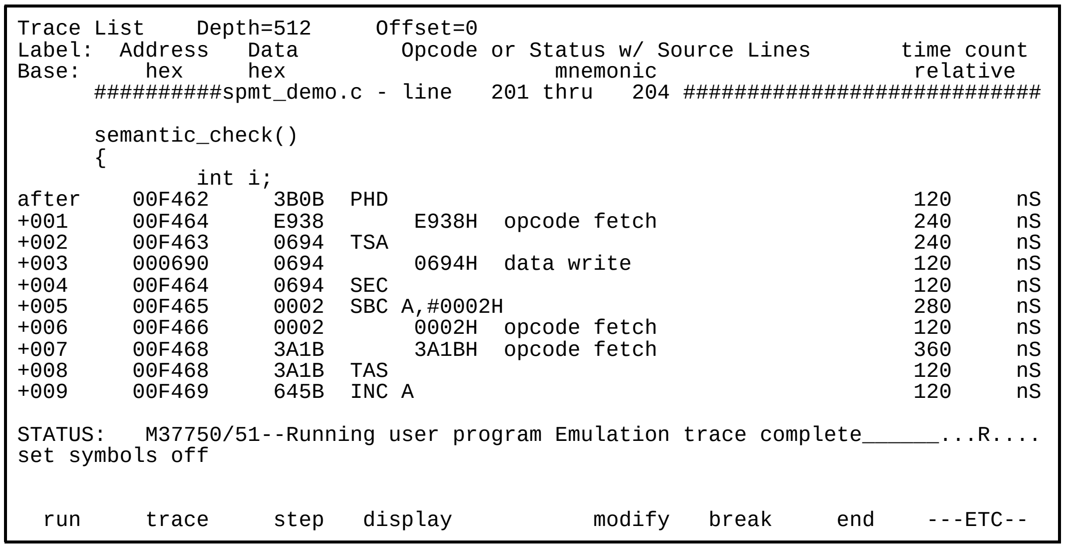
The trace list shows the trace after line (semantic_check()).

To list the next lines of the trace, press the <PGDN> or <NEXT> key.

Displaying Trace with No Symbol

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

set symbols off <RETURN>



```
Trace List    Depth=512     Offset=0
Label:  Address   Data        Opcode or Status w/ Source Lines       time count
Base:     hex     hex                     mnemonic                    relative
      ##########spmt_demo.c - line   201 thru   204 ############################

      semantic_check()
      {
              int i;
after    00F462     3B0B  PHD                                            120     nS
+001     00F464     E938       E938H  opcode fetch                       240     nS
+002     00F463     0694  TSA                                            240     nS
+003     000690     0694       0694H  data write                         120     nS
+004     00F464     0694  SEC                                            120     nS
+005     00F465     0002  SBC A,#0002H                                   280     nS
+006     00F466     0002       0002H  opcode fetch                       120     nS
+007     00F468     3A1B       3A1BH  opcode fetch                       360     nS
+008     00F468     3A1B  TAS                                            120     nS
+009     00F469     645B  INC A                                          120     nS

STATUS:   M37750/51--Running user program Emulation trace complete______...R....
set symbols off


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

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

Displaying Trace with Compress Mode

If you want to see more executed instructions on a display, the 7750/51 emulator Softkey Interface provides **compress mode** for analysis display. To see trace display with compress mode, enter the following command:

display trace compress on <RETURN>

```
Trace List    Depth=512     Offset=0
Label:  Address   Data        Opcode or Status w/ Source Lines       time count
Base:     hex     hex                     mnemonic                     relative
      ##########spmt_demo.c - line   201 thru   204 ############################

      semantic_check()
      {
              int i;
after    00F462     3B0B  PHD                                           120     nS
+002     00F463     0694  TSA                                           480     nS
+003     000690     0694       0694H  data write                        120     nS
+004     00F464     0694  SEC                                           120     nS
+005     00F465     0002  SBC A,#0002H                                  280     nS
+008     00F468     3A1B  TAS                                           600     nS
+009     00F469     645B  INC A                                         120     nS
+011     00F46A     645B  TAD                                           280     nS
      ##########spmt_demo.c - line   205 #######################################
              for (i = 0; i  4; i++)

STATUS:   M37750/51--Running user program Emulation trace complete______...R....
display trace compress on


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

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

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

display trace compress off <RETURN>

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

Displaying Trace with Time Count Absolute

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

display trace count absolute <RETURN>

Emulator Analysis Status Qualifiers

The following analysis status qualifiers may also be used with the 7750/51 emulator.

Qualifier	Status bits	Description
bg	0x1xxxxxxy	Background cycle
byte	0xx1x1x1xy	Byte memory cycle
cpu	0xx11xxxxy	CPU cycle
data	0xx1x10xxy	Data cycle
dma	0xx10xxxxy	DMA cycle
exec	0xx1101xxy	Execution cycle
fetch	0xx1111x1y	Fetch cycle
fg	0x0xxxxxxy	Foreground cycle
hold	0xx01xxxxy	HOLD cycle
mx	1xxxxxxxy	Value of MX signal
read	0xx1x1xx1y	Read cycle
ref	0xx00xxxxy	Refresh cycle
word	0xx1x1x0xy	Word access
write	0xx1x1xx0y	Write to ROM cycle

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

Resetting the Emulator

To reset the emulator, enter the following command.

```
reset <RETURN>
```

Exiting the Softkey Interface

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

End Release System

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

```
end release_system <RETURN>
```

Ending to Continue Later

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

```
end <RETURN>
```

Ending Locked from All Windows

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

```
end locked <RETURN>
```

This option only appears when you enter the Softkey Interface via the **emul700** command.

3

"In-Circuit" Emulation

Introduction

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

This chapter will:

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

Prerequisites

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

Installing the Target System Probe

Caution

POSSIBLE DAMAGE TO THE EMULATOR PROBE. The emulation probe contains devices that are susceptible to damage by static discharge. Therefore, precautionary measures should be taken before handling the microprocessor connector attached to the end of the probe cable to avoid damaging the internal components of the probe by static electricity.

Caution

POSSIBLE DAMAGE TO THE EMULATOR. Make sure target system power is OFF before installing the emulator probe into the target system. Do not install the emulator probe into the processor socket with power applied to the target system.

Caution

DAMAGE TO THE EMULATOR WILL RESULT IF THE PROBE IS NOT INSTALLED CORRECTLY. Make sure pin 1 of probe connector is aligned with pin 1 of the socket. When installing the emulation probe, be sure that the probe is installed into the processor socket so that pin 1 of the connector aligns with pin 1 of the socket.

Note

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

Installing the Target System Probe

1. Set up the switches inside the emulation pod. When you are using the HP 64146-61002 emulation pod, refer to the *7750/51 Series Emulation Pod User's Guide*. When you are using an other emulation pod, refer to the manual provided with your emulation pod.

2. Remove the 7750/51 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. (See figure 3-1.)

5. Turn on power of your target system, and then, turn on the emulator.

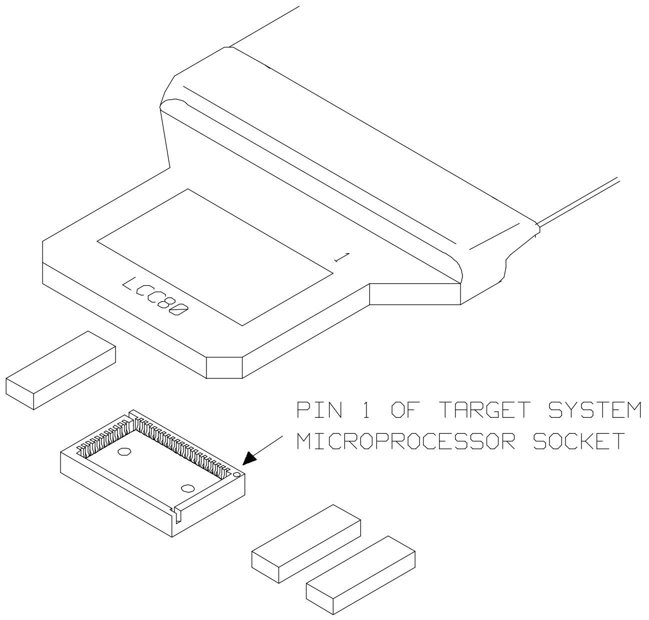


Figure 3-1. Installing the Probe to LCC80 Socket

When your target system uses 64 pin shrink DIP socket, use the adapter as shown in figure 3-2.

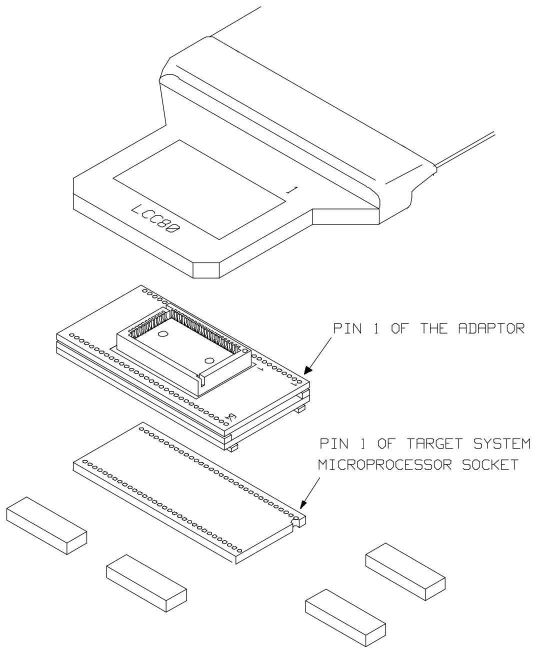


Figure 3-2. Installing the Probe to SDIP64 Socket

In-Circuit Configuration Options

The 7750/51 Series emulator provides configuration options for the following in-circuit emulation issues.
Refer to the "Configuring the Emulator" for more information on these configuration options.

Using the Target System Clock Source

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

Note

Your target system **must** have a clock generation circuit. The emulation pod cannot generate clock signal using a ceramic (or quartz crystal) resonator.

Target Memory Access Size

You can configure the emulator to access target system memory by byte access or word access to perform emulation commands.

Respond to Target System Interrupts

You can configure the emulator whether or not the emulator responds to interrupt signals from the target system during foreground operation.

Note

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

Running the Emulation from Target Reset

You can specify that the 7750/51 emulator begins execution from target system reset. When the target system RESET line becomes active and then inactive, the 7750/51 emulator will start reset sequence as actual microprocessor.

To specify a run from target system reset, enter the following command:

```
run from reset
```

The status now shows that the 7750/51 emulator is "Awaiting target reset". After the target system is reset, the status line message will change to show the appropriate emulator status.

Note

In the "Awaiting target reset" status, you can not break into the monitor. If you want to exit this status, you need to enter **reset** command.

Configuring the Emulator

Introduction

Your 7750/51 Series emulator can be used in all stages of target system development. For instance, you can run the emulator out-of-circuit when developing your target system software, or you can use the emulator in-circuit when integrating software with target system hardware. You can use the emulator's internal clock or the target system clock. Emulation memory can be used in place of, or along with, target system memory. You can execute target programs in real-time or allow emulator execution to be diverted into the monitor when commands request access of target system resources (target system memory, register contents, etc).

The emulator is a flexible instrument and may be configured to suit your needs at any stage of the development process. This chapter describes the options available when configuring the HP 64147A emulator.

The configuration options are accessed with the following command.

```
modify configuration <RETURN>
```

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

General Emulator Configuration:

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

Emulator Reconfiguration:

- Selecting microprocessor to be emulated.

- Selecting CPU operation mode.
- Defining the reset value of the stack pointer.

Memory Configuration:

- Enabling the high speed access mode
- Selecting the background or foreground emulation monitor.
- Mapping memory.

Emulator Pod Configuration:

- Selecting target memory access data size.
- Enabling interrupts from the target system.
- Enabling watch dog timer.

Debug/Trace Configuration:

- Enabling breaks on writes to ROM.
- Specifying trace cycle. (foreground/background/both)
- Enabling tracing refresh cycles.
- Enabling tracing DMA cycles.
- Enabling tracing HOLD/HLDA cycles.
- Enabling 16bit symbol display.
- Defining the DT register value for symbol display.

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

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

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

General Emulator Configuration

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

Micro-processor clock source?

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

internal	Selects the internal clock oscillator as the emulator clock source. The internal clock is provided from the emulation pod. In the case of HP 64146-61602 emulation pod, the clock speed is 1/8/16/25 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, one wait state is inserted by the emulator.
external	Selects the clock input to the emulator probe from the target system. You must use a clock input conforming to the specifications for the 7750/51 Series microprocessor. The HP 64147A 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, one wait state is inserted by the emulator.

Note

Maximum clock speed of HP 64147A 7750/51 emulator is 25MHz. This emulator does not support any operation with clock faster than 25MHz.

Note

Your target system **must** have a clock generation circuit. The emulation pod cannot generate clock signal using a ceramic (or quartz crystal) resonator.

Note

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

Enter monitor after configuration?

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

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

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

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

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

Restrict to real-time runs?

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

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

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

Caution

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

Emulator Reconfiguration

The emulator reconfiguration questions allows you to reconfigure the emulator for your system. Type of the processor, processor operation mode and reset value for the stack pointer will be configured here. To access the emulator reconfiguration questions, you must answer "yes" to the following question.

Reconfigure emulator?

Processor type?

This configuration item allows you to select the processor you are going to emulate.

Processors supported by the 7750/51 emulator are listed in the *Processor Support List for HP MELPS Emulators*. This list describes appropriate <chip_name> which is needed for this configuration item. Refer to this list to know appropriate <chip_name>, or to determine if your processor is supported or not.

Note

If you enter incorrect <chip_name> for this item, processor type is not changed and this configuration will be finished. Then, you need to RECALL this configuration item and re-enter correct <chip_name>.

Note

If your processor is not included in the *Processor Support List for HP MELPS Emulators*. You need to select "other" in this item and answer to the following questions.

Usually, the previous question set up internal memory addresses automatically. However, when your processor is not supported by the previous question, you must configure the following questions by yourself.

SFR area start address?
SFR area end address?
Second SFR area start address?
Second SFR area end address?

Specify the start address and end address of internal SFR of your processor. These addresses can be defined on 16 byte boundaries.

If your processor has only one SFR area, specify the same value as the first one for the "Second SFR ..." questions.

Internal RAM area start address?
Internal RAM area end address?
Second internal RAM area start address?
Second internal RAM area end address?

Specify the start address and end address of internal RAM of your processor. These addresses can be specified on 16 byte boundaries. If your processor has no internal RAM, enter 0 as start address and end address of internal RAM area. If your processor has only one SFR area, specify the same value as the first one for the "Second internal RAM ..." questions.

Internal ROM area start address?
Internal ROM area end address?

Specify the start address and end address of internal ROM of your processor. These addresses can be defined on 16 byte boundaries. If your processor has no internal ROM, enter 0 as start address and end address of internal ROM area.

Processor mode register address?

Specify the address of processor mode register. This is needed to manage processor operation modes.

Processor mode?

This configuration defines operation mode of the processor.

single	The emulator will operate in single-chip mode.
expand8	The emulator will operate in memory expansion mode with 8 bit data width.
expand16	The emulator will operate in memory expansion mode with 16 bit data width.
proc8	The emulator will operate in microprocessor mode with 8 bit data width.
proc16	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.

Modify value for Stack Pointer (SP)?

Reset value for the stack pointer is automatically set up to the end of internal RAM area. When the processor you select has no internal RAM, it is set up to FFF hex. If you would like to change the value, answer "yes" to this question.

Reset value for Stack Pointer (SP)?

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 address specified in response to this question must be a 16-bit hexadecimal address.

This address should be defined in RAM area (internal RAM, target RAM or emulation RAM) which is not used by user program. When the emulator breaks to the background monitor, the background monitor uses 5 bytes of stack area.

Caution

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

Memory Configuration

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

Modify memory configuration?

Is speed of input clock faster than 16 MHz?

This question allows you to configure the emulator for clock (internal/external) faster than 16 MHz.

no	When the clock speed is equal or slower than 16 MHz, select this answer. The emulator runs with no wait state.
yes	When the clock speed is faster than 16 MHz, select this answer. You will be asked the following question.

Enable high speed access mode for emulation memory?

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 inserts 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:

- You can define up to 16 different map terms which can be placed wherever you like. (Refer to "Mapping memory" section in this chapter.)
- The emulator generates the /RDY signal, and inserts one wait state for all memory access.

Note

Changing this configuration will reset the memory map,

Monitor type?

The monitor type configuration question allows you to choose between a foreground monitor (which is supplied with the emulation software but must be assembled, linked, converted, and loaded into emulation memory) or the background monitor (which resides in the emulator).

The *emulation monitor* is a program that is executed by the emulation processor. It allows the emulation system controller to access target system resources. For example, when you enter a command that requires access to target system resources, say a command to display target system memory, the system controller writes a command code to the monitor communications area and breaks execution of the emulation processor from the user program into the monitor program. The monitor program then reads the command from the communications area and executes the 7700 Series instructions which read the contents of the target system memory locations. After the monitor has completed its task, execution returns to the user program.

The *background monitor* , resident in the emulator, offers the greatest degree of transparency to your target system (that is, your target system should generally be unaffected by monitor execution). However, in some cases you may require an emulation monitor tailored to the requirements of your system. In this case, you will need to use a foreground monitor linked into your program modules. See the "Using the Foreground Monitor" appendix for more information on foreground monitors.

background	Selects the use of the background monitor. When you select the background monitor and the current monitor type is "foreground", you are asked the following question.

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

This question must be answered "yes" to change the monitor type.

foreground	Specifies that a foreground monitor will be used. Foreground monitor programs are shipped with the Softkey Interface. When you select a foreground monitor, you will be asked additional questions.

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

This question must be answered "yes" or else the foreground monitor will not be selected.

Monitor address?

The default configuration specifies a monitor address of 0b800 hex. The monitor base address must be located on a 2K byte boundary other than internal RAM and SFR area; otherwise, configuration will fail.

Monitor filename?

This question allows you to specify the name of the foreground monitor program absolute file. Remember that the foreground monitor must already be assembled and linked starting at the 2K byte boundary specified for the previous "Monitor address?" question.

The monitor program will be loaded after you have answered all the configuration questions; therefore, you should not link the foreground monitor to the user program. If it is important that the symbol database contain both monitor and user program symbols, you can create a different absolute file in which the monitor and user program are linked. Then, you can load this file after configuration.

Mapping memory

The emulation memory consists of 128K/512K/1M/2M bytes, mappable in 256 byte blocks. You can define up to 16 different map terms.

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 the emulation memory for that address range. You can also specify whether the target system memory is ROM or RAM, and you can specify that emulation memory be treated as ROM or RAM.

Note

When you use background monitor, the emulator occupies 2K byte, which is used for background monitor program, leaving 122K, 506K, 1018K, 2042K byte of emulation memory which you may use.

Note

You cannot map Internal RAM and SFR as guarded.

Note

Target system accesses to emulation memory are not allowed. Target system devices that take control of the bus (for example, external DMA controller) cannot access emulation memory.

Caution

The default emulator configuration maps location C000 hex through FFFF hex as emulation ROM. This must be needed when you use the internal ROM. You don't have to map internal RAM area since the emulator uses internal RAM of the emulation processor.

When you answered "yes" to the **"Reset map (change of monitor type requires map reset)?** question , you must map again for memory space where internal ROM is located as emulation ROM.

Blocks of memory can also be characterized as guarded memory.

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

To map memory for the sample program, enter the following mapper commands:

```
delete all <RETURN>
0c000h thru 0ffffh emulation rom <RETURN>
end <RETURN>
```

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

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

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 display 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 with "display memory" command without suspending user program execution. You still can display the contents of internal RAM by specifying "fcode i" syntax in "display memory" command.

For example, to see the contents of address 100 hex in internal RAM, you can do both of the following:

```
display memory fcode none 100h
(This command accesses emulation memory)
display memory fcode i 100h
(This command accesses internal RAM of
emulation processor.)
```

Note

When you specify "fcode", the "fcode" becomes the new default to display memory. That is, once you specify "fcode i", you need to specify "fcode none" to display emulation memory.

When you don’t map the internal RAM and SFR area to emulation RAM (when you don’t copy the contents to emulation memory), you can access the internal RAM and SFR without specifying "fcode" syntax.

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. 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.

Emulator Pod Configuration

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

Modify emulator pod configuration?

Target memory access size?

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.

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

Respond to target system interrupts?

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

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.

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.

Enable watchdog timer?

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

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

Debug/Trace Configuration

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

Modify debug/trace options?

Break processor on write to ROM?

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

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

Trace background or foreground operation?

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

foreground	Specifies that the analyzer trace only foreground cycles. This option is specified by the default emulator configuration.
background	Specifies that the analyzer trace only background cycles. (This is rarely a useful setting.)
both	Specifies that the analyzer trace both foreground and background cycles. You may wish to specify this option so that all emulation processor cycles may be viewed in the trace display.

Trace refresh cycles by emulation analyzer?

This question is asked only when the 7720 processor is selected in the configuration.

You can direct the emulator to send refresh cycle data to emulation analyzer or not to send it.

yes	Enables the emulator to trace refresh cycles.
no	Refresh cycles will not appear on analysis trace list.

Trace DMA cycles by emulation analyzer?

This question is asked only when the 7720 processor is selected in in the configuration.

You can direct the emulator to send DMA cycle data to emulation analyzer or not to send it.

yes	When you enable tracing DMA cycles, DMA cycles will appear as one analysis trace line.
no	DMA cycles will not appear on analysis trace list.

Trace HOLD/HLDA cycles by emulation analyzer?

You can direct the emulator to send HOLD/HLDA cycle data to emulation analyzer or not to send it.

yes	When you enable tracing HOLD/HLDA cycles, these cycles will appear as one analysis trace line.
no	HOLD/HLDA cycles will not appear on analysis trace list.

Replace 16-bit addresses with symbolic references?

You can direct the emulator whether or not to display symbols in 16bit addresses in mnemonic field of memory and trace display.

no	Symbols are displayed only in 24bit addresses of mnemonic field.
yes	Symbols are displayed both in 16 and 24bit addresses of mnemonic field. When you select this answer, you are asked the following question.

Data bank register value for symbolic references?

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

Simulated I/O Configuration

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

Interactive Measurement Configuration

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

External Analyzer Configuration

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

Saving a Configuration

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

Configuration file name? <FILE>

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

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

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

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

Loading a Configuration

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

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

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

```
load configuration <RETURN>
```

Limitations and Considerations

Clock Speed

Maximum clock speed of HP 64147A 7750/51 emulator is 25MHz. This emulator does not support any operation with clock faster than 25MHz.

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 M37780/81/82/83/85/95/96 emulation pod.
- The processor is operating in the memory expansion or microprocessor mode with 8 bit external bus.

Step Command to Internal RAM

Step command to internal RAM area is not available.

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 microprocessor is in the stop or wait mode, emulation commands which access memory or registers will fail. In the case of using M37782/83/85 emulation pod, you need to reset the emulator to release stop or wait mode. And, in the case of using other emulation pod, you need to break the the emulator.

RDY/HOLD Input in Background Cycles

The 64147A M37750/51 emulator does not accept RDY/HOLD input while in background monitor. However, when you use M37780/81/82/83/85/95/96 emulation pod, M37750/51 emulator accept RDY/HOLD input while in background monitor.

Accessing External Memory Area in SFR

When operation mode is memory expansion or microprocessor mode, there is external memory area in SFR. However, accessing to this area is not allowed.

High Speed Bus Mode

Always set bus mode as low speed bus mode, when you use M37751 emulation pod. HP 64147A 7750/51 emulator does not support high speed bus mode. Note that bus mode is automatically configured as high speed bus mode when you do **run from reset** command. Then, you need to re-configure bus mode as low speed bus mode before accessing SFR area.

RMPA Instruction

Disassembling in trace list may not be correct for next instruction of RMPA instruction. This failure will occur when RMPA instruction is repeated over about fifty times.

Stack Address

In some versions of 7720 microprocessor, the stack can be located in Bank FF. However, the HP 64147A 7750/51 Series emulator doesn’t support the feature. The stack must be located in Bank 0.

Evaluation Chip

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

Notes

A

Using the Foreground Monitor

Introduction

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

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

/usr/hp64000/monitor/*

The monitor programs named **fm7750.a77** is written for Mitsubishi RASM77 Assembler, and **fm7750.src** is written for MRI ASMM77 Assembler.

In this capter, **fm7750.a77** is used for the sample.

Comparison of Foreground and Background Monitors

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

Background Monitors

A *background* monitor is an emulation monitor which overlays the processor's memory space with a separate memory region. Entry into the monitor is normally accomplished by jamming the monitor addresses onto the processor's address bus.

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

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

Foreground Monitors

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

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

An Example Using the Foreground Monitor

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

For this example, we will locate the monitor at b800 hex; the sample program will be located at f000 hex with the data table at 400 hex.

```
$ cp /usr/hp64000/monitor/fm7750.a77 .
<RETURN>
```

Assemble and Link the Monitor

You can assemble, link and convert the foreground monitor program with the following commands.

```
$ rasm77 -s fm7750.a77 <RETURN>
$ link77 fm7750 <RETURN>
```

Enter **-s** as command parameter.

```
$ m77cnvhp fm7750 <RETURN>
```

If you haven’t already compiled the sample program, do that now. Refer to the "Getting Started" chapter for instructions on assembling, linking, and converting the sample program.

Modifying Location Declaration Statement

You may need to modify the foreground monitor program to adjust it to your needs.

Monitor Address

You can load the monitor "fm7750.a77" at any base address on a 2K byte boundary except internal RAM and SFR area. To relocate the monitor, you must modify the "LOCATE_ADRS" label statement near the top of the monitor listing to point the base address where the monitor will be loaded. You will see the statement in the monitor listing as follows:

```
LOCATE_ADRS .EQU 0B000H ;start monitor on 2k boundary in bank 0
                        ;rather than sfr/iram area
PROCMODEREG .EQU 0005EH ;processor mode register’s address
```

For example, if you want to locate the monitor at a000 hex, you may change the address "0B800H" to "0A000H".

Processor Mode Register Address

You may need to modify the .EQU statement at the PROCMODEREG label. This value defines the location of processor mode register. If your processor has processor mode register at address other than 5e hex, modify this value to appropriate value.

Processor Name

To use the foreground monitor with 7751 microprocessor, you need to modify the processor name section of foreground monitor. Default setting is following.

```
CHIP7751        .WORD   0  ; OTHER_THAN_7751
```

You can specify 7751 microprocessor by modifying this section like below.

```
CHIP7751        .WORD   1  ; 7751
```

Modifying the Emulator Configuration

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

Modify memory configuration? yes

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

Reconfigure emulator? yes

You need to answer yes for this question to select your processor.

Processor type? <Processor_Name>

Select the processor you are going to emulate.

Monitor type? foreground

Specifies that you will be using a foreground monitor program.

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

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

Monitor address? 0b800h

Specifies that the monitor will reside in the 2K byte block from b800 hex through bfff hex.

Monitor file name? fm7750

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

Mapping Memory for the Example

When you specify a foreground monitor and enter the monitor address, all existing memory mapper terms are deleted and a term for the monitor block will be added. Add the additional term to map memory for the sample program and, map other area as target RAM.

```
400h thru 5ffh emulation ram <RETURN>
0c000h thru 0ffffh emulation ram <RETURN>
default target ram <RETURN>
end <RETURN>
```

If your processor has no internal RAM, map 0 hex through 2ff hex as emulation RAM.

See the "Mapping Memory" section of the "Configuring the Emulator" chapter for more information.

Configuration file name? fmconfig

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

Load the Program Code

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

```
load spmt_demo <RETURN>
```

Running User Program

Before running the user program, you should initialize the stack pointer by breaking the emulator out of reset.

```
break <RETURN>
```

To run the sample program from address **Init**, enter the following command:

```
run from scrt0.a77:start <RETURN>
```

Now you can use the emulator with the foreground monitor.

Limitations of Foreground Monitors

Step Command

Step command is not available when you are using the foreground monitor.

Synchronized measurements

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

Notes

Using the Format Converter

How to Use the Converter

The format converter is a program that generates HP format files from MELPS 7700 Hex format file and its symbol file. This means you can use available language tools to create MELPS 7700 Hex format file, then load the file into the emulator using the format converter.

Note

If you make your program as IEEE-695 format, you don't need to convert the program to HP Absolute file. The HP 64147A Softkey Interface can load IEEE-695 format without any conversion.

To execute the converter program, use the following command:

```
$ m77cnvhp [-q] <file_name> <RETURN>
```

<file_name> is the name of MELPS 7700 Hex format file without suffix. The converter program will read the MELPS 7700 Hex format file (with .hex suffix) and the symbol file (with .sym suffix). It will generate the following HP format files:

- HP Absolute file (with .X suffix)
- HP Linker symbol file (with .L suffix)
- HP Assembler symbol files (with .A suffix)

When the **-q** option is specified, warning messages are suppressed.

Suppose that you have the following two files:

sample.hex (MELPS 7700 Hex format file)
sample.sym (Symbol file)

You can generate HP format files from these two files with the following command:

```
$ m77cnvhp sample <RETURN>
```

Note

The converter uses both .hex file and .sym file. You need to direct your assembler and linker to generate .sym file.

Specifications

The following are specifications of the format converter.

- Label names and Symbol names must be 15 and less characters in length.
- File name must be 14 and less characters in length.
- Up to 10000 sections can be handled.
- Up to 1000 functions can be handled.
- If a label name or symbol name contains "?", it will be replaced with "_".

Note

When you convert files which contain no local symbols, the assembler symbol files (.A file) won't be generated. In this case, you will see an error message when you load the program into the emulator. However, this error will cause no damage on your operation.

Index

Notes