HP 64753

Z80 Emulator Softkey Interface

User's Guide

HP Part No. 64753-97000
Printed In U.S.A.
June, 1990

Edition 2

Certification and Warranty

Certification

Hewlett-Packard Company certifies that this product met its published specifications at the time of shipment from the factory. Hewlett-Packard further certifies that its calibration measurements are traceable to the United States National Bureau of Standards, to the extent allowed by the Bureau's calibration facility, and to the calibration facilities of other International Standards Organization members.

Warranty

This Hewlett-Packard system product is warranted against defects in materials and workmanship for a period of 90 days from date of installation. During the warranty period, HP will, at its option, either repair or replace products which prove to be defective.

Warranty service of this product will be performed at Buyer's facility at no charge within HP service travel areas. Outside HP service travel areas, warranty service will be performed at Buyer's facility only upon HP's prior agreement and Buyer shall pay HP's round trip travel expenses. In all other cases, products must be returned to a service facility designated by HP.

For products returned to HP for warranty service, Buyer shall prepay shipping charges to HP and HP shall pay shipping charges to return the product to Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to HP from another country. HP warrants that its software and firmware designated by HP for use with an instrument will execute its programming instructions when properly installed on that instrument. HP does not warrant that the operation of the instrument, or software, or firmware will be uninterrupted or error free.

Limitation of Warranty

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environment specifications for the product, or improper site preparation or maintenance.

No other warranty is expressed or implied. HP specifically disclaims the implied warranties of merchantability and fitness for a particular purpose.

Exclusive Remedies

The remedies provided herein are buyer's sole and exclusive remedies. HP shall not be liable for any direct, indirect, special, incidental, or consequential damages, whether based on contract, tort, or any other legal theory.

Product maintenance agreements and other customer assistance agreements are available for Hewlett-Packard products.

For any assistance, contact your nearest Hewlett-Packard Sales and Service Office.

Notice

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

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

IBM and PC AT are registered trademarks of International Business Machines Corporation.

MS-DOS is a trademark of Microsoft Corporation.

UNIX is a registered trademark of AT&T.

Hewlett-Packard Company
Logic Systems Division
8245 North Union Boulevard
Colorado Springs, CO 80920, 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	64753-90902, January 1989 E0189
Edition 2	**64753-97000, June 1990**

Using this Manual

Organization

- Introducing the Z80 Softkey Interface - Chapter 1
- Getting Started - Chapter 2
- Configuring the Emulator - Chapter 3
- Using the Emulator - Chapter 4
- Index

Use this manual with the *Softkey Interface Reference*.

Conventions Used

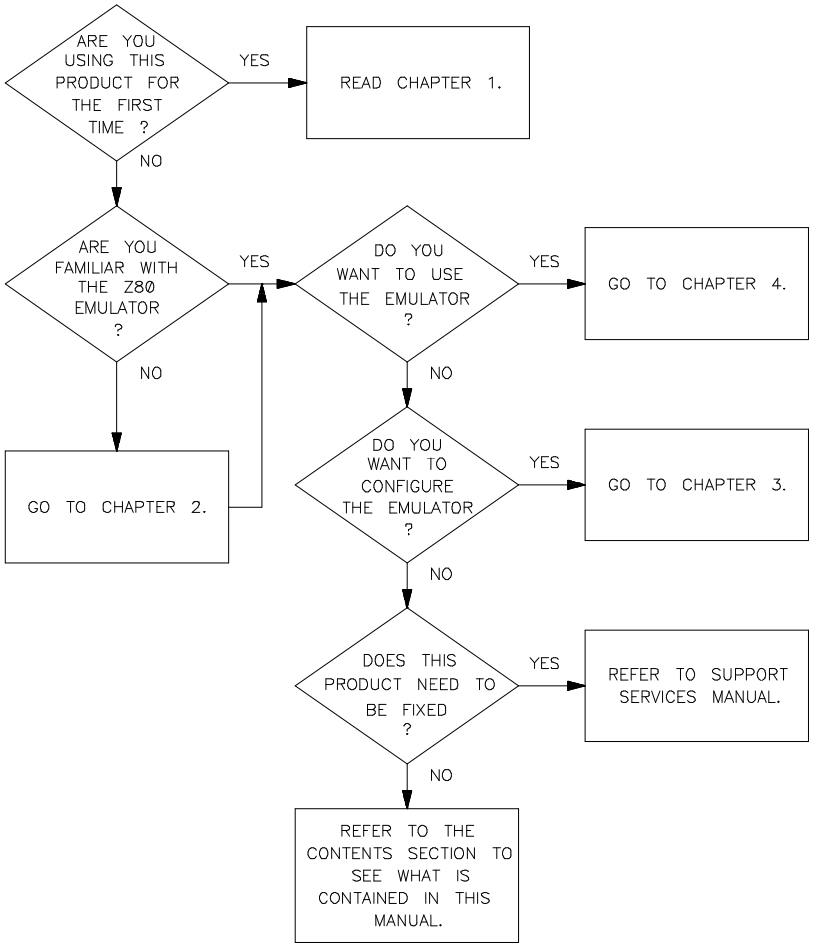
Examples in this manual use these conventions:

run from START <RETURN>

run from	Softkeys are in bold italic type.
START	Entries you make are in normal text.
step	Bold type signifies commands and options in text.
<RETURN>	Press the keyboard **Return** key.

Where to Start

Follow the diagram below to decide where to start.



Refer to the Maps

The *HP 64700-Series Manual Maps* also can help you get started. The maps are in the package marked *Read Me First.*

Contents

1 Introducing the Z80 Softkey Interface

Introduction 1-1
About the Z80 Emulator 1-1
Emulator Features 1-2
In-Circuit versus Out-of-Circuit 1-2
Tasks Performed by the Emulator 1-2
Real-Time versus Nonreal-Time 1-3
Restricted to Real-Time 1-4
Nonreal-Time 1-4
Activity Occurring while Your Program Runs 1-4
Precautions while Using the Emulator 1-5
Power Down Target System 1-5
Verify User Probe Orientation 1-5
Protect Against Static Discharge 1-5
Getting Help 1-5
Before Getting Started 1-6

2 Getting Started

Introduction 2-1
Overview of the Emulation Process 2-1
An Example for Getting Started 2-1
Example Program 2-2
Function of the Example Program 2-2
Copy the Example Program 2-3
Assemble the Example Program 2-3
Link the Example Program 2-3
Access the Emulator 2-4
From the HP-UX shell 2-4
From the Measurement System 2-5
Using the Softkeys 2-5
Map Memory 2-6
Create a Command File 2-8
Use the Emulator 2-8
Load and Run the Example Program 2-9

Observe Registers 2-12
Single-step Through the Program 2-13
Trace Program Execution 2-14
Observe the Command File 2-15
Use the Command File 2-16
End the Emulation Session 2-16
Release the Emulation System 2-17
Continue the Session Later 2-17
Keep the Emulator Locked 2-17
Select Another Measurement System 2-17
Emulator Operation After an end Command 2-17
How to Enter Numeric Values 2-18

3 Configuring the Emulator

Introduction 3-1
When to Modify the Emulation Configuration 3-1
Emulation Configuration Questions 3-2
How to Modify the Configuration 3-2
Microprocessor clock source? 3-4
Enter monitor after configuration? 3-4
Restrict to real-time runs? 3-5
Modify memory configuration? 3-5
Mapping Memory 3-6
Modify emulator pod configuration? 3-6
Enable BUSREQ input from target system? 3-7
Respond to target system Maskable Interrupt? 3-8
Respond to target system Non-Maskable Interrupt? 3-9
Enable quick-break mode? 3-9
Enable WAIT input during emulation memory accesses? 3-10
Write data to target system during emulation memory reads? 3-10
Drive background cycles to target system? 3-10
Value for address bits A15-A12 during background cycles? 3-11
Modify debug/trace options? 3-11
Break processor on write to ROM? 3-11
Trace background or foreground operation? 3-11
Trace refresh cycles? 3-12
Trace busack cycles? 3-12
Modify simulated I/O configuration? 3-12

Enable polling for simulated I/O? 3-13
Simio control address 1? 3-13
File used for standard input? 3-13
File used for standard output? 3-13
File used for standard error? 3-14
Enable simio status messages? 3-14
Modify external analyzer configuration? 3-14
Modify interactive measurement specification? 3-14
Do you want to modify the interactive measurement specification? 3-15
Should BNC drive or receive Trig1? neither 3-16
Should CMBT drive or receive Trig1? neither 3-16
Should BNC drive or receive Trig2? neither 3-16
Should CMBT drive or receive Trig2? neither 3-16
Should Emulator break receive Trig2? no 3-16
Should Analyzer drive or receive Trig2? neither 3-16
Configuration file name? 3-17
How to Load a Configuration File 3-17

4 Using the Emulator

Introduction 4-1
Using the Emulator Out-of-Circuit 4-2
Using the Emulator In-Circuit 4-3
Real-Time Operation Restrictions 4-4
Load Files 4-4
Displaying Symbols 4-5
Global 4-6
Local 4-7
Displaying Data 4-8
Run a Program 4-8
Trace Program Execution 4-9
Stop a Trace 4-10
Display Emulation Resources 4-11
Display Memory 4-11
Display Registers 4-14
Display Software Breakpoints 4-17
Display I/O Ports 4-18
Step through a Program 4-20
Modify Emulation Resources 4-22
Modify Memory 4-22
Modify Registers 4-24

Modify Software Breakpoints 4-26
Modify I/O Ports 4-28
Modify the Configuration 4-28
Storing Information 4-28
Copying to the Printer 4-29
Other Commands that Control the Emulator 4-29
Reset the Emulator 4-29
Send CMB EXECUTE to the CMB 4-30
Specify a Run or Trace 4-30
Execute a Pod Command 4-30
Make Performance Measurements 4-30
Wait 4-31

Index

Illustrations

Figure 2-1. Example Z80 Program (getstart.S) 2-2
Figure 3-1. Should You Modify the Configuration? 3-3
Figure 3-2. Emulation and Target System Memory 3-7
Figure 3-3. Example Memory Map 3-8
Figure 3-4. Modify Interactive Measurements 3-15
Figure 4-1. Using the Emulator Out-of-Circuit 4-2
Figure 4-2. Using the Emulator In-Circuit 4-3

1

Introducing the Z80 Softkey Interface

Introduction

The Z80 Emulator Softkey Interface allows you to operate the HP 64753 Z80 Emulator by pressing softkeys. The Softkey Interface operates on the HP 9000 host computer.

If you have used HP emulators previously, you may find that operating this product is similar. They are all softkey-driven.

About the Z80 Emulator

The HP 64753 Z80 Emulator is an 8-bit emulator that replaces the Z80 microprocessor in your target system. You may use the emulator for software development of your programs before your target system is completed. You can execute your programs on the emulator with complete control of memory and registers. You can analyze and debug your programs by making trace measurements using the emulation analyzer.

The Z80 emulator performs just like the Z80 microprocessor. You control the emulator from a terminal, personal computer (PC), or HP 9000 host computer. You can access and modify emulation and target system registers and locations or blocks of memory with the emulator. You also can step through programs to execute the code instruction-by-instruction.

When target system hardware is complete, you can use the emulator (either separately or with other products) to integrate the target system hardware and software.

Emulator Features

These are the features of the HP 64753 Z80 Emulator:

- Reset and run/stop control of the emulator.
- Software control of target system memory mapping.
- Symbolic debugging capability with both assembly and high-level language programs.
- Real-time emulation without wait states.
- Simulated I/O for emulator access to the HP 9000 resources (disk files, printer, keyboard, display and RS-232 port).
- Fast downloading of programs into emulation and/or target system memory.
- Analysis for complex program tracing and debugging.

In-Circuit versus Out-of-Circuit

Chapter 4 contains explanations of in-circuit and out-of-circuit emulation, and examples of how to use the emulator.

Tasks Performed by the Emulator

The emulator can help you with software and hardware debugging and system integration. You accomplish these tasks using the basic emulator features listed and described in table 1-1.

Task	Description
Program Download and Execution	Programs developed on the host computer or PC (using an editor, and compiler or assembler and linker) can be downloaded to memory using the emulator.

Task	Description
Run and Stop Controls	Programs may be executed from address or symbolic locations. The emulator stops running when you cause it to break into background with the break command, or when you reset the emulation processor with the reset command.
Memory Display and Modify	You can display and modify locations or blocks of emulation or target system memory.
Register Display and Modify	You can display and modify the contents of emulation processor and target system registers**.**
Analysis	You can capture and display activity on the emulation processor bus using the emulation analyzer.
Program Stepping	You can execute a user program one or more instructions at a time, and observe the contents of registers between instruction executions.
Memory Mapping	You can map any or all of the emulation processor address space to 256-byte blocks of emulation RAM or ROM, or user RAM or ROM, or guarded memory.
Memory Characterization	User and emulation memory can be configured as RAM or ROM. You can define memory as ROM to test ROM code without using ROM hardware.
Breakpoint Generation and Error Detection	The emulator transfers program execution to background when it encounters a software breakpoint or illegal opcode during writes to ROM and guarded memory accesses.
Clock Source Selection	You can select to use the internal emulation clock or the target system clock as the clock source for the emulator**.**

Real-Time versus Nonreal-Time

Two modes are available for operating the emulator: real-time and nonreal-time. Real-time refers to the continuous execution of the target system program without interference from the host computer, except

by your request. Interference occurs when you break emulator execution to background, or when a break occurs automatically. Whenever the emulator is running in background it is no longer executing your program in real-time.

Restricted to Real-Time

Emulator features performed in real-time mode include running and tracing.

Nonreal-Time

When the emulator is not restricted to real-time operation, it can perform all other tasks while your program is running, by temporarily breaking to the monitor. These tasks include memory accesses (displaying, loading, modifying, and storing), register accesses (displaying and modifying), single-stepping, and symbol accesses (displaying). (Memory accesses to emulation memory do not require a break to the monitor and therefore can be performed in real time.)

Activity Occurring while Your Program Runs

While your program runs, several things are happening. As the emulation processor generates address information for each cycle:

- If the memory mapper identifies a target system resource with the current address, the data path buffers between the target system and the emulation processor are enabled.
- If the address was mapped to emulation resource space, the data path buffers between the emulation processor and the emulation bus resources are enabled.
- The emulation analyzer observes activity on the emulation bus. It stores the program flow, which you can display later without interrupting the real-time flow of the program.

For information about the timing specifications of the Z80 emulator, refer to the *Z80 Terminal Interface User's Guide* .

Precautions while Using the Emulator

You should take the following precautions while using Hewlett-Packard emulators. Damage to the emulator circuitry may result if these precautions are not observed.

Power Down Target System

Turn off power to the target system and the emulation system before inserting the user probe. This helps to avoid circuit damage that can result from voltage transients or improperly inserting the probe.

Verify User Probe Orientation

Make sure to properly align pin 1 of the target system microprocessor socket and pin 1 of the user probe before inserting the probe into the socket. Failure to do so may result in damage to the emulator circuitry.

Protect Against Static Discharge

The emulator contains devices that are susceptible to damage by static discharge. Therefore, you should take precautionary measures before handling the user probe to avoid emulator damage.

Getting Help

If you need help using the Z80 emulator at any time, enter:

```
help  <HELP_FILE>  <RETURN>
```

The help file names will appear on the softkey labels. When you select one of these help files, information about that topic will appear on the screen.

Note

You can use a question mark (?) for the word “help” when accessing information about the emulator.

Before Getting Started

Before you begin using the Z80 emulator, make sure that you have completed the following preparation steps:

1. Connect the emulator to the host computer. Refer to the *HP 64700-Series Emulators Softkey Interface Installation Notice* and the *HP 64700-Series Emulators Hardware Installation and Configuration* manual.

Note

If you are using the HP 98659A High-Speed RS-422 Interface Card, and have any problems with it, refer to the *Installation Notice* for the HP 64700-Series Emulators Softkey Interface, and the *Installation Guide* provided with the HP 98659A.

2. Install the Softkey Interface software on your host computer. Refer to the *HP 64700-Series Emulators Softkey Interface Installation Notice* for details.

3. Review the *HP 64700-Series Emulators System Overview* and *Softkey Interface Reference* manuals. These should give you an understanding of HP 64700-Series emulators and the Softkey Interface.

When you are finished, continue to the next chapter to get started using the emulator.

2

Getting Started

Introduction

This chapter contains procedures to help you learn to use the emulator. The topics in this chapter include:

- An overview of the emulation process.
- An example for getting started.

Overview of the Emulation Process

The overview of emulation process outlines what you need to do to use your Z80 emulator. The *Softkey Interface Reference* elaborates on this process.

Step	Explanation
Prepare the software.	Create, assemble, and link your Z80 programs. Correct any errors that occur.
Prepare the emulator.	Modify the emulation configuration and the */usr/hp64000/etc/64700tab* file to suit your needs.
Use the emulator and analyzer.	Load a program into the emulator and run it. Use the analyzer to trace program execution.

An Example for Getting Started

This example should help you become more familiar with how to operate the Z80 Emulator Softkey Interface. The steps involved are:

1. Copy the example program to your working directory.
2. Assemble the example program.

3. Link the example program.
4. Access the emulator.
5. Map memory.
6. Create a command file.
7. Use the emulator.
8. End the emulation session.

Example Program

Figure 2-1 shows the example program used in this chapter, which comes with the Z80 Softkey Interface. When you load the Z80 Softkey Interface software, the program files are stored on the HP 9000 system. The files are in the directory:

/usr/hp64000/demo/emul/hp64753

The source file is named *getstart.S*.

Function of the Example Program

The example program shows how the Z80 features can be used as a timer or counter. Using the features of the emulator, you will learn how

```
"Z80"
;
; This example program loads four registers with values, then increments and
; decrements the appropriate registers until the count process is complete.
; The program then repeats.
;
;LABEL      OPCODE    OPERANDS          COMMENT
;
START       LD        A,1         ;SET REGISTER A TO 1
            LD        B,7         ;SET REGISTER B TO 7
            LD        DE,1111H    ;LOAD DE REGISTER
            LD        HL,0FFH     ;LOAD HL REGISTER
LOOP        INC       HL          ;INCREMENT HL REGISTER
            INC       A           ;INCREMENT A REGISTER
            DJNZ      LOOP        ;DECREMENT B AND LOOP IF B IS NOT EQUAL TO 0
JUMP        JP        START       ;REPEAT PROGRAM
```

Figure 2-1. Example Z80 Program (getstart.S)

to load the example program into emulation memory and execute it, and trace program operation.

Copy the Example Program

Copy the example program (shown in figure 2-1) to the directory you are working in. Keep the name *getstart.S*.

Assemble the Example Program

The example program was assembled with the HP 64842 Z80 Cross Assembler/Linker. To assemble the example program, follow these steps:

```
asm  -oex  getstart.S  <RETURN>
```

If errors occurred during assembly, debug the example program source file, assemble it again, and check to make sure that no errors occur.

When the example program has assembled correctly, two additional files will be created with the same base name given to the source file: the relocatable file (*getstart.R*) and the assembler symbol file (*getstart.A*).

Link the Example Program

To link the relocatable file (*getstart.R*) produced by the assembler, enter:

```
lnk  <RETURN>
```

Answer the linker questions with the responses shown in bold type:

```
object files  getstart
library files  <RETURN>
Load addresses: PROG,DATA,COMN=0,0,0  <RETURN>
more files (y or n)  n  <RETURN>
absolute file name  getstart  <RETURN>
```

The name of the example program absolute file created during the link process is *getstart.X*. The files created during the link process include:

Filename	Purpose

getstart.S	Assembly language source file.
getstart.A	Assembler symbol file.
getstart.K	Linker command file.
getstart.L	Linker symbol file.
getstart.R	Relocatable object module file.
getstart.X	Absolute file.

Access the Emulator

You can access the Z80 Emulator Softkey Interface using two different methods.

From the HP-UX shell

You can access the emulator quickly and directly from the HP-UX shell using the **emul700** command. If your PATH environment variable includes */usr/hp64000/bin*, you can use the **emul700** command directly.

For this example, "z80emul" represents the logical name specified for the Z80 emulator in the emulator device table file named */usr/hp64000/etc/64700tab*. The emulator name you defined may differ. Enter:

```
emul700  z80emul  <RETURN>
```

Note

The logical name for the emulator must be included in the emulator device table file, or else the emulator will not operate properly. In addition, switches on the rear panel of the emulator must be set correctly to correspond to the entries in the */usr/hp64000/etc/64700tab* file.

Refer to the *Softkey Interface Installation Notice* for details.

From the Measurement System

You can put your HP 64753 Z80 emulator in a measurement system. This lets you include the emulator in coordinated measurements. For more information, refer to the chapter on *Coordinated Measurements* in the *Softkey Interface Reference*.

If your emulator is part of a measurement system (as described in the *Coordinated Measurements* section of the *Softkey Interface Reference*), you can use your emulator with the HP 64808 User Interface Software (pmon).

If you PATH environment variable includes */usr/hp64000/bin*, you can use the **pmon** command directly. To configure the emulator into a measurement system using pmon, and then access the emulator, you must enter the following commands:

```
pmon  <RETURN>
MEAS_SYS  msinit  <RETURN>
msconfig  <RETURN>
make_measurement_system  z80 <RETURN>
add <MOD#> naming_it z80emul <RETURN>
end  <RETURN>
z80  default  z80emul  <RETURN>
```

MOD# is the module number of the emulator from the list displayed on screen. The *HP 64000-UX User Interface Software Manual* explains this process.

Using the Softkeys

Three rows of softkeys are available after you start the emulator. They allow you to operate the emulation system.

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

  load   store   stop trc   copy   reset   specify  cmb exec ---ETC--

pod cmd perfinit perfrun  perfend  ________ ________ ________ ---ETC--
```

Each level of softkeys contains sublevels where additional commands let you perform specific measurements and tasks. You can find detailed

information about the commands in these levels by referring to the syntax diagrams in the *Softkey Interface Reference*.

Note

When you enter numeric values, make sure to enter a standard base letter following the number. If you do not enter a base letter, the system interprets the number as decimal.

Map Memory

You must choose a memory type for the memory in the Z80 emulator, so that you can load the example program into emulation or user (target system) memory. You do this by modifying the emulation configuration.

To modify the emulator configuration, enter:

```
modify configuration <RETURN>
```

Answer the configuration questions for this example by responding to each question with the entries shown below in bold. (Chapter 3 explains the emulation configuration process.)

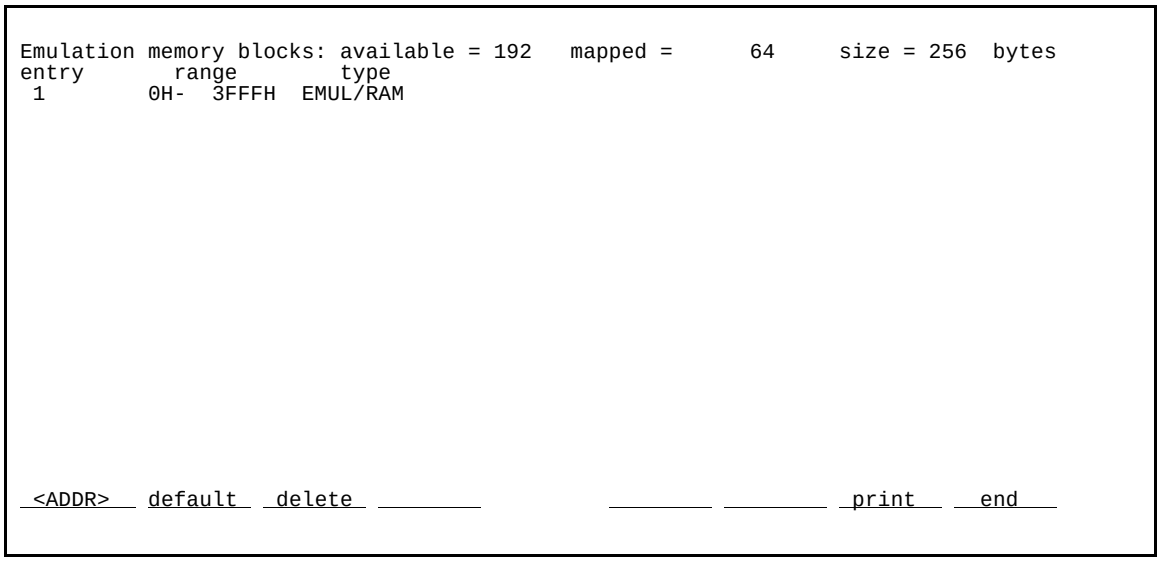
```
Microprocessor clock source?  internal  <RETURN>
Enter monitor after configuration?  yes  <RETURN>
Restrict to real-time runs?  no  <RETURN>
Modify memory configuration?  yes  <RETURN>
```

You can now modify the Z80 emulation memory map. Using the softkeys, define the memory map so that addresses 0 through 3fff hexadecimal are configured as emulation RAM. For example:

```
0  thru  3FFFH  emulation ram <RETURN>
```

The rest of memory maps to emulation RAM by default.

Your memory map should resemble:



When you finish, exit the memory map. Enter:

end <RETURN>

Continue answering the remaining configuration questions:

```
Modify emulator pod configuration?  no  <RETURN>
Modify debug/trace options?  no  <RETURN>
Modify simulated I/O configuration?  no  <RETURN>
Modify external analyzer configuration?  no  <RETURN>
Modify interactive measurement specification?  no  <RETURN>
Configuration file name?  config1  <RETURN>
```

If a file named *config1.EA* already exists, you must either overwrite the existing one or supply a different name.

These responses to the emulation configuration questions modify the emulator's default configuration with a new memory map definition.

All the answers to these questions are now stored in a configuration file named *config1.EA*. The file name extension (.EA) indicates this is a configuration file.

If you want, you can take time now to observe the content of the configuration file. To do this, enter:

```
!more  config1.EA  <RETURN>
```

You can verify that the memory map is configured as you specified. The memory map definition at the start of the configuration file should resemble:

```
BEGIN MEMORY MAP
default emulation ram
0H thru 03FFFH emulation ram
END MEMORY MAP
```

Create a Command File

Creating and using a command file is a convenient method for performing a series of tasks using the emulator features. You can create a command file by recording emulation commands in a file that you execute later. Or you can type commands into a file using an editor.

For this example, turn on the "log_commands" feature to record commands in a file. While you go through each step in this chapter, commands that you execute will be logged to (recorded in) the file until the "log_commands" feature is turned off.

To begin logging commands to a file, enter:

```
log_commands  to  CMDFILE <RETURN>
```

We will examine the command file later.

Use the Emulator

In this example, you will use the emulator to:

1. Load and run the example program.
2. Break into the monitor and observe registers.
3. Single-step through the program.

4. Trace program execution.

Press the space bar to clear the command line. Observe the status line and make sure the emulator is running in the monitor. If it isn't, enter **break <RETURN>**.

Load and Run the Example Program

You must load the example program into memory before the emulator can execute it. If you do not specify a memory type, the program will be loaded into emulation memory. To load the absolute file, enter:

```
load  getstart  <RETURN>
```

Display the contents of memory to make sure that the example program has loaded properly. Enter:

```
set symbols on
display  memory  0  mnemonic  <RETURN>
```

The listing of emulation memory should resemble:

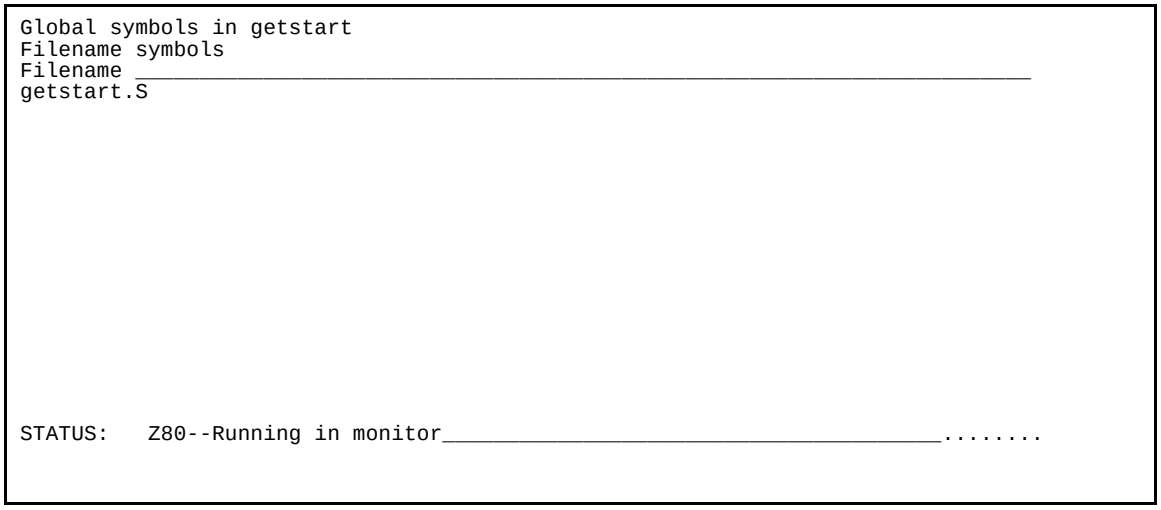
```
 Memory  :mnemonic :file = getstart.S:
   address  label          data
      0000  getsta:START  3E01        LD A,01
      0002                0607        LD B,07
      0004                111111      LD DE,1111
      0007                21FF00      LD HL,00FF
      000A  getstar:LOOP  23          INC HL
      000B                3C          INC A
      000C                10FC        DJNZ |getstart.S:LOOP
      000E  getstar:JUMP  C30000      JP getstart.S:START
      0011                04          INC B
      0012                3006        JR NC,001A
      0014                80          ADD A,B
      0015                FF          RST 38
      0016                FF          RST 38
      0017                FF          RST 38
      0018                91          SUB C
      0019                FEFF        CP FF

STATUS:   Z80--Running in monitor_____________________________________.........
```

Notice that local symbols are displayed for address locations that correspond to a particular symbol.

Display global symbols in the example program by entering:

display global_symbols

The result on screen resembles:



Note that the only global symbol is the name of the source module itself. No other symbols are declared as global in the assembler program.

To display local symbols in the example program, enter:

```
display local_symbols_in getstart.S:
<RETURN>
```

The result on screen resembles:

```
Symbols in getstart.S:
Static symbols
Symbol name ____________________ Address range __ Segment _____________ Offset
JUMP                              000E            PROG                      000E
LOOP                              000A            PROG                      000A
START                             0000            PROG                      0000









STATUS:   Z80--Running in monitor          _____________________________.........
```

See chapter 4 for more information on symbols.

To make sure the program executes the instructions correctly, you will set a software breakpoint, start the emulation processor, then observe the contents of the registers. Enter:

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

Set a software breakpoint after the first pass of the loop. Enter:

```
modify software_breakpoints set 0EH
```

The status of the breakpoint is “pending.” Start the emulation processor running the program. Enter:

```
run from 0 <RETURN>
```

When the emulation processor encounters the software breakpoint, it stops execution of the example program and begins executing in the monitor.

```
Software breakpoints  :enabled  :offset =       0004
   address          label         status
      000A         getstar:LOOP  inactivated









STATUS:   Z80--Running in monitor         Software break: 0000e_________........
```

Observe Registers

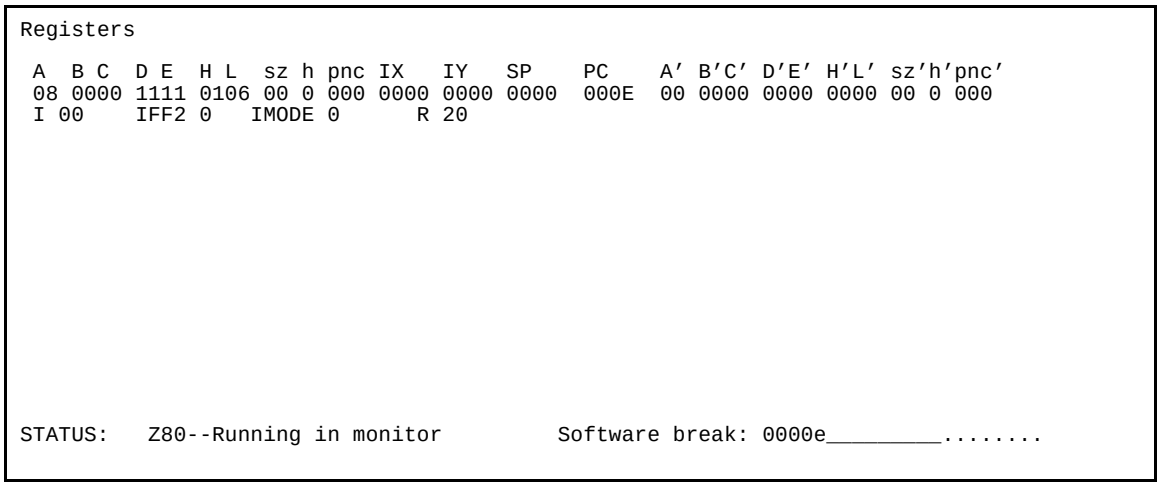
When the emulation processor is running or executing in the monitor you can observe the contents of registers.

To observe the contents of the Z80 registers, enter:

display registers ALL <RETURN>

You can observe the contents of the emulation processor registers, including flags, the stack pointer, and the program counter.

The result on screen resembles:



Some of these register values may differ on your system since the program did not initialize them.

Single-step Through the Program

To observe the contents of the processor registers as the program is running, step through the program. When you step the emulation processor through the program, you also will see the opcode, program instruction, and the next program counter address for each instruction step.

To step through the example program, enter:

```
step 1 from 0 <RETURN>
step <RETURN>
```

The first two instructions of the example program have been executed. The Z80 emulation processor registers will be displayed on screen.

The result should resemble:

```
Registers

 A  B C  D E  H L  sz h pnc IX   IY   SP    PC    A' B'C' D'E' H'L' sz'h'pnc'
 08 0000 1111 0106 00 0 000 0000 0000 0000  000E  00 0000 0000 0000 00 0 000
 I 00    IFF2 0   IMODE 0      R 20

Step_PC 0000  LD A,01
 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 01 0000 1111 0106 00 0 000 0000 0000 0000  0002

Step_PC 0002  LD B,07
 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 01 0700 1111 0106 00 0 000 0000 0000 0000  0004




STATUS:   Z80--Stepping complete_____________________________________________.........
```

To execute the remaining instructions in the example program, enter the **step** command until all instructions in the program have completed. The display shows that the registers are tracking program execution.

Trace Program Execution

You can use the emulation analyzer to trace the execution of the example program. The resulting trace is a collection of bus cycle states captured by the analyzer that you can use to view program activity.

To start a trace of the example program, enter:

trace after getstart.S:START <RETURN>

The status line shows that the trace began.

The emulation processor will begin tracing after address 0. The status line will show that a trace was started.

Set a software breakpoint after the first pass of the loop. Enter:

modify software_breakpoints set JUMP
<RETURN>

Start the emulation processor executing the program. Enter:

```
run from START <RETURN>
```

When the analyzer finds its trigger, the trace data can be displayed. To display the trace, enter:

```
display trace <RETURN>
```

An example trace listing resembles:

```
Trace List                  Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:      Address       Data            Opcode or Status          time count
Base:       symbols       hex           mnemonic w/symbols          relative
after  getstart.S:START       3E  LD A,01                          ------------
+001   getstart.S:+0001       01      01  operand                     560     nS
+002   getstart.S:+0002       06  LD B,07                           320     nS
+003   getstart.S:+0003       07      07  operand                     560     nS
+004   getstart.S:+0004       11  LD DE,1111                        320     nS
+005   getstart.S:+0005       11      11  operand                     560     nS
+006   getstart.S:+0006       11      11  operand                     360     nS
+007   getstart.S:+0007       21  LD HL,00FF                        320     nS
+008   getstart.S:+0008       FF      FF  operand                     560     nS
+009   getstart.S:+0009       00      00  operand                     400     nS
+010   |getstart.S:LOOP       23  INC HL                            280     nS
+011   getstart.S:+000B       3C  INC A                             760     nS
+012   getstart.S:+000C       10  DJNZ |getstart.S:LOOP             520     nS
+013   getstart.S:+000D       FC      FC  operand                     680     nS
+014   |getstart.S:LOOP       23  INC HL                            920     nS

STATUS:   Z80--Running in monitor         Software break: 0000e_________........
```

For more information about setting up a trace, refer to the *Analyzer Softkey Interface User's Guide.*

Observe the Command File

Because the "log_commands" feature is turned on, all commands have been recorded in a file named *CMDFILE*. Turn off the "log_commands" feature and look at the contents of the command file. Enter:

```
log_commands off <RETURN>
```

You must type "log_commands" (or the first few letters, then press **Tab**) because this option does not appear on the softkey labels. Enter:

```
!more CMDFILE <RETURN>
```

The contents of the command file should resemble:

```
load getstart
set symbols on
display memory 0 mnemonic
display global_symbols
display local_symbols_in getstart.S:
display software_breakpoints
modify software_breakpoints enable
modify software_breakpoints set 0EH
run from 0
display registers ALL
step 1 from 0
step
trace after getstart.S:START
modify software_breakpoints set JUMP
run from START
display trace
```

Use the Command File

You can use the command file as is, or you can edit the file to add, delete, or change these commands.

Note

If you add commands to the command file, make sure that you enter the commands exactly as they appear on the command line. Otherwise, an error will result.

To execute the command file, enter:

```
CMDFILE  <RETURN>
```

You can observe the commands as they execute.

End the Emulation Session

Several options are available when you end an emulation session. How you end the session depends on how you are operating the emulator. You can refer to the **end** syntax in the *Softkey Interface Reference* for more information.

For this example, either don't end the emulation session, or end the emulator locked. Then, when you enter it again later, the same configuration and memory map will be used.

Release the Emulation System

You can end the emulation session and release the emulator so that others can access it. To do this, enter:

```
end release_system <RETURN>
```

Continue the Session Later

You can end the emulation session and continue with the same session later. To do this, enter:

```
end <RETURN>
```

If you are using the emulator Softkey Interface from within a windowing environment, ending the emulator will end that individual window.

Keep the Emulator Locked

You can end the emulation session and keep the emulator locked to you. To do this, enter:

```
end locked <RETURN>
```

If you are using the emulator Softkey Interface from within a windowing environment, ending the emulator locked will end all windows.

Select Another Measurement System

You can end the current measurement system and select another measurement system if the emulator has been configured into a measurement system.

Emulator Operation After an end Command

When you end the emulation session, the emulator will remain in the last state specified. If the emulator was running when you ended emulation, it will continue to run. If the emulator was searching for a trace specification when you ended emulation, and it has not located the trigger qualifier, it will continue to search for the qualifier.

How to Enter Numeric Values

You can enter numeric values in these four standard bases: binary, octal, decimal, and hexadecimal. You must include a base letter with the number you specify, as shown below, or else the decimal base is assumed.

Numeric Value	Base
1001B	Binary
1001O or 1001Q	Octal
1001 or 1001D	Decimal
1001H	Hexadecimal

Note

If you do not supply a base letter, decimal base is assumed.

3

Configuring the Emulator

Introduction

These topics describe the emulation configuration process.

- When to Modify the Emulation Configuration
- Emulation Configuration Questions
- How to Load a Configuration File

When to Modify the Emulation Configuration

Microprocessor-based target systems, like the one you might be designing, have certain defined resources. One target system will operate differently from another, depending on the design.

The emulation configuration questions allow you to configure the emulator to work with target system resources. For example, to use the emulator with a target system, you must configure the emulator to work with the clock and memory resources of the target system.

The emulator contains memory that can be used if target system memory is not yet available. To use emulation memory as target system memory, you must define the mapping of memory resources.

Once you start modifying the emulation configuration, you must supply answers to all configuration questions that appear, whether you are using a target system or not. If your selections are the default answers, just press the <RETURN> key to make the selection.

Emulation Configuration Questions

These questions are used to configure the Z80 emulator Softkey Interface. Default answers are shown in bold.

Microprocessor clock source? **internal**

Enter monitor after configuration? **yes**

Restrict to real-time runs? **no**

Modify memory configuration? **no**

Modify emulator pod configuration? **no**

Modify debug/trace options? **no**

Modify simulated I/O configuration? **no**

Modify interactive measurement specification? **no**

Modify external analyzer configuration? **no**

Configuration file name?

These questions are explained on the following pages. Decide if you need to modify the emulation configuration by following figure 3-1.

How to Modify the Configuration

To begin modifying the emulation configuration, enter the Softkey Interface. See chapter 2 if you need help doing this. Then, enter:

```
modify configuration <RETURN>
```

The configuration questions are explained in detail throughout the rest of this chapter. Details about loading a configuration file are included at the end of this chapter.

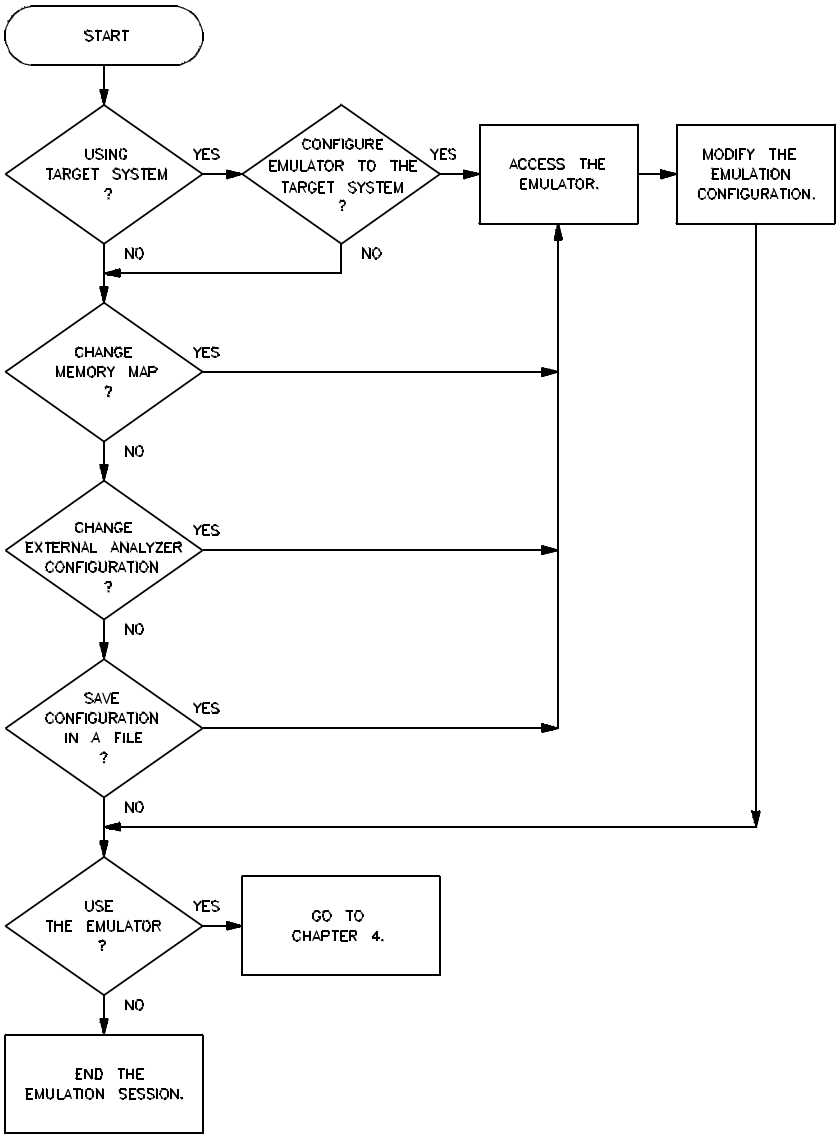


Figure 3-1. Should You Modify the Configuration?

Microprocessor clock source?

internal	Selects the 8 MHz clock source in the Z80 emulator.
external	Choose “external” to select the clock source in the target system. External clock speeds up to a maximum of 10 MHz are supported. When using the internal clock of the emulator, code execution time is relative to the internal clock speed. The internal clock specification is selected when you perform out-of-circuit emulation (for example, while debugging software without a target system).

Enter monitor after configuration?

yes	When you choose “yes,” the emulator will begin executing in background after you modify the configuration.
no	When you choose “no,” the emulator will not begin executing in background after you modify the configuration.

Note

When the external clock is selected and the target system is powered off, answer “no” to this question. Otherwise the configuration will fail.

Restrict to real-time runs?

no	When you answer “no” to this configuration question, the emulator is not restricted to real-time operation. This means that the emulator will break and then resume when displaying or modifying user memory, registers, or I/O ports, or modifying software breakpoints in user memory.
yes	Restricts the emulator to real-time runs. This means that displaying or modifying user memory, registers, or I/O ports is allowed only when the Z80 processor is executing in background. A **run from <ADDRESS>** command will break the emulator into background. When the emulator is restricted to real-time runs, simulated I/O is not available, and run control commands are not inhibited.

By restricting the emulator to real-time runs, the user program will not experience any interference once the program starts running. When restricting the emulator to real-time runs, breaks can be generated by the emulation analyzer. You also can press **break** to stop program execution and begin running in background.

Modify memory configuration?

no	If you choose “no,” all 64 kilobytes of memory in the emulator will default to emulation RAM. If the memory map was previously configured, those definitions will remain unchanged.
yes	If you answer “yes, the memory map appears. This displays the available blocks of emulation memory, the number of blocks that are mapped,

and the block size. Now, you can map emulation and user memory as desired. After you have set up the memory map and want to map memory again, you must delete the entry before the space can be mapped again. If you try to map a block of memory that is already mapped, the emulator will respond with a message indicating that a duplicate definition exists for the address you specify. You must delete the memory map term, then redefine those locations.

See the *Softkey Interface Reference* for more information on memory mapping commands.

Mapping Memory

To perform emulation with a target system connected, the memory mapper must be programmed to correspond to emulation and user memory resources. Figure 3-2 shows an overview of emulation and target system memory.

Reads or writes to guarded memory will cause the emulator to break into background. If user or emulation ROM is specified, it is possible to generate a break on write attempts to those address locations.

After configuring the emulation memory map as described in chapter 2, your map should resemble the one shown in figure 3-3.

Modify emulator pod configuration?

This question defines how the emulator will interact with the target system. Answers to these questions may differ between various Z80 target systems, but will typically remain constant for any given target system.

no	If you answer “no” to this question, you will bypass any modifications to the emulation configuration, and the next configuration question will be presented.

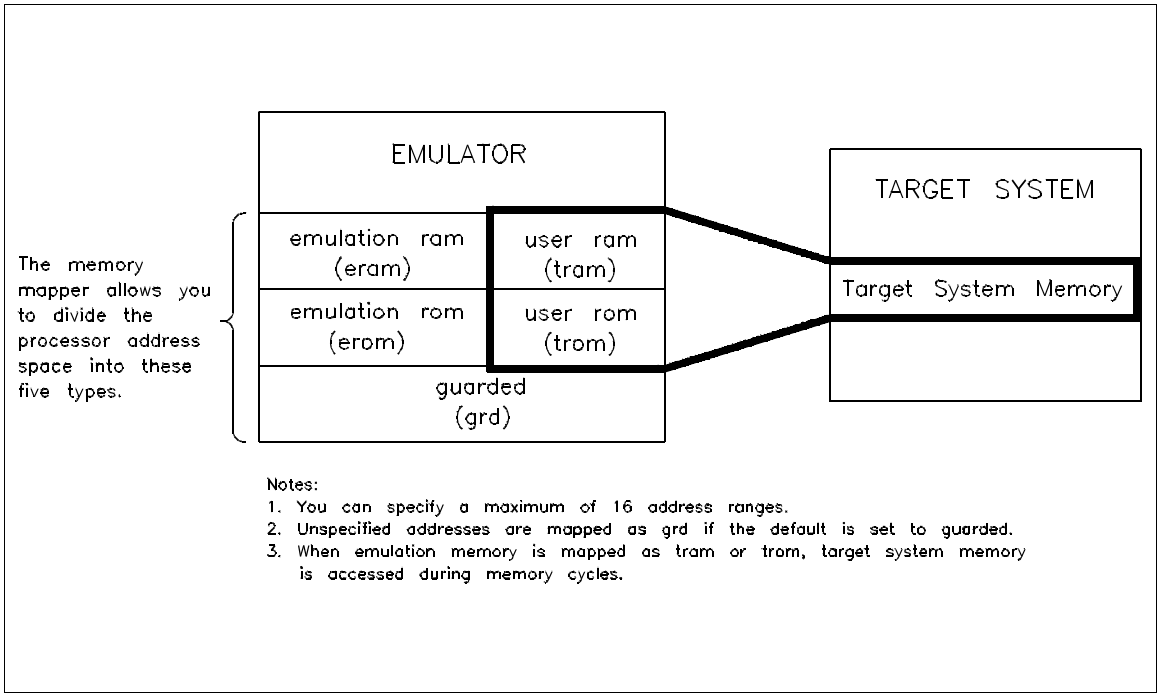


Figure 3-2. Emulation and Target System Memory

yes — If you answer “yes” to this question, you can make modifications to sublevel configuration questions that appear. You can select answers to the questions by pressing a softkey.

When you answer “yes,” the following questions will appear to allow you to configure the Z80 emulator.

Enable BUSREQ input from target system?

yes — When you select “yes” the emulator will respond to a BUSREQ input from the target system. During this time, the target system will put valid data on the data bus.

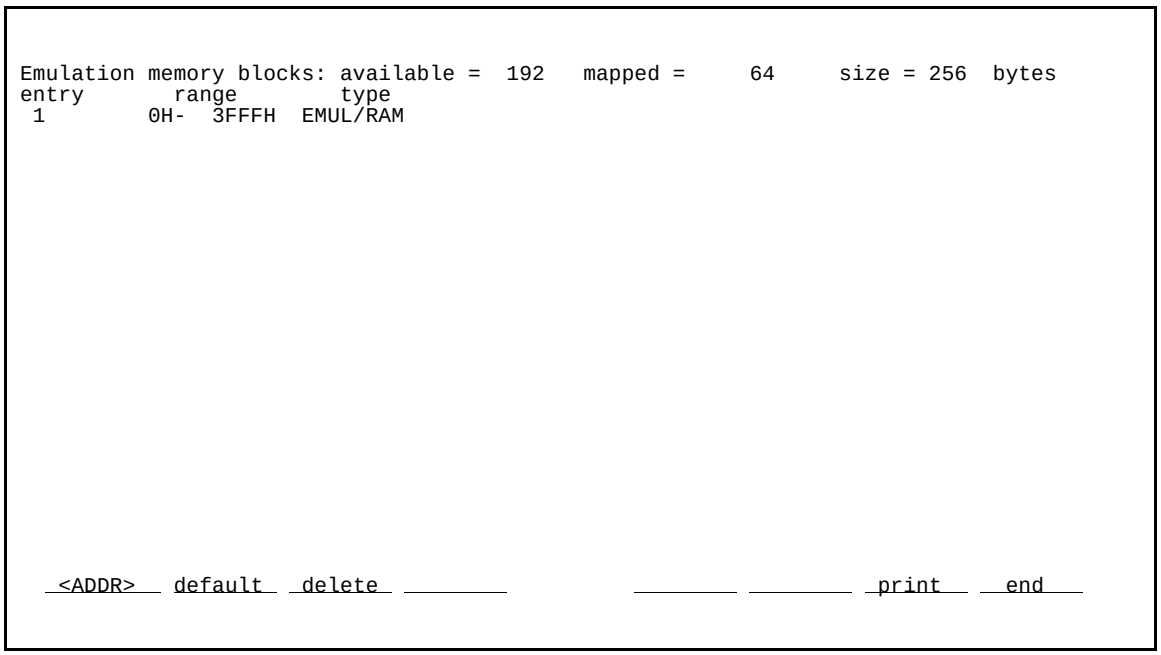


Figure 3-3. Example Memory Map

no	Select “no” if you want the emulation processor to ignore bus requests by the target system.

Respond to target system Maskable Interrupt?

yes	When you select “yes,” the emulator will respond to target system interrupt requests while it is running a user program. The emulator will ignore the interrupt request if it is running in the monitor when the request occurs.

no	Select “no” if you want the emulation processor to ignore interrupt requests by the target system.

Respond to target system Non-Maskable Interrupt?

yes	When you select “yes,” the emulator will respond to the non-maskable interrupt request from the target system while it is running a user program. If the emulator is running in the monitor when a request is received, the request will be serviced when the emulator returns to execute the user program.
no	Select “no” if you want the emulation processor to ignore non-maskable interrupt requests by the target system.

Enable quick-break mode?

no	When you select “no” the emulation processor will spend a typical amount of time in the monitor when displaying registers, I/O, or user (target system) memory. If CMB operation is enabled, other emulators on the CMB also will break to their monitors when this emulator quickly breaks.
yes	When you select “yes” the emulation processor will quickly break to the monitor to display registers, I/O, or target system memory. If CMB operation is enabled, any other emulators on the CMB will not break to the monitor when this emulator quickly breaks.

Enable WAIT input during emulation memory accesses?

yes	When you select “yes” the emulation processor will respond to a wait input by the target system during emulation memory reads and writes. Wait states are always inserted during target system memory accesses when requested.
no	Select “no” if you don’t want wait states inserted during emulation memory accesses.

Write data to target system during emulation memory reads?

no	When you select “no” the emulator will drive the data bus only during memory write and output cycles.
yes	When you select “yes” the emulator will drive the data bus to the target system (with the value read from emulation memory) during all read cycles from emulation memory. This could cause bus contention in the target system! For more information on this configuration item, refer to the *Emulator Terminal Interface User’s Guide* for the Z80.

Drive background cycles to target system?

no	When you select “no,” while the emulator is executing in the monitor, it will appear to be passive to the target system.
yes	When you select “yes,” while the emulator is executing in the monitor, the target system will recognize that the monitor program is running.

When you select “yes” to this question, the following question is presented.

Value for address bits A15-A12 during background cycles?

You can select the value that will be driven to the target system on these address lines during background monitor operation. This value can be in the range of 0 to 0FH. The default value is 0.

The entire 64-kilobyte address range is available for the user program. If bus cycles are visible to the target system while the emulator is running the monitor program, you should locate the monitor program in a memory block where memory read operations will not cause undesirable interaction with the target system.

Modify debug/trace options?

no	If you answer “no” to this configuration question, you cannot modify the debug/trace configuration.
yes	If you answer “yes” to this question, a series of four debug and trace questions will be presented. You can then modify these options as desired.

Break processor on write to ROM?

yes	Answer “yes” to this question to cause the emulator to break into background when program execution performs a write to an address mapped as ROM.
no	Answering “no” keeps the emulator from breaking when a write to ROM occurs.

Trace background or foreground operation?

foreground	When you select “foreground” the analyzer will trace emulation foreground cycles.

background	When you select “background” the analyzer will trace emulation background cycles. (This is useful only for troubleshooting emulation problems.)
both	When you select “both” the analyzer will trace both emulation foreground and background cycles.

Trace refresh cycles?

no	When you select “no,” memory refresh cycles by the Z80 will not appear in an analysis trace but will still occur (in foreground and background cycles).
yes	When you select “yes,” memory refresh cycles (both foreground and background) will appear in the analysis trace unless excluded by the trace specification.

Trace busack cycles?

no	When you select “no,” bus acknowledge cycles by the Z80 will not appear in an analysis trace but will still occur (in foreground and background cycles).
yes	When you select “yes,” bus acknowledge cycles (both foreground and background) will appear in the analysis trace unless excluded by the trace specification.

Modify simulated I/O configuration?

no	This is the default answer. If you choose “no,” modifications will not be allowed to the simulated I/O configuration.

Note 

If you restrict the emulator to real-time operation, this question does not appear during the configuration process.

yes	If you choose “yes,” a new menu for the simulated I/O configuration is displayed. The following questions will appear:

Enable polling for simulated I/O?

no	Answering “no” stops the simulated I/O configuration process.
yes	When you answer “yes,” additional questions for the simulated I/O configuration process are displayed. These questions are described in the following paragraphs.

Simio control address 1?

The first of six control addresses is SIMIO_CA_ONE by default. This and the following five addresses you specify can be used to define the simulated I/O display, printer, RS-232 communication port, and keyboard. The emulation system looks for the control addresses in data memory.

After you have defined the control addresses, the following questions are presented:

File used for standard input?

Enter the name of a file that will be used to receive input. The default file is */dev/simio/keyboard*.

File used for standard output?

Enter the name of a file that will be used to store output. The default file is */dev/simio/display*.

File used for standard error?

Enter the name of a file that will be used to store error messages. The default file is */dev/simio/display*.

Enable simio status messages?

yes	When you select "yes," command and return code status messages will be displayed in the upper right corner of the simulated I/O display.
no	When you select "no," simulated I/O status messages are not displayed. Simulated I/O operates faster when status messages are not enabled.

Refer to the *HP 64000-UX Simulated I/O Manual* for more information about Simulated I/O.

Modify external analyzer configuration?

no	If you choose "no," all modifications to the external analyzer configuration are bypassed.
yes	If you choose "yes," you can modify the external analyzer configuration.

For more information about configuring and using the external analyzer, refer to the *Analyzer Softkey Interface User's Guide.*

Modify interactive measurement specification?

You can decide whether you need to modify the interactive measurement specification by reviewing the diagram and information that follows.

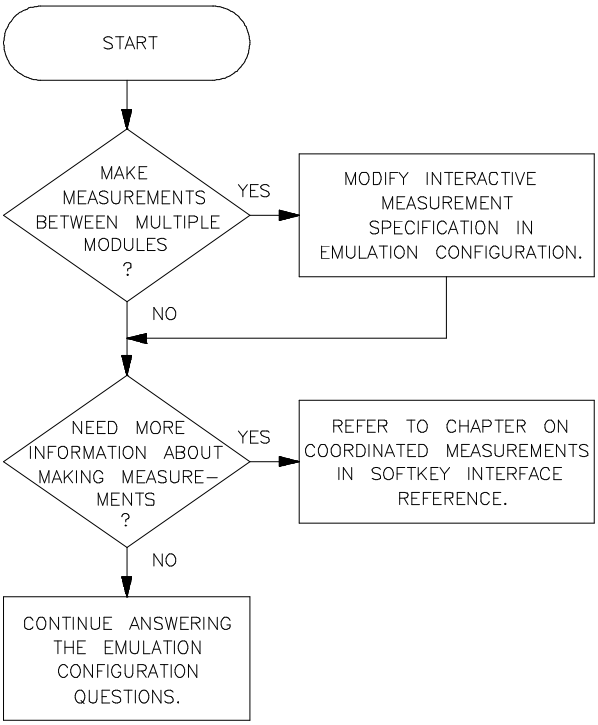


Figure 3-4. Modify Interactive Measurements

Do you want to modify the interactive measurement specification?

no	By answering “no” to this configuration question, the interactive measurement specification won’t be changed.
yes	To coordinate measurements between the modules of a multiple module system, an interactive measurement specification is required. By answering “yes,” you can modify

the driver and receiver specification for the BNC trigger, CMB trigger, emulator, and analyzer.

When modifying the interactive measurement specification, the following questions appear:

Should BNC drive or receive Trig1? neither

You can select whether the rear panel BNC connector will drive or receive the emulation analyzer trig1 signal.

Should CMBT drive or receive Trig1? neither

You can select whether the CMB trigger will drive or receive the emulation analyzer trig1 signal.

Should BNC drive or receive Trig2? neither

You can select whether the rear panel BNC connector will drive or receive the emulation analyzer trig2 signal.

Should CMBT drive or receive Trig2? neither

You can select whether the CMB trigger will drive or receive the emulation analyzer trig2 signal.

Should Emulator break receive Trig2? no

You can select whether the analyzer trig2 signal causes the emulator to break from user program execution and begin executing in the monitor.

Should Analyzer drive or receive Trig2? neither

You can select whether the analyzer drives or receives the trig2 signal.

For more information about the interactive measurement specification, refer to the chapter on *Coordinated Measurements* in the *Softkey Interface Reference.*

Configuration file name?

This allows you to save modifications to the emulation configuration in a file. You can use this file for future emulation sessions.

To save the current emulation configuration to a file, enter:

```
config2  <RETURN>
```

You can edit the configuration file using a text editor. When you reload the configuration file, the changes will be reflected in the new emulation configuration.

This allows you to use an existing configuration file as a starting point for a new configuration.

How to Load a Configuration File

You can load a previously saved configuration file into the emulator to change the emulation configuration. The options specified in the configuration file define the new emulator configuration. This saves you the trouble of reentering the modify configuration process to define or change the emulation configuration.

For example, if a configuration file named *config1.EA* exists on your host computer, and you want to load it into the emulator, enter:

```
load configuration config1  <RETURN>
```

All memory map, emulator, analyzer, and simulated I/O specifications contained in *config1.EA* are set by the emulator.

Notes

4

Using the Emulator

Introduction

This chapter contains descriptions of using the Z80 emulator in-circuit (emulator probe connected to a target system) and out-of-circuit (emulator probe not connected to a target system). For more information about using the emulator in-circuit and out-of-circuit, refer to the *Z80 Terminal Interface User's Guide.*

The examples in this chapter should familiarize you with using the Z80 Emulator Softkey Interface. These include:

- Load files.
- Display symbols.
- Run a program.
- Trace program execution.
- Display emulation resources (memory, registers, software breakpoints, and I/O ports).
- Step through a program.
- Modify emulation resources (memory, registers, software breakpoints, and I/O ports).
- Store information.
- Copying to the Printer

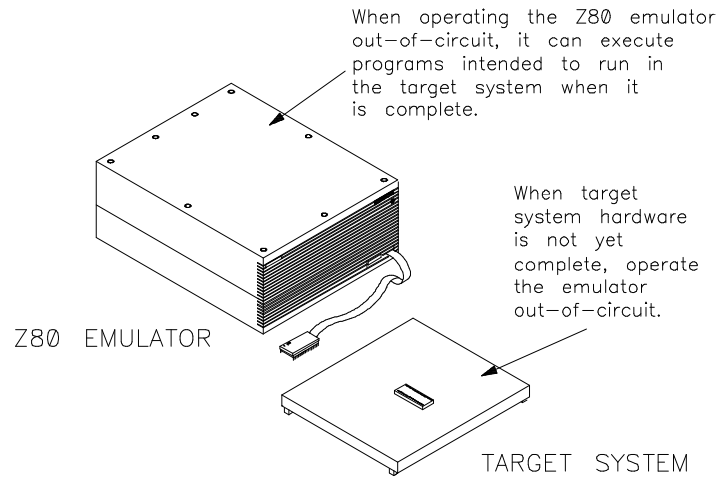
For more information about using the emulator in-circuit and out-of-circuit, refer to the *Z80 Terminal Interface User's Guide.*

Using the Emulator Out-of-Circuit

You use out-of-circuit emulation (no target system connected) to develop or debug software that will be used in a target system.

Note

If the clock speed of your target system and the clock speed of the emulator differ, program execution speed will be relative to the internal clock speed of the emulator when the internal clock source is selected.



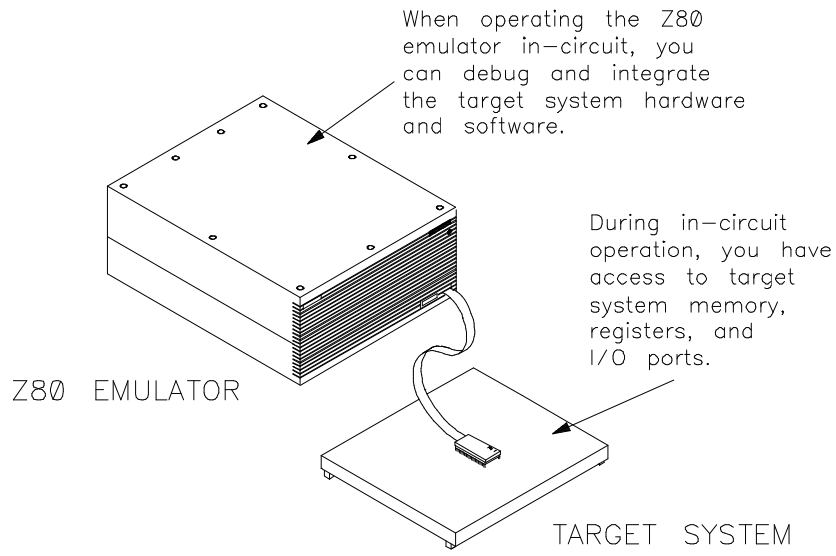
Notes:
1. Select the clock source in the emulator.
2. Program execution time is relative to the internal clock speed of the emulator.
3. Emulation memory is used in place of target system memory while the target system is being developed.

Figure 4-1. Using the Emulator Out-of-Circuit

Using the Emulator In-Circuit

The HP 64753 Z80 emulator can perform emulation in real-time or nonreal-time while connected to a target system. If real-time performance of the target system is important, use your emulator in real-time mode.

You may need to run the emulator in real-time with some target systems. Such systems typically are used for process control or other time-critical applications requiring low interrupt latency and instantaneous response. Target systems like these cannot be emulated thoroughly if real-time emulation is not specified.



Notes:

1. Select the clock source in the target system, if one is available.
2. Operate the emulator in real-time mode if the target system is dependent on a real-time clock.
3. Map memory in the target system as "tram" or "trom."

Figure 4-2. Using the Emulator In-Circuit

Real-Time Operation Restrictions

Some emulator commands cannot be performed in real-time. When the emulator is restricted to real-time operation:

1. A **run** or **step** command will cause the emulator to break into the monitor.

2. You can access registers and I/O ports only while the emulator is executing in the monitor.

3. Access to target system memory is allowed only while the emulator is executing in the monitor. You can enable or disable breakpoints in user (target system) memory only while the emulator is executing in the monitor.

Load Files

You can load programs and other types of files into memory, including configuration files, trace files, symbol files, and a background monitor program. Some examples include:

load `getstart <RETURN>`

This command loads an absolute file named *getstart.X* into memory.

load emul_mem `z80prog <RETURN>`

This command loads an absolute file named *z80prog.X* into emulation memory.

load user_mem `z80prog <RETURN>`

This command loads an absolute file named *z80prog.X* into user (target system) memory.

load configuration `z80config <RETURN>`

This command loads a previously stored configuration file named *z80config.EA* into emulation memory.

load trace `z80trace <RETURN>`

This command loads a trace file named *z80trace.TR* into the emulator. A trace file contains information captured by the analyzer during program execution.

load symbols `getstart <RETURN>`

This command loads the SRU symbol database for the module getstart into the emulation system.

load bkg_mon `monfile <RETURN>`

This loads an absolute file into background memory.

Note

The chapter about *In-Circuit Emulation* in the *Z80 Terminal Interface User's Guide* contains additional information about modifying the operation of the monitor program.

Displaying Symbols

When you load a program for the first time, the emulator uses the Symbolic Retrieval Utilities (SRU) to build a symbol database for each module. This database associates symbol names and symbol type information (not data types) with logical addresses. You will see a message on screen indicating the module for which the database is being built.

Once a symbol database is created for a particular module, it does not need to be rebuilt unless the module is changed. You can rebuild modules using the **srubuild** utility (see the *HP 64000-UX System User's Guide*). Or, if you reenter emulation without building symbols, the emulator software will automatically rebuild portions of the symbol database as you reference symbols in modified modules.

Global symbol information is immediately available for the file that you loaded. To obtain local symbol information, you need to specify the module that contains the symbols.

You can use the symbol names instead of addresses when entering expressions as part of an emulation command. Therefore, you don't have to remember segment:offset information to make a measurement. Also, the emulator can display symbols as part of a measurement, using the **set symbols on** command. This helps you relate the measurement to your original program.

The HP 64753 emulator can read absolute files in HP-OMF format. For more information on SRU, refer to the *HP 64000-UX System User's Guide*. Additional information on symbol entry syntax is in the **--SYMB--** syntax pages of the *Softkey Interface Reference*.

When you load an absolute file into memory, symbol information is also loaded (unless you use the "nosymbols" syntax). 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>

This command displays global symbols in the file loaded in emulation memory. If there are no global symbols in the program, none will be displayed.

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

```
Global symbols in getstart
Filename symbols
Filename ____________________________________________________________________
getstart.S









STATUS:   Z80--Running in monitor_____________________________________.........
```

The module names are listed under the heading "Filename symbols." For programs where several different object files are linked to form a single absolute, you will see several names listed here. You can enter these names as part of a

symbol expression to specify symbols local to a particular module.

Local

display local_symbols_in `getstart.S: <RETURN>`

This command displays local symbols in the example program. When displaying local symbols, you must include the name of the module in which the symbols are defined.

The address range, segment, and offset are displayed. The result on screen resembles:

```
Symbols in getstart.S:
Static symbols
Symbol name _____________________ Address range __ Segment _____________ Offset
JUMP                               000E            PROG                      000E
LOOP                               000A            PROG                      000A
START                              0000            PROG                      0000









STATUS:   Z80--Running in monitor_____________________________________.........
```

Displaying Data

You can display the data values of a variable in your program using the display data command. For example, suppose you want to see the value of a string at address 1000h that can be up to 32 bytes in length. Enter:

```
display data 1000h thru +32 char <RETURN>
```

You'll see the following display:

```
Data  :update
  address  label         type        data
     1000                char[]      This is a string.-.......... F.




STATUS:   Z80--Running in monitor_____________________________________........
```

The command **set symbols on** displays the label column, which indicates the symbol associated with each address. It also enables symbol display in other measurement screens, such as display memory, display registers, and display trace.

Run a Program

These commands show various ways to execute a program using the Z80 emulator.

```
run  <RETURN>
```

This command starts the emulator executing from the current program counter address.

```
run  from  0  <RETURN>
```

This command causes the emulator to begin executing from the specified address. It will continue to run until interrupted by a **break** or **reset** command, an illegal opcode, a write to ROM (if this emulator configuration item is enabled), a software break, or a guarded memory access.

run from transfer_address <RETURN>

The emulator will begin executing from the starting address of the program loaded into emulation memory. The transfer address is defined in the linker map.

run from getstart.S:START <RETURN>

The emulator will begin executing from the symbol named "START" in the user program named *getstart*, which is loaded in memory.

run from START <RETURN>

The above variation will work if the current working symbol (cws) is **getstart.S:**.

run from reset <RETURN>

This command begins execution from a reset address, or when a target system reset signal occurs, depending on your setup.

Trace Program Execution

You can use the emulation analyzer to trace program execution. During a trace, the analyzer captures information which is stored in trace memory. The **trace** command can be specified with various available qualifiers. The trace may be executed repetitively (program execution continues while the trace memory and display are continuously updated).

trace <RETURN>

This initiates a trace with the default trace specification. The analyzer will trigger on anything and will fill the trace buffer with the first 512 or 1024 states captured, depending on whether "count" is on or off. The **display trace depth** command allows you to decide how much of the trace memory to view.

trace again <RETURN>

This command starts the analyzer, without changing the trace specification.

trace repetitively <RETURN>

This command repeatedly executes the trace. Program execution will continue while trace memory and the trace display are updated.

trace after data 7 ***occurs*** 2 <RETURN>

The trace will be executed after the specified data (07H) is encountered twice during program execution.

trace before JUMP ***or*** 0EH <RETURN>

The trace will execute before the specified global or local symbol, or before the specified address, whichever occurs first.

trace only address range 0H ***thru*** 0FH <RETURN>

The trace command will store only addresses 0 through 0F hexadecimal.

trace counting state data 10H <RETURN>

The analyzer will count only instructions with the specified data value. The relative state count will be displayed.

trace break_on_trigger <RETURN>

This command will stop user program execution and cause the emulator to begin executing in background when the analyzer finds its trigger.

trace modify_command <RETURN>

The last **trace** command you executed will be displayed on the command line for your modification. Modify the command, then press <RETURN>.

Stop a Trace

To stop a currently executing trace, enter:

stop_trace <RETURN>

The status line will indicate that the trace is halted.

Display Emulation Resources

You can display emulation resources, including memory, registers, software breakpoints, and I/O ports. This section shows you how.

Display Memory

Various commands are available for displaying the contents of emulation and user (target system) memory. Some examples are shown below.

Note

When the emulator is running, and is restricted to real-time runs, you cannot display or modify user memory.

display memory getstart.S:START ***thru*** getstart.S:JUMP <RETURN>

This command displays a range of addresses starting at the symbol "START," and ending at the symbol "JUMP." The result on screen resembles:

```
 Memory  :bytes :blocked :update
   address       data       :hex                                    :ascii
      0000-07    3E   01   06   07   11   11   11   21       . . .  . . . !
      0008-0E    FF   00   23   3C   10   FC   C3           . . #   . . .









STATUS:   Z80--Running user program       Emulation trace complete______........
```

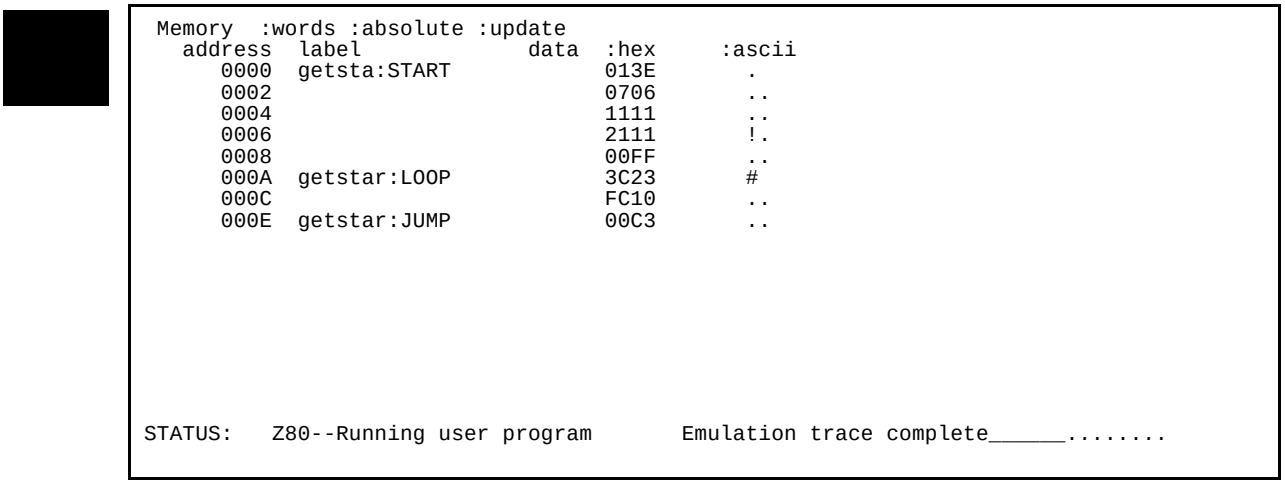
display memory mnemonic <RETURN>

This command displays addresses and mnemonic representations of the corresponding opcodes and operands. If

you do not specify a starting address or an address range, the screen will show the range of addresses previously specified.

display memory absolute words <RETURN>

Absolute addresses and corresponding data will be displayed in word format. Also displayed are ASCII representations of the data. The result resembles:



```
 Memory  :words :absolute :update
   address  label             data  :hex     :ascii
      0000  getsta:START             013E       .
      0002                          0706       ..
      0004                          1111       ..
      0006                          2111       !.
      0008                          00FF       ..
      000A  getstar:LOOP            3C23       #
      000C                          FC10       ..
      000E  getstar:JUMP            00C3       ..




STATUS:   Z80--Running user program       Emulation trace complete______........
```

display memory blocked long <RETURN>

This command displays blocked addresses and corresponding data. One long word of data will be displayed for each address location. ASCII representations of the data are also displayed. The result resembles:

```
Memory  :long words :blocked :update
  address       data       :hex                                    :ascii
     0000-0C    0706013E 21111111 3C2300FF 00C3FC10       ...!...  #......









STATUS:   Z80--Running user program       Emulation trace complete______........
```

display memory 1020h ***real*** <RETURN>

This command displays the contents of memory as real numbers. The result resembles:

```
Memory  :short real :update
  address  label             data  :real
     1020                   -2.25671E-018
     1024                    NaN
     1028                   -2.00000E+000
     102C                    3.94475E-030
     1030                    4.82321E-037
     1034                    NaN
     1038                    NaN
     103C                    1.01484E-037
     1040                    1.50956E-021
     1044                    NaN
     1048                    NaN
     104C                    1.19112E-037
     1050                    3.77227E-008
     1054                    NaN
     1058                    NaN
     105C                    1.16893E-019

STATUS:   Z80--Running user program       Emulation trace complete______........
```

(NaN means Not a Number.)

display memory absolute offset_by 10 <RETURN>

This command displays addresses in absolute format. The beginning of the list will be offset by the expression you specify.

Display Registers

By observing the emulation processor registers you can verify that your program is running properly. Assemble and link your program (see chapter 2), then load the absolute file into memory. Now, display the emulation processor registers and step through the program to observe activity in the registers.

When you display registers, the current program counter address is displayed, with instruction opcodes (if stepping), register contents, status of the flags, and the next program counter address. These commands show you how to display the emulation processor registers.

display registers <RETURN>

This command displays the contents of the BASIC emulation processor registers. To observe program activity beginning at a specified address, you can step the processor from an address, and display registers after each **step** command. The result on screen resembles:

```
Registers

 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 03 0500 1111 0102 00 0 000 0000 0000 0000  000B









STATUS:   Z80--Running user program       Emulation trace complete______........
```

display registers ALT <RETURN>

This command displays the alternate set of Z80 registers. The result on screen resembles:

```
Registers

 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 03 0500 1111 0102 00 0 000 0000 0000 0000  000B

 A' B'C' D'E' H'L' sz'h'pnc'
 00 0000 0000 0000 00 0 000









STATUS:   Z80--Running user program       Emulation trace complete______........
```

display registers INT <RETURN>

This command displays the set of Z80 interrupt registers. The result on screen resembles:

```
Registers

 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 03 0500 1111 0102 00 0 000 0000 0000 0000  000B

 A' B'C' D'E' H'L' sz'h'pnc'
 00 0000 0000 0000 00 0 000

 I 00    IFF2 0   IMODE 0









STATUS:   Z80--Running user program       Emulation trace complete______........
```

display registers ALL <RETURN>

This command displays all emulation registers. Available classes of Z80 registers include: primary, complete, alternate, and interrupt. The result on screen resembles:

```
Registers

 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 03 0500 1111 0102 00 0 000 0000 0000 0000  000B

 A' B'C' D'E' H'L' sz'h'pnc'
 00 0000 0000 0000 00 0 000

 I 00    IFF2 0   IMODE 0

 A  B C  D E  H L  sz h pnc IX   IY   SP    PC    A' B'C' D'E' H'L' sz'h'pnc'
 01 0700 1111 00FF 00 0 000 0000 0000 0000  000A  00 0000 0000 0000 00 0 000
 I 00    IFF2 0   IMODE 0      R 15




STATUS:   Z80--Running user program       Emulation trace complete______........
```

display registers BASIC DE <RETURN>

This command allows you to display an individual register or register pair (like the "DE" register pair). The result on screen resembles:

```
Registers

 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 03 0500 1111 0102 00 0 000 0000 0000 0000  000B

 A' B'C' D'E' H'L' sz'h'pnc'
 00 0000 0000 0000 00 0 000

 I 00    IFF2 0   IMODE 0

 A  B C  D E  H L  sz h pnc IX   IY   SP    PC    A' B'C' D'E' H'L' sz'h'pnc'
 01 0700 1111 00FF 00 0 000 0000 0000 0000  000A  00 0000 0000 0000 00 0 000
 I 00    IFF2 0   IMODE 0      R 15

 DE 1111



STATUS:   Z80--Running user program       Emulation trace complete______........
```

Display Software Breakpoints

You can display currently defined software breakpoints, whether they are set or cleared. Some examples are included below.

Note

If no software breakpoints are set, or if software breakpoints are disabled, a message will be displayed. To enable software breakpoints, enter the command **modify software_breakpoints enable**. See the *Modify Emulation Resources* section in this chapter for more information.

display software_breakpoints <RETURN>

This command displays all currently defined software breakpoints. The result resembles:

```
Software breakpoints  :enabled
   address         label         status
      000E         getstar:JUMP  inactivated




STATUS:   Z80--Running user program       Emulation trace complete_____.......
```

display software_breakpoints offset_by 4 <RETURN>

This command causes the software breakpoints to be offset from the actual breakpoint address by the number specified (4). The result resembles:

```
Software breakpoints  :enabled  :offset =       0004
   address          label          status
      000A          getstar:LOOP   inactivated




STATUS:   Z80--Running user program       Emulation trace complete______........
```

Display I/O Ports

You can display data at the emulation processor I/O ports. Data at these ports change, so your display may differ. Some examples follow.

display io_port 4 ***thru*** 8 <RETURN>

The contents of I/O ports 4 through 8 will be displayed. The result on screen resembles:

```
I/O port  :bytes :absolute
   address      data  :hex     :ascii
      0004             04       ..
      0005             00       ..
      0006             00       ..
      0007             00       ..
      0008             00       ..









STATUS:   Z80--Running user program       Emulation trace complete______........
```

display io_port 0 ***repetitively*** <RETURN>

The content of I/O ports 0 through 0fh will be displayed continuously. Using the **repetitively** option, you can observe what is continuously being sent to the target system I/O ports. The result resembles:

```
I/O port  :bytes :absolute :repetitively
   address      data  :hex     :ascii
      0000             00       ..
      0001             01       ..
      0002             02       ..
      0003             03       ..
      0004             04       ..
      0005             00       ..
      0006             00       ..
      0007             00       ..
      0008             00       ..
      0009             00       ..
      000A             00       ..
      000B             00       ..
      000C             00       ..
      000D             00       ..
      000E             00       ..
      000F             11       ..
STATUS:   Z80--Running user program       Emulation trace complete______........
```

Step through a Program

Stepping through a program one instruction at a time allows you to observe activity in the emulation processor registers as the program executes. In this way, you can verify that the program is executing properly. If problems exist with the program, stepping through it and observing the processor registers can help you locate the problem area.

Some examples are included below.

Note

Before stepping through a program, you should execute the **display registers** command to observe the contents of the registers. Then all successive **step** commands will automatically display registers.

step 1 ***from*** 0 <RETURN>

This executes one instruction from address 0. If the address is the start of your program, the first instruction in your program is executed. The result resembles:

```
Registers

 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 03 0500 1111 0101 00 0 000 0000 0000 0000  000A

Step_PC 0000  LD A,01
 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 01 0700 1111 0106 00 0 000 0000 0000 0000  0002









STATUS:   Z80--Running in monitor         Emulation trace complete_____.......
```

step <RETURN>

When used with the previous command, this command causes the emulation processor to execute the next program instruction. You can continue to step through the program one

instruction at a time by holding down the <RETURN> key. The result on screen resembles:

```
Registers

 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 03 0500 1111 0101 00 0 000 0000 0000 0000  000A

Step_PC 0000  LD A,01
 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 01 0700 1111 0106 00 0 000 0000 0000 0000  0002

Step_PC 0002  LD B,07
 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 01 0700 1111 0106 00 0 000 0000 0000 0000  0004




STATUS:   Z80--Stepping complete          Emulation trace complete_____.......
```

step from `getstart.S:START <RETURN>`

The emulation processor will execute one instruction from the global or local symbol you specify. (An absolute file must be loaded into emulation or user memory before you can use symbols in expressions.) The result resembles:

```
Registers

 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 03 0500 1111 0101 00 0 000 0000 0000 0000  000A

Step_PC 0000  LD A,01
 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 01 0700 1111 0106 00 0 000 0000 0000 0000  0002

Step_PC 0002  LD B,07
 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 01 0700 1111 0106 00 0 000 0000 0000 0000  0004

Step_PC 0000  LD A,01
 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 01 0700 1111 0106 00 0 000 0000 0000 0000  0002


STATUS:   Z80--Stepping complete          Emulation trace complete_____.......
```

step 4 <RETURN>

This command causes the emulation processor to execute four instructions from the current program counter address. The result resembles:

```
Registers


Step_PC 0002  LD B,07
 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 01 0700 1111 0106 00 0 000 0000 0000 0000  0004

Step_PC 0004  LD DE,1111
 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 01 0700 1111 0106 00 0 000 0000 0000 0000  0007

Step_PC 0007  LD HL,00FF
 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 01 0700 1111 00FF 00 0 000 0000 0000 0000  000A

Step_PC 000A  INC HL
 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 01 0700 1111 0100 00 0 000 0000 0000 0000  000B

STATUS:   Z80--Stepping complete          Emulation trace complete______........
```

Modify Emulation Resources

You can modify the contents of memory, registers, software breakpoints, I/O ports, and the emulation configuration. Hexadecimal numbers can be specified with “h” or “H.” Some examples are included below.

Modify Memory

These commands show you how to modify memory. When you modify memory mapped as user (target system) RAM (tram) or user ROM (trom), the modifications are made to the target system memory.

modify memory 0 ***to*** 0FFH <RETURN>

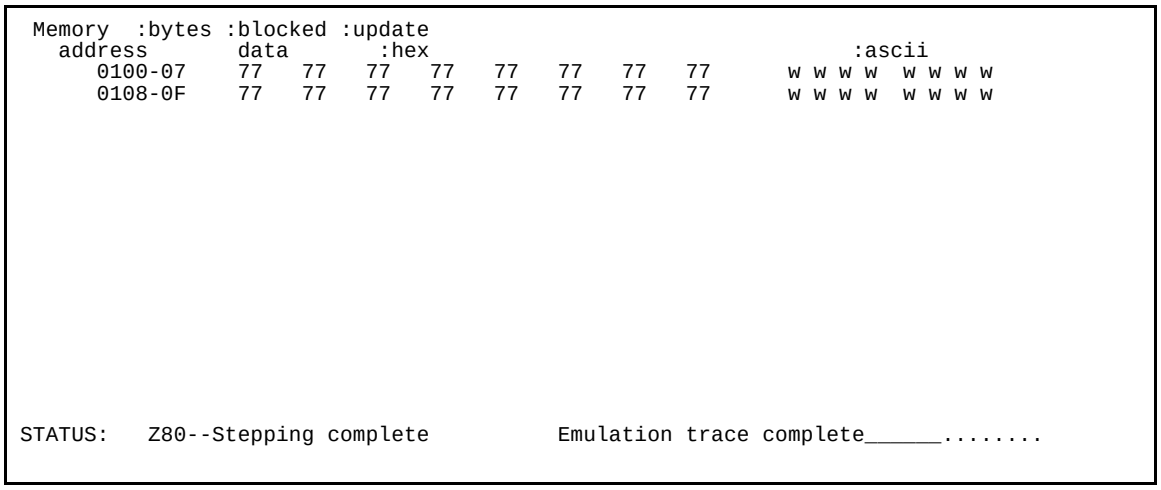
This command modifies the content of a single address (0) to the data you specify (FF hexadecimal).

modify memory 100H ***thru*** 10FH ***to*** 0 <RETURN>

This command modifies the contents of a range of addresses (100H through 10FH) to zero.

modify memory 100h ***thru*** 10fh ***to*** 77h <RETURN>

This command modifies the contents of the memory locations specified to the data specified (77h). If you use the **display memory** command to view these locations, the result resembles:



modify memory 100h ***thru*** 10fh ***to*** 1,2,3

This comand modifies the contents of the specified memory locations to the values 1,2, and 3 in sequence until the memory range is filled. The result resembles:

```
 Memory  :bytes :blocked :update
   address       data       :hex                                    :ascii
      0100-07    01   02    03   01   02   03   01   02      . . . .  . . . .
      0108-0F    03   01    02   03   01   02   03   01      . . . .  . . . .




STATUS:   Z80--Stepping complete          Emulation trace complete______........
```

Modify Registers

You can change the contents of emulation processor registers to help locate problems in your programs. These commands show you some methods for modifying registers.

Note

When the Z80 emulator is running, and is restricted to real-time runs, you cannot display or modify registers.

modify register BASIC HL ***to*** 7 <RETURN>

This command stores the value 7 in the HL register pair. If registers are already displayed, this command will automatically display the new values. Otherwise use the **display registers** command to display registers and view the contents of the HL register pair. The result resembles:

```
Registers

 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 00 0700 1111 0007 00 0 000 0000 0000 0000  000B









STATUS:   Z80--Running in monitor         Emulation trace complete______........
```

modify register A ***to*** 0 <RETURN>

This command initializes the accumulator to zero. The result resembles:

```
Registers

 A  B C  D E  H L  sz h pnc IX   IY   SP    PC
 00 0700 1111 0007 00 0 000 0000 0000 0000  000B

 A 00









STATUS:   Z80--Running in monitor         Emulation trace complete______........
```

Modify Software Breakpoints

You use software breakpoints to stop execution of a program at a particular location or locations. You can enable, set, clear, and disable software breakpoints.

Note

If you use the "LD B,B" instruction in your program and software breaks are enabled, the Z80 emulator will interpret this instruction as a software break.

modify software_breakpoints enable <RETURN>

This command turns on the software breakpoint feature. This must be done before you can set any software breakpoints.

modify software_breakpoints set 0CH , 0EH <RETURN>

This command sets two software breakpoints at addresses 12 and 14. Execution of the program will stop when the emulator encounters either of these breakpoints. If software breakpoints are not already displayed, use the **display software_breakpoints offset_by 0** command to view the breakpoints. Use **set symbols on** to enable symbol display. The result on screen resembles:

```
Software breakpoints  :enabled
   address         label         status
      000C                       pending
      000E         getstar:JUMP  pending
```

```
STATUS:   Z80--Running in monitor         Emulation trace complete_____.......
```

modify software_breakpoints clear 0CH <RETURN>

This command clears a single breakpoint at address 12. You may want to clear a breakpoint so that the program does not stop at that specific point. The result on screen resembles:

```
Software breakpoints  :enabled
   address          label          status
      000E          getstar:JUMP   pending









STATUS:   Z80--Running in monitor         Emulation trace complete______........
```

modify software_breakpoints disable <RETURN>

This command turns off the software breakpoint feature. The result on screen resembles:

```
Software breakpoints  :disabled
   address          label          status
      000E          getstar:JUMP   inactivated









STATUS:   Z80--Running in monitor         Emulation trace complete______........
```

Modify I/O Ports

You can send a data value to a specified I/O address (called a port) or range of I/O addresses. For the following commands to be valid, the emulator must be connected to a target system.

modify io_port 0FH ***to*** 011H <RETURN>

This command modifies the data at I/O port address 0FH to 11 hexadecimal.

modify io_port 1 ***thru*** 4 ***to*** 1 , 2 , 3 , 4 <RETURN>

This command sends data values 1, 2, 3, and 4 to the four I/O ports specified.

Modify the Configuration

You can view or modify the emulation configuration using this feature.

modify configuration <RETURN>

This command provides you with access to the emulation configuration questions. You can verify the current configuration, or modify answers to the configuration questions to change the configuration. Chapter 3 contains detailed information about configuring the emulator.

Storing Information

Emulation memory or trace information can be saved in files to be used for later analysis. The following commands show you how to use the store feature.

store memory 0H ***thru*** 20H ***to*** memlist <RETURN>

This causes the contents of memory locations 0 through 20H to be stored in a file named *memlist.X*.

store memory START ***thru*** JUMP ***to*** list2 <RETURN>

This command stores the contents of memory beginning at the symbol named “START,” through the address of the symbol named “JUMP,” to a file named *list2.X*

store trace tracelist <RETURN>

This command copies information stored in the trace buffer to a file named *tracelist.TR*.

Copying to the Printer

You can copy emulation information to the printer, such as the contents of memory, trace, registers, symbols, I/O ports, software breakpoints, and others. When you execute a **copy** command, you may notice that the print message writes over the command line. For example, enter:

copy display to printer <RETURN>

Does the print message overwrite the command line? If so, before copying to the printer, you may want to set the PRINTER environment variable so that the print message does not write over the command line. To do this, enter:

set PRINTER = "lp -s" <RETURN>

Execute the **copy** command again, and notice that the command you execute remains on the command line without being overwritten.

If you choose not to set the PRINTER environment variable as described here, you can refresh the command line by pressing **^L** (**CTRL L**).

Other Commands that Control the Emulator

Other commands are available that let you to perform various tasks. For more information about any of these, refer to the *Softkey Interface Reference.*

Reset the Emulator

To reset the emulator, enter:

reset <RETURN>

Target system operation is suspended while the emulator is in the reset state. In this state, you cannot display registers or target system memory. To release the emulator from the reset state, execute a **run** or **break** command.

Send CMB EXECUTE to the CMB

To produce a CMB EXECUTE signal on the CMB, enter:

```
cmb_execute <RETURN>
```

When a CMB EXECUTE pulse is sent to the CMB, all emulators connected to the CMB which are configured to respond will take part in the measurement.

Specify a Run or Trace

You can use the **specify** command to prepare a **run** or **trace** command for execution. When you use the **specify** command, the emulator/analyzer will not execute the **run** or **trace** command immediately after you enter it. You must follow that command with a **cmb_execute** command. Then the **run** or **trace** command will execute. For example, enter:

```
specify run from 0 <RETURN>
cmb_execute <RETURN>
```

Execute a Pod Command

You can send commands directly to the emulator using the command named **pod_command**. You must then execute **display pod_command** to view the result.

To display the emulator memory map, enter:

```
pod_command "map" <RETURN>
```

To display the version of firmware for your emulator, enter:

```
pod_command "ver" <RETURN>
```

Make Performance Measurements

You can make software performance measurements on your programs using the Software Performance Measurement Tool (SPMT). This post-processes information from the analyzer trace list, and stores the results in a binary file.

To initialize software performance measurements, enter:

```
performance_measurement_initialize <RETURN>
```

To run a software performance measurement, enter:

```
performance_measurement_run <RETURN>
```

To end a software performance measurement, enter:

```
performance_measurement_end <RETURN>
```

For more information on making software performance measurements, refer to the *Analyzer Softkey Interface User's Guide*.

Wait

You can cause the emulator to wait for a specified amount of time to pass, or until a measurement completes. The **wait** command suspends further command processing until one of these conditions is satisfied.

To cause the emulator to wait 30 seconds, enter:

```
wait 30 <RETURN>
```

To pause the emulator until a measurement completes, enter:

```
wait measurement_complete <RETURN>
```

Notes

Index

A
access the emulator, **2-4**
analyzer, **2-14**
analyzing programs, **1-1**
assemble the example program, **2-3**
assembler symbol file (getstart.A), **2-3**

B
base letter for numeric values, **2-18**
breakpoint, **1-3**

C
clock
  selection, **1-3**
  speed of emulator, **4-2**
  speed of target system, **4-2**
CMB EXECUTE signal, **4-30**
command file, **2-8**
configuration
  file, **2-8, 3-17**
  process, **3-1**
configure memory, **1-3**
configuring the emulator, **3-1**
copy information to printer, **4-29**
create a command file, **2-8**

D
debugging programs, **1-1**
display
  activity on emulation processor bus, **1-3**
  emulation resources, **4-11**
  I/O ports, **4-18**
  memory, **1-3, 4-11**
  registers, **2-12, 4-14**
  software breakpoints, **4-17**
downloading programs, **1-2**

E
edit the configuration file, **3-17**
emul700 command, **2-4**
emulation
  analyzer, **1-1, 2-14**

emulation (cont'd)
clock, **1-3**
configuration questions, **3-1**
memory map, **2-6**
process, **2-1**
emulator device table file, **2-4**
end
all windows, **2-17**
and keep emulator locked, **2-17**
and release emulator, **2-17**
and select measurement system, **2-17**
emulation session, **2-16**
one window, **2-17**
example
configuration file, **2-8**
for getting started, **2-1**
program (getstart.S), **2-3**
example program
files, **2-2**
example program absolute file (getstart.X), **2-3**
execute a program, **1-3, 4-8**

F features of the emulator, **1-2**

G global symbols
displaying, **4-6**
guarded memory accesses, **1-3**

H HP 64000-UX emulators, **1-1**
HP 9000 host computer, **1-1**

I I/O address (port), **4-28**
I/O ports, **4-18**
illegal opcode, **1-3**
in-circuit emulation, **4-3**

L LD B,B instruction read as software break, **4-26**
link the example program, **2-3**
linker questions, **2-3**
load
example program, **2-9**
files into memory, **4-4**
locate the problem area (in a program), **4-20**

log_commands feature, **2-8**
logical name, **2-4**

M manuals
Analyzer Softkey Interface User's Guide, **4-31**
Hardware Installation and Configuration, **1-6**
Softkey Interface Installation Notice, **1-6**
Softkey Interface Reference, **2-5, 2-16**
System Overview, **1-6**
User Interface Software Manual (pmon), **2-5**
map memory, **1-3, 2-6**
measurement system, **2-5**
measurements, **1-1**
memory
display, **1-3, 4-11**
map, **2-6**
modify, **1-3, 4-22**
modify
emulation configuration, **2-6, 4-28**
emulation resources, **4-22**
I/O ports, **4-28**
memory, **1-3, 4-22**
registers, **4-24**
software breakpoints, **4-26**

N nonreal-time mode, **1-3, 4-3**
numeric values, **2-18**

O out-of-circuit emulation, **4-2**

P performance measurements, **4-30**
personal computer (PC), **1-1**
pod_command execution, **4-30**
power down the target system, **1-5**
precautions, **1-5**
print message overwrites command line, **4-29**
PRINTER environment variable, **4-29**

R real-time mode, **1-3, 4-3**
register
display, **1-3**
modify, **1-3, 4-24**
register display and modify, **1-3**

registers, **4-14**
relocatable file (getstart.R), **2-3**
reset the emulator, **4-29**
run
  a program, **4-8**
  example program, **2-11**

S save
  current emulation configuration, **3-17**
  emulation memory in a file, **4-28**
  trace information in a file, **4-28**
set command, **4-29**
set PRINTER environment variable, **4-29**
setting up a trace (for more information), **2-15**
simulated I/O configuration questions
  control address, **3-13**
  enable status messages, **3-14**
  polling for simulated I/O, **3-13**
  standard error file, **3-14**
  standard input file, **3-13**
  standard output file, **3-13**
Softkey Interface, **1-1**
softkey levels, **2-5**
softkeys, **1-1**
software
  breakpoints, **1-3, 4-17, 4-26**
  development, **1-1**
  performance measurements, **4-30**
specify a run or trace, **4-30**
standard bases for numeric values, **2-18**
static discharge, **1-5**
step through a program, **1-3, 2-13, 4-20**
stop emulator execution, **1-3**
store information, **4-28**

T target system, **1-1**
  clock, **1-3**
  memory, **4-22**
  resources, **3-1**
tasks performed by the emulator, **1-2**
terminal, **1-1**

trace
execution of example program, **2-14**
measurements, **1-1**
program execution, **4-9**
trigger, **2-15**

U use the emulator, **2-8, 4-1**
user
probe orientation, **1-5**
RAM (tram), **4-22**
ROM (trom), **4-22**

W wait command, **4-31**
windowing environment, **2-17**
writes to ROM, **1-3**

Z Z80
emulator, **1-1**
emulator restricted to real-time runs, **4-24**
microprocessor, **1-1**

Notes