
HP 64753A/AL

Z80 Emulator PC Interface

User's Guide



HP Part No. 64753-97001

Printed in U.S.A.

October, 1990

Edition 3

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.

© Copyright 1990, Hewlett-Packard Company.

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

IBM, PC 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	64753-90903, November 1987 E1187
Edition 2	64753-90903, June 1988 E0688
Edition 3	64753-97001, October 1990

Introduction

What is the PC Interface?

You can operate your Z80 emulator with a user-friendly PC Interface that works with HP Vectra and IBM-compatible PCs.

When you operate the Z80 emulator with the PC Interface, you have all the Z80 features found in the Terminal Interface. By using the Z80 PC Interface on your Personal Computer, you can perform functions without having to remember the commands. The PC Interface leads you in the right direction.

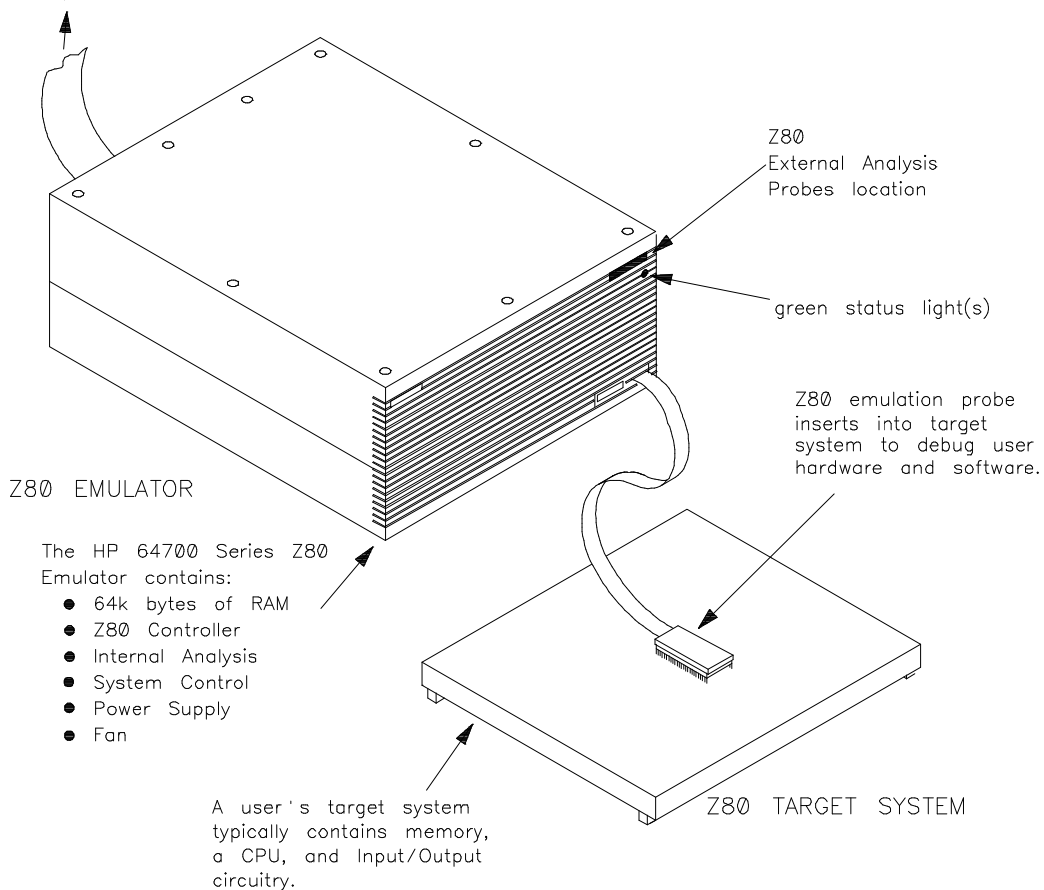
The HP 64740 PC Interface provides a “friendly user interface.” You are guided by forms and menus to control the Z80 emulator either in-circuit (in a target system) or out-of-circuit (not connected to a target system).

What’s in this Manual?

This manual, the *Z80 Emulator PC Interface User’s Guide*, explains how to use the Z80 Emulator with the PC Interface. It covers specific use of the Z80 Emulator with the PC Interface.

- Chapter 1, “Getting Started,” shows you how to start the PC Interface software and make a simple measurement.
- Chapter 2, “Use the Emulator,” shows how to make various measurements using the emulator with the PC Interface.
- Chapter 3, “Configure the Emulator,” shows you how to set emulation configuration items. You can adjust the emulation configuration to adapt the emulator to your target system design.

The HP 64700 Series Emulator can connect to a terminal, personal computer, or host computer.



The Z80 Emulator

- Chapter 4, "Solve Problems," contains information to help you fix any problems you may have while operating the emulator.
- The index contains terms and corresponding page numbers so that you can locate information quickly.

Note

If you do not understand a term in this manual, refer to the *HP 64700 Emulators Glossary Of Terms* for a definition.

Where Do I Find More Information?

You may need to use other manuals to learn about all the features of your HP 64753 Z80 emulator.

- *PC Interface Reference*
- *Analyzer PC Interface User's Guide*

The PC Interface Reference

The *PC Interface Reference* contains details about the PC Interface that you will not find in this manual. Use the *PC Interface Reference* to learn about the entire feature set of the PC Interface.

The Analyzer PC Interface User's Guide

The *Analyzer PC Interface User's Guide* gives complete descriptions of the emulation and external analyzers, and information on how to use the analyzers to help debug your programs.

Other Manuals You Can Use

The *Z80 Terminal Interface User's Guide* contains details about using the emulator in the Terminal Interface.

The *Terminal Interface Reference* describes all Terminal Interface commands.

The *Terminal Interface Analyzer User's Guide* contains details about using the emulation analyzer in the Terminal Interface.

The *CMB User's Guide* tells you how to make measurements between multiple emulators.

The *Support Services* manual contains information on what to do if you think the emulator isn't working correctly. It also tells you how to contact Hewlett-Packard for assistance.

Notes

Contents

1	Getting Started	
	Topics Covered	1-1
	Before You Begin	1-1
	Build Your Programs	1-2
	Apply Power	1-3
	Start the PC Interface	1-3
	Select PC Interface Commands	1-4
	Emulator Status	1-4
	Configure the Emulator	1-5
	Set Configuration Items	1-5
	Map Memory	1-5
	Load Programs	1-6
	Run the Emulator	1-7
	Break to Monitor	1-7
	Leave the PC Interface	1-8
	What Next?	1-8
2	Use the Emulator	
	Introduction	2-1
	Loading Programs	2-2
	What the Reader Does	2-2
	The Absolute File	2-2
	The ASCII Symbol File	2-2
	Location of the Reader Programs	2-4
	Using a Reader from MS-DOS	2-5
	HP64000 Reader	2-5
	IEEE-695 Reader	2-5
	Using a Reader from the PC Interface	2-5
	Including the Reader in a Make File	2-7
	Using Symbols	2-8
	Displaying Global Symbols	2-8
	Load and Display Local Symbols	2-10
	Using Local Symbols without Loading and Displaying	2-11
	Transferring Symbols to the Emulator	2-11

Removing Symbols from the Emulator	2-12
The Scope of Symbols	2-12
Modify and Display Memory	2-13
Modify Memory	2-13
Display Memory (Blocked Format)	2-13
Display Memory (Mnemonic Format)	2-15
When Symbol Handling is Supported	2-15
Display and Modify Registers	2-16
Register Names and Classes	2-17
Step Through the Program	2-17
Specifying a Step Count	2-19
Run the Program	2-19
Break Into the Monitor	2-20
Making Coverage Measurements	2-20
Using Software Breakpoints	2-21
Defining a Software Breakpoint	2-21
Displaying Software Breakpoints	2-22
Setting a Software Breakpoint	2-22
Clearing a Software Breakpoint	2-22
Using the Analyzer	2-22
Resetting the Analysis Specification	2-23
Specifying a Simple Trigger	2-23
Starting the Trace	2-26
Displaying the Trace	2-26
Changing the Trace Format	2-28
For a Complete Description	2-31
Reset the Emulator	2-31
In-Circuit Emulation	2-32
Installing the Emulator Probe	2-32
Precautions	2-32
Installation	2-33
Real-Time versus Non-Real-Time Operation	2-33
Emulation Memory and Target System Memory	2-34
High-Speed CMOS Target System Interface	2-34
Run from Reset	2-37
Emulation Monitor Description	2-37
Background Operation	2-37
Modifying Operation Of The Monitor Program	2-38
Make Bus Cycles Visible	2-38
Define Addresses.	2-38
Reduce Monitor Program Access Time	2-38

2-Contents

Tailoring The Monitor For Target System Interaction	2-39
Loading the User Monitor Code.	2-39
Restrictions	2-41
User Code Separate From Monitor Code.	2-41
Location of the Subroutines.	2-41
Communication With Target System I/O Ports.	2-41
You Cannot Use Some Instructions.	2-42
Registers Used by the User Monitor Code.	2-42
A Stack is Provided.	2-43
No Access To Emulation Memory	2-43
Creating/Loading The User Monitor Subroutines	2-43
Observe Monitor Operation	2-44

3 Configure the Emulator

Topics Covered	3-1
Prerequisites	3-1
Access Emulator Configuration Options	3-2
Internal emulator clock?	3-3
Enable /WAIT input from target?	3-4
Enable /BUSREQ input from target?	3-4
Enable data output to target?	3-4
Enable /INT input from target?	3-5
Enable software breakpoints?	3-5
Enable monitor cycles to target?	3-5
Enable breaks on writes to ROM?	3-6
Enable /NMI input from target?	3-6
Enable real-time mode?	3-6
Enable quick-break algorithm?	3-7
Enable tracing of refresh cycles?	3-8
Enable tracing of bus ack cycles?	3-8
Monitor block	3-8
Mapping Memory	3-9
Which Memory Locations Should Be Mapped?	3-9
Display and Modify the Memory Map	3-9
Specifying Memory Type	3-10
Modifying the Memory Map	3-11
Exit the Memory Map	3-13
Reset the Memory Map	3-14
Hardware Breakpoints	3-14
Modify the Cross Trigger Configuration	3-14
Trigger Signal Interaction	3-15

Exit the Cross Trigger Configuration	3-16
Storing an Emulator Configuration	3-16
Loading an Emulator Configuration	3-17
Where to Find More Information	3-18

4 Solve Problems

Problems with Software Breakpoints	4-1
Insufficient Memory	4-2
In the PC Interface	4-2
General Memory Problems	4-2
If You Can't Load a Program (File Format Reader Won't Run)4-2	

Illustrations

Figure 1-1. PC Interface Display	1-4
Figure 1-2. Memory Map for Sample Program	1-5
Figure 2-1. Analyzer Pattern Specification	2-25
Figure 2-2. Analyzer Trigger Specification	2-25
Figure 2-3. Analyzer Format Specification	2-29
Figure 2-4. High-Speed CMOS Interface Circuit	2-35
Figure 2-5. CMOS Interface Circuit Implementation	2-36
Figure 3-1. Emulator Configuration (Default)	3-3
Figure 3-2. Memory Map Types	3-10
Figure 3-3. Default Memory Map	3-11
Figure 3-4. Cross Trigger Configuration	3-15

Tables

Table 2-1. How to Access Variables	2-4
Table 2-2. Z80 Register Names	2-17
Table 2-3. Instructions Used With a Target System	2-42



Getting Started

Topics Covered

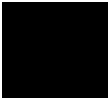
This chapter leads you through a quick setup of the emulator. When you're finished with the chapter, you'll understand the general process of setting up an emulation measurement, and will know where to go for more information.

- Before You Begin
- Build Your Programs
- Apply Power
- Start the PC Interface
- Select PC Interface Commands
- Configure the Emulator
- Load Programs
- Run the Emulator
- Break to Monitor
- Leave the PC Interface

Before You Begin

Before performing the examples in this manual, you must:

1. Install the emulator. See the *Hardware Installation and Configuration* manual for instructions. If you want to use the emulator in-circuit, see chapter 2 for emulator probe installation instructions.
2. Install the PC Interface software. See the *MS-DOS Applications Installation* manual for more information. You also should look at the *PC Interface Reference* for information on installing and modifying the emulator device table file.
3. Familiarize yourself with the PC Interface features by reading the *PC Interface Reference*.

- 
4. You should read and understand the emulation concepts presented in the *System Overview Manual*. Understanding these concepts may help you avoid questions later.

Build Your Programs

When you're working with your programs and target system, you need to compile or assemble and link those programs before you can load them into the emulator. You may use the HP 64842 Z80 Assembler to do this. The assembler comes with the PC Interface.

A sample program is on a diskette that came with your PC Interface software. You will use this program for the examples in this manual. The sample program is installed in the directory **\hp64700\demo\64753** when you install the PC Interface.

Copy the sample program files to a new directory before you work the examples. For example:

```
C> md z80demo
C> cd z80demo
C> copy \hp64700\demo\*.*
```

You should assemble and link the program files before you work the examples. To do this, enter:

```
C> make
```

The file **make.bat** is a batch file that assembles and links the program. If you're interested in the commands, print a copy of the batch file.

You should print the sample program files to use as a reference while you work the examples in this manual. The filenames are:

CMD_RDR.O	<i>Assembler output listing</i>
CMD_RDR.K	<i>Linker command file</i>
CMD_RDR.MAP	<i>Linker load map listing</i>

Apply Power

Once you've installed the emulator and emulation software, you can proceed with an emulation session.

The examples in this chapter (and the next) assume you are using the emulator out-of-circuit. Turn on emulator power by moving the rocker switch on the back of the emulator to the 1 position.

When you want to use the emulator in-circuit, you first install the target system probe. Then you turn on both target system and emulator power.

Start the PC Interface

When the emulator is powered on, you can start the emulation software to begin a measurement. To start the Z80 PC Interface, enter:

```
C> pcz80 /m emul_com1
```

pcz80 is the name of the PC Interface program for the HP 64753 emulator.

/m is an optional flag indicating that you are using a monochrome monitor. (See the *PC Interface Reference* for more information on this and other command options.)

emul_com1 is the name of the emulator in the emulator device table file. If you've connected the emulator to a different port, or you modified the device table file, substitute the proper emulator name for **emul_com1**. See the *PC Interface Reference* for more information on the emulator device table file.

If the above command is successful, you will see the display shown in figure 1-1. Otherwise, you will see an error message explaining the problem, and you will return to the MS-DOS prompt.

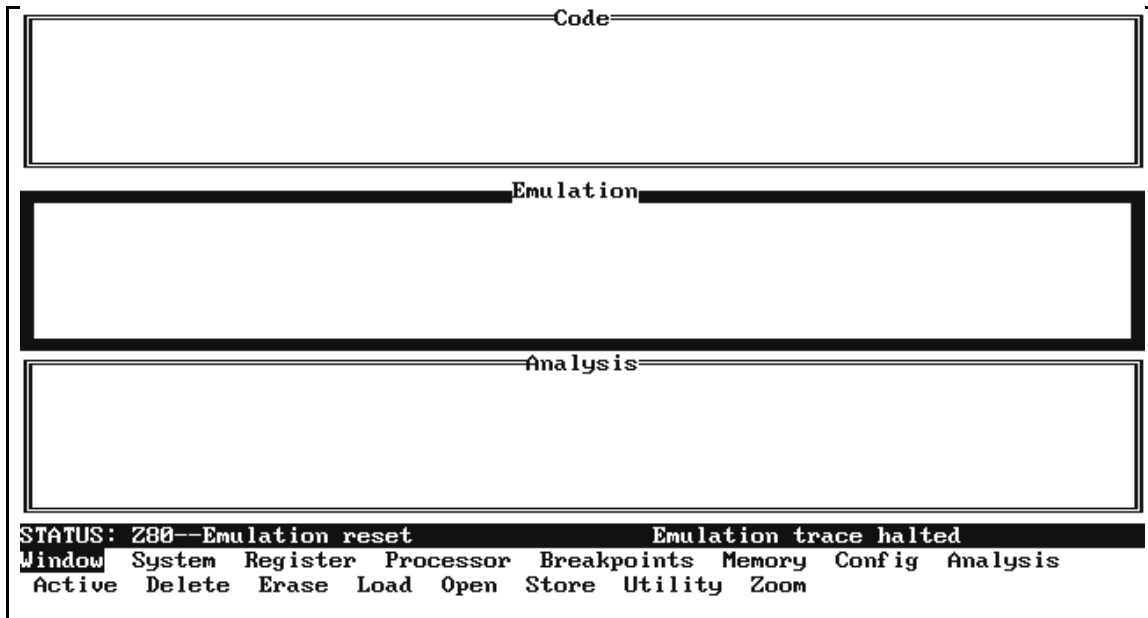


Figure 1-1. PC Interface Display

Select PC Interface Commands

You can select command options by either using the left and right arrow keys to highlight the option. Then, press the **Enter** key. Or, you can simply type the first letter of that option. If you select the wrong option, you can press the **Esc** key to retrace the command tree.

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

Emulator Status

The line above the command options displays the emulator's status. The PC Interface periodically checks the status and updates the status line.

Configure the Emulator

You need to map memory and configure the emulator before you use the emulator features. For out-of-circuit operation, the following procedure will get you started.

Set Configuration Items

You use the **Config General** command to set various configuration items such as clock source, interrupt handling, and so on. For your first use of the emulator in this chapter, the default configuration is correct. When you're ready to run your programs, you'll probably need to change one or more of these items. See chapter 3 of this manual for help on emulation configuration.

Map Memory

This assigns memory blocks to different regions of emulation or target memory.

The default available memory for your program is 0 through ffff hex. For the sample program, you will need to map one block of memory, since you load the main program at 2000 hex. To add a memory map term, enter:

```
Memory Map Configuration
Unmapped memory type eram
Term  Address Range  Memory Type
1  02000..030ff  eram
2  Empty          grd
3  Empty          grd
4  Empty          grd
5  Empty          grd
6  Empty          grd
7  Empty          grd
8  Empty          grd
9  Empty          grd
10 Empty          grd
11 Empty          grd
12 Empty          grd
13 Empty          grd
14 Empty          grd
15 Empty          grd
16 Empty          grd
<↑↓> :Interfield movement  Ctrl <↔> :Field editing  TAB :Scroll choices
STATUS: Z80--Emulation reset          Emulation trace halted
Use the TAB and Shift-TAB keys to pick memory type for unmapped ranges.
```

Figure 1-2. Memory Map for Sample Program

Config Map Modify

Use the down arrow key to move to the term next to the number 1. Type in **2000..30ff** and press **Enter**. **Tab** until the type field displays **eram**.

The memory map now looks like figure 1-2. Press **End**, then **Enter**, to save the map.

If you have questions about the configuration, memory mapping, or the emulation monitor, see chapter 3. When you're ready to use the emulator in-circuit, see chapter 2.

Load Programs

Now you need to load your programs.

To load the sample program, enter:

Memory Load

The sample program is in HP64000 format, so leave the first field as it is. Use the arrow keys to move the cursor to the last field (marked File

```
Memory Load Configuration

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

File name
cmd_rdr.l

<f1> : Interfield movement  Ctrl <f2> : Field editing  TAB : Scroll choices

STATUS: Z80--Running user program      Emulation trace halted

Use the TAB and Shift-TAB keys to select the absolute file format.
```

1-6 Getting Started

Name) and type in the name of the linker symbol file (**cmd_rdr.l**). Include the full pathname, if the file isn't in the current directory.

Press **Enter**. The PC Interface starts the File Format Reader for the HP64000 format and loads the absolute code into memory. This will take a few seconds.



Run the Emulator

You can use the PC Interface commands to run from reset, a specific location, or the current program counter address. To start the program running from the first address, enter:

```
Processor Go Address
```

Type in the symbol **INIT**, and press Enter. The emulation status will change to "Running user program."

Break to Monitor

The emulation monitor performs functions that the emulation system controller can't do by itself. The monitor is a short Z80 program that is loaded into background memory. The monitor is used to display and modify registers or target system memory, among other things.

You can force the emulator to jump to the monitor program. Enter:


```
Processor Break
```

Notice that the emulation status changes to "Running in monitor."

Leave the PC Interface

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

```
System Exit Locked
```



Another way to exit the PC Interface is with the “unlocked” option, which reinitializes the emulator the next time you start the PC Interface. You can select this option with the following command.

```
System Exit Unlocked
```

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

```
System Exit No_save
```

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

What Next?

Now that you’ve seen how to start up the emulator, you can begin using it. To learn some simple emulation measurements, go on to chapter 2.

Use the Emulator

Introduction

This chapter shows how to use the HP 64753 emulator with the PC Interface. The sample program used in chapter 1 also is used in this chapter.

This chapter will show you how to perform emulator operations and make some basic emulation measurements, including:

- Loading Programs
- Displaying and Transferring Symbols
- Modifying and Displaying Memory
- Displaying Registers
- Stepping the Processor
- Run and Break a Program
- Checking Memory Coverage
- Using Software Breakpoints
- Making an Analyzer Measurement
- Reset the Emulator
- Using the Emulator In-Circuit
- Emulation Monitor Description

Note



The examples in this chapter assume that you've configured the emulator, mapped memory and loaded the sample program as explained in chapter 1.

Loading Programs

Two *file format readers* come with the Z80 PC Interface. The readers convert either HP64000 or IEEE-695 files into two files that are usable with the HP 64753 emulator. This means that you can use available language tools to create HP64000 or IEEE-695 absolute files. You can load those files into the emulator using the Z80 PC Interface.

The reader can operate from within the PC Interface or as a separate process. If you don't have enough memory on your PC to use the PC Interface and the reader simultaneously, you may need to use the reader separately. You also can operate the reader as part of a "make file."

What the Reader Does

The readers use the following files as input:

- The IEEE-695 reader uses any IEEE-695 MUFOM (Microprocessor Universal Format for Object Modules) absolute file in the form <file>.<ext>.
- The HP64000 reader obtains information from the .X file (absolute code), .L file (linker symbols) and .A file (assembler symbols).

The readers produce two new files, an "absolute" file and an ASCII symbol file, that will be used by the Z80 PC Interface.

The Absolute File

The reader creates an absolute file (<file>.hpa). The absolute file is a binary memory image optimized for efficient downloading into the emulator.

The ASCII Symbol File

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

Note

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

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

```
module_name1
module_name2
...
module_nameN
global_symbol1  01234
global_symbol2  01567
...
global_symbolN  0ABCD
|module_name|# 1234          00872
|module_name|local_symbol1  00653
|module_name|local_symbol2  00872
...
|module_name|local_symbolN  00986
```

Symbols are sorted alphabetically in the order: module names, global symbols, and local symbols.

Line numbers will appear similar to a local symbol except that “local_symbolX” will be replaced by “#NNNNN” where NNNNN is a five digit decimal line number. Line numbers appear in ascending order corresponding to the line numbers and associated addresses.

Note

The PC Interface displays line number symbols in brackets. Therefore, the symbol “MODNAME:line 345” will be displayed as “MODNAME:[345]” in mnemonic memory and trace list displays.

You enter line number symbols by typing the following on one line in the order shown:

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

For example:

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

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

The local symbols obey static scoping rules. This means that to access a variable named "count" in a function named "Foo" in a source file module named "MAIN.C," you would enter "MAIN.C:Foo.count." See table 2-1.

Table 2-1. How to Access Variables

Module Name	Function Name	Variable Name	You Enter:
MAIN.C	Foo	count	MAIN.C:Foo.count
MAIN.C	bar	count	MAIN.C:bar.count
MAIN.C	line number 23		MAIN.C: line 23

Location of the Reader Programs

The reader is installed in the directory named `\hp64700\bin` by default, with the PC Interface. This directory must be in the environment variable `PATH` for the readers and PC Interface to operate properly. This is usually defined in the "`\autoexec.bat`" file.

The following examples assume that you have "`\hp64700\bin`" included in your `PATH` variable. If not, you must supply the directory name when executing a reader program.

2-4 Use the Emulator

Using a Reader from MS-DOS

HP64000 Reader

The command name for the HP 64000 Reader is **RHP64000.EXE**. The reader command syntax is as follows:

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

-q This option specifies the “quiet” mode, and suppresses the display of messages.

<filename> This represents the name of the HP 64000 linker symbol file (file.L) for the absolute file to be loaded. (Both the filename and the extension are required.)

The following command will create the files “TESTPROG.HPA” and “TESTPROG.HPS”:

```
RHP64000 TESTPROG.L
```

IEEE-695 Reader

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

```
RIEEE695 [-u] [-q] <filename>
```

-u Specifies that the first leading underscore of a symbol is not removed.

-q Specifies the “quiet” mode. This option suppresses the display of messages.

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

Using a Reader from the PC Interface

The Z80 PC Interface has a file format option under the “Memory Load” command. After you select the file format, the reader will operate on the file you specify. After this completes successfully, the

Z80 PC Interface will accept the absolute and symbol files produced by the Reader.

To use the Reader from the PC Interface:

1. Start up the Z80 PC Interface.
2. Map memory (if it isn't already done by your configuration file). Then select "Memory Load." The memory load menu will appear.
3. Specify the file format as "HP64000" or "IEEE-695." You can use the **Tab** and **Shift-Tab** keys to cycle through the choices.
4. Select a memory type. You can use the **Tab** and **Shift-Tab** keys to cycle through the choices:
 - **emulation** (loads address ranges mapped to emulation memory only)
 - **target** (loads address ranges mapped to target memory only)
 - **monitor** (loads optional user background monitor code; see the explanation later in this chapter)
 - **both** (loads all address ranges)
5. Specify Y or N for "Force Absolute file Read?" If you select Y, the HP64000 absolute file is always read, even if it is older than the .HPA and .HPS files with the same name.
6. Specify a file in the appropriate format. For the HP64000 format, select the linker symbol file ("TESTFILE.L," for example). For the IEEE-695 format, specify a filename <file>.<ext>. You cannot specify files with extensions of ".HPA," ".HPT," or ".HPS."

Note



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

Using the file that you specify (TESTFILE.ABS, for example), the PC Interface does the following:

- It checks to see if two files with the same base name and extensions .HPS and .HPA already exist (for example, TESTFILE.HPS and TESTFILE.HPA).
- If TESTFILE.HPS and TESTFILE.HPA don't exist, the PC Interface starts the reader. The new absolute file, TESTFILE.HPA, is then loaded into the emulator.
- If TESTFILE.HPS and TESTFILE.HPA already exist but the creation dates and times are earlier than the .ABS file creation date/time, the PC Interface starts the reader to rebuild them. The new absolute file, TESTFILE.HPA, is then loaded into the emulator.
- If TESTFILE.HPS and TESTFILE.HPA already exist, but the creation dates and times are later than those for the .ABS file, the PC Interface does not start the reader. The current absolute file, TESTFILE.HPA, is then loaded into the emulator.

Note



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

When a reader operates on a file, a status message will be displayed showing that the reader is operating and the file type that is being read. When the reader finishes, another message will be displayed showing that the absolute file is being loaded.

The PC Interface executes the reader with the “-q” (quiet) option by default.

Including the Reader in a Make File

You may want to incorporate the reader command as the last step in your “make file,” or as a step in your code construction process. Then,

whenever you modify and rebuild your program, the files necessary for PC Interface operation will be rebuilt automatically.



Using Symbols

Symbols are available when you load HP64000 or IEEE-695 files.

When you build an absolute file using the HP 64000 development tools, an assembler symbol file is created. This file has the same base name as the source file and a “.A” extension. The assembler symbol file contains local symbol information. When you link relocatable assembly modules, a linker symbol file (with the same base name as the absolute file and a “.L” extension) is created. The linker symbol file contains global symbol information and information about the relocatable assembly modules that were combined to form the absolute.

When you load a file using the HP64000 file format, the file format reader collects global symbol information from the linker symbol file and local symbol information from the assembler symbol files. It uses this information to create a single symbol database with the extension .hps. The IEEE-695 reader performs a similar function, but gets all the information from the absolute file.

The Z80 emulator PC Interface provides two methods for manipulating symbols. It allows you to use both global and local symbols to reference memory variables within the PC Interface. You also can transfer symbols to the emulator for use in displaying memory values with the “**M**emory **D**isplay **M**nemonic” command, the “**A**nalysis **D**isplay” command, and the mnemonic field of single-step commands.

Displaying Global Symbols

When you load an HP64000 format or IEEE-695 format file into the emulator, the corresponding symbol database is also loaded.

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

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

To display global symbols, select:

System Symbols Global Display

The symbols window automatically becomes active. Press <CTRL>z to zoom the window. The resulting display follows.

The global symbols display has two parts. The first part lists all the modules that were linked to produce this object file. You use these module names when you want to refer to a local symbol. The names

```

                                     Symbols
-----
Modules
-----
CMD_RDR.S

Address  Symbol
-----  -----
02400    CMD_INPUT
02000    INIT
02600    MESSAGES
02500    OUTPUT_BUFFER

STATUS: Z80--Running user program           Emulation trace halted
Window System Register Processor Breakpoints Memory Config Analysis
Command_file Wait MS-DOS Log Terminal Symbols Exit
```

are case-sensitive. The second part of the display lists all global symbols in this module. These names can be used in measurement specifications, and are case-sensitive. For example, if you want to make a measurement using the symbol **INIT**, you must specify **INIT**. The strings **init** and **Init** are not valid symbol names here.

Load and Display Local Symbols

To load and display local symbols, select:

System Symbols Local Display

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

Note

```

                                     Symbols
-----
Address  Symbol
-----
02003    CLEAR
02016    CLEAR_LOOP
0200D    CLEAR_OLD
02400    CMD_INPUT
02026    COMMAND_A
0202E    COMMAND_B
02631    END_MESSAGES
02000    INIT
02622    INVALID_INPUT
02600    MESSAGES
02600    MESSAGE_A
02611    MESSAGE_B
0203C    OUTPUT
02500    OUTPUT_BUFFER
0201C    PROCESS_COMM
02008    READ_INPUT
03000    STACK_POINTER

STATUS: Z80--Running user program           Emulation trace halted
Window System Register Processor Breakpoints Memory Config Analysis
Command_file Wait MS-DOS Log Terminal Symbols Exit
```



Emulators with system firmware below version 2.00 do not have symbol handling capability and thus have a restricted symbol command set in the PC Interface. To display local symbols in these emulators, use the command:

System Symbols Local

After you display local symbols with the “**System Symbols Local Display**” command, you can enter local symbols as they appear in the source file or local symbol display. When you display local symbols

for a given module, that module becomes the default local symbol module.

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

Using Local Symbols without Loading and Displaying

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

```
module_name : symbol
```

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

When you include the name of a source file with a local symbol, that module becomes the default local symbol module, as with the “System Symbols Local Display” command.

Transferring Symbols to the Emulator

If your emulator has system firmware version 2.0 or later, you can use the emulator’s symbol-handling capability to improve measurement displays. You do this by transferring the symbol database information to the emulator.

You transfer the global symbols to the emulator with the command:

```
System Symbols Global Transfer
```

Then the global symbol names are displayed instead of the absolute address values in subsequent memory references.

You transfer all local symbols from the absolute file using the command:

```
System Symbols Local Transfer All
```

You can transfer the local symbols from one or more modules using the “System Symbols Local Transfer Group <module_name>” command. Therefore, to transfer the local symbols from an example module named “CMD_RDR.S,” enter:

```
System Symbols Local Transfer Group
```

Enter "CMD_RDR.S," and press <Enter>. The symbols are transferred to the emulator.

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

To determine which version of emulator firmware is present, use the Terminal Interface **ver** command while in System Terminal Mode. See the *Emulator PC Interface Reference* for more information on System Terminal mode.

If your emulator does not have symbol handling capability, the symbol transfer commands will not appear in the PC Interface. Also, the displays in this chapter will appear somewhat different. For example, in the memory mnemonic example, the "Symbol" column will be replaced by a "Data" column showing the data found at that address.

Contact your local HP Sales and Service office (listed in the *Support Services* guide) for information on firmware upgrades.

Removing Symbols from the Emulator

If you transfer many symbols to the emulator, you can fill the available memory used to store symbolic references. The PC Interface allows you to remove global and local symbols to free symbol memory in the emulator.

You can delete all global symbols in the emulator with the "System Symbols **Global Remove**" command, or all local symbols with the "System Symbols **Local Remove All**" command. You can delete local symbols from one or more modules with the "System Symbols **Local Remove Group** <module_name>" command.

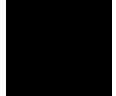
The Scope of Symbols

A symbol can be local in one module and global in another, which may cause confusion. Suppose symbol "XYZ" is defined as a global in module A and as a local in module B, and that these modules are linked to form the absolute file. After you load the absolute file (and the corresponding symbol database), entering "XYZ" in an expression refers to the symbol from module A. Then, if you display local symbols from module B, entering "XYZ" in an expression refers to the symbol from module B, *not the global symbol*.

Now, if you want to enter "XYZ" to refer to the global symbol from module A, you must display the local symbols from module A (since the global symbol is also local to that module). Loading local symbols

from a third module, if it was linked with modules A and B and did not contain an “XYZ” local symbol, also would cause “XYZ” to refer to the global symbol from module A.

You can always refer to a global symbol from the current symbol database by adding a colon before the symbol name. For example, **:XYZ** refers to the global symbol named XYZ in the current symbol database (though there may be a local symbol with the name XYZ).



Modify and Display Memory

Modify Memory

You can use the memory modification features of the PC interface to change program or data locations. For example, you can enter a command for the sample program. First, start the program:

```
Processor Go Address
```

Enter the symbol **INIT** to start from the program initialization code. Now, enter a command by selecting:

```
Memory Modify Bytes
```

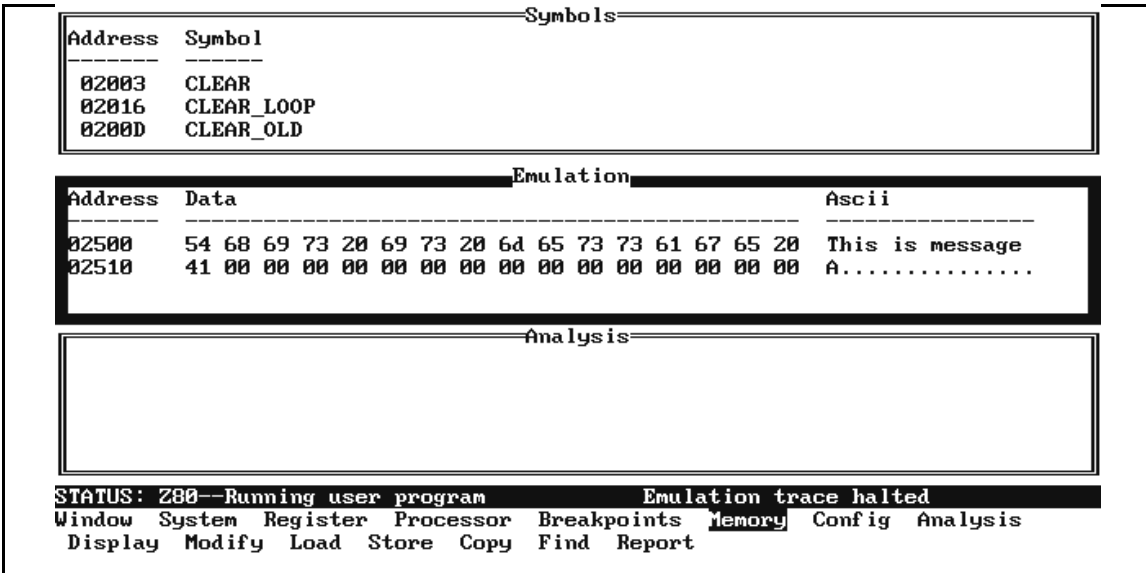
Enter the address you want to modify and the new value, such as **2400="A"**.

Display Memory (Blocked Format)

You can display memory as hexadecimal and ASCII values in byte or word formats. For example, if you gave the sample program the **A** command, there will be a corresponding message written to the output buffer. You can display the buffer by entering:

```
Memory Display Bytes
```

Now enter the address range to display. In this example, enter **OUTPUT_BUFFER..OUTPUT_BUFFER+1f**. You'll see the following display:



The screenshot displays an emulator interface with three main sections:

- Symbols:** A table listing memory addresses and their corresponding symbols.

Address	Symbol
02003	CLEAR
02016	CLEAR_LOOP
0200D	CLEAR_OLD
- Emulation:** A table showing memory addresses, their data values, and the corresponding ASCII characters.

Address	Data	Ascii
02500	54 68 69 73 20 69 73 20 6d 65 73 73 61 67 65 20	This is message
02510	41 00 00 00 00 00 00 00 00 00 00 00 00 00 00	A.....
- Analysis:** A menu of options for analyzing the program's execution.

STATUS: Z80--Running user program Emulation trace halted

Window System Register Processor Breakpoints **Memory** Config Analysis

Display Modify Load Store Copy Find Report

Display Memory (Mnemonic Format)

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

Memory Display Mnemonic

Enter the address range **INIT..INIT+30**. The emulation window automatically becomes active. You can press <CTRL>z to zoom the memory window. The resulting display follows.

```
Emulation
Address Symbol Mnemonic
-----
02000 INIT LD SP,CMD_RDR.S:STACK_POIN
02003 CMD_RDR.S:CLEAR LD HL,CMD_INPUT
02006 - LD (HL),00
02008 DR.S:READ_INPUT LD A,(HL)
02009 - CP 00
0200b - JR Z,2008
0200d RDR.S:CLEAR_OLD PUSH AF
0200e - LD DE,OUTPUT_BUFFER
02011 - LD BC,0020
02014 - LD A,00
02016 DR.S:CLEAR_LOOP LD (DE),A
02017 - INC DE
02018 - DEC C
02019 - JR NZ,2016
0201b - POP AF
0201c .S:PROCESS_COMM CP 41
0201e - JR Z,CMD_RDR.S:COMMAND_A

STATUS: Z80--Running user program Emulation trace halted
Window System Register Processor Breakpoints Memory Config Analysis
Display Modify Load Store Copy Find Report
```

To display more of the sample program, reenter the **Memory Display Mnemonic** command and change the address range.

When Symbol Handling is Supported

If your emulator supports symbol handling, when you issue a “**Memory Display Mnemonic**” command, the memory list will include symbols as shown in the display. (You must first transfer the symbols to the emulator as described earlier in this chapter.)

If your emulator supports symbols, but no symbols have been transferred to the emulator, only “-” will be displayed in the Symbol column.

Display and Modify Registers

You can display various combinations of processor registers to check the logic of your program. To display the primary set of Z80 registers, enter:

Registers Display Basic

You will see the following display:

```
-----Symbols-----
Address  Symbol
-----
02003   CLEAR
02016   CLEAR_LOOP
0200D   CLEAR_OLD

-----Emulation-----
pc = 200b      sp = 3000  ix = 0000  iy = 0000
a  = 00  f = <42,_Zx_x_M_>  bc = 0000  de = 2511  hl = 2400

-----Analysis-----

STATUS: Z80--Running user program      Emulation trace halted
Window System Register Processor Breakpoints Memory Config Analysis
Display Modify
```

To display all Z80 registers, enter:

Registers Display Class

Press the **Tab** key until **all** is displayed, and press **Enter**. Or, you can simply type in **all** and press **Enter**.

To display the alternate register set, enter:

Registers Display Class

Select **alt** as the register class and press **Enter**.

To display the Z80 interrupt registers, enter:

Registers Display Class

Select **int** as the register class and press **Enter**.

To display a single register or register pair, enter:

Registers Display Single

Tab to select a register or register pair. Or, simply enter the name of the register or register pair (such as **hl**).

You also can modify registers using the PC Interface. For example, if you want to modify the **ix** register to 254C hex, enter:

Registers Modify ix

Type in **254c** and press **Enter**.

Register Names and Classes

Table 2-2 lists the register names and classes that may be used with the **Registers Display** and **Registers Modify** commands.

Table 2-2. Z80 Register Names

Register Class	Register Name
basic	pc, sp, af, bc, de, hl, ix, iy
int	i, iff2, imode
alt	a', f', bc', de', hl'
all	All the above registers, plus the r register.

Step Through the Program

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

Processor Step Address

Enter a step count of 1, enter the symbol **INIT** (defined as a global in the source file), and press **Enter** to step from the address of **INIT**. The emulation window will automatically become the active window. Press **<CTRL>z** to view a full screen of information. The following display

shows the executed instruction, the program counter address, and the resulting register contents. Some registers in your display may differ (the program does not initialize or use all the registers).

```
Emulation
02000 INIT LD SP,CMD_RDR.S:STACK_POIN
PC = 02003
pc = 2003 sp = 3000 ix = 254c iy = 0000
a = 00 f = <42,_Zx_x_N_> bc = 0000 de = 2511 hl = 2400

STATUS: Z80--Running in monitor Emulation trace halted
Window System Register Processor Breakpoints Memory Config Analysis
Go Break Reset I/O CMB Step
```

An individual step may execute multiple instructions. When this happens, the step display includes all those instructions.

To continue stepping through the program, you can select:

Processor Step Pc

After selecting the command above, you can change the previous step count. If you want to step the same number of times, you can press **Enter** to start the step. To repeat the previous command, you can press **<CTRL>r**.

Note



The default configuration assigns the command **Processor Step Pc 1** to function key **<F1>**. See the *PC Interface Reference* for more information on function key macros.

Specifying a Step Count

If you want to continue to step a number of times from the current program counter, select:

Processor Step Pc

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

```
Emulation
a = 00 f = <42,_Zx_x_N_> bc = 0000 de = 2511 hl = 2400
02003 CMD_RDR.S:CLEAR LD HL,CMD_INPUT
PC = 02006
pc = 2006 sp = 3000 ix = 254c iy = 0000
a = 00 f = <42,_Zx_x_N_> bc = 0000 de = 2511 hl = 2400

02006 - LD (HL),00
02008 DR.S:READ_INPUT LD A,(HL)
02009 - CP 00
0200b - JR Z,2008
02008 DR.S:READ_INPUT LD A,(HL)
PC = 02009
pc = 2009 sp = 3000 ix = 254c iy = 0000
a = 00 f = <42,_Zx_x_N_> bc = 0000 de = 2511 hl = 2400

STATUS: Z80--Running in monitor Emulation trace halted
Window System Register Processor Breakpoints Memory Config Analysis
Go Break Reset I/O CMB Step
```

When you specify step counts greater than one, all the instructions and the register contents after the last instruction are displayed.

Run the Program

To start the emulator executing the sample program, select:

Processor Go Pc

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

Break Into the Monitor

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

Processor Break

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

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

Making Coverage Measurements

You can measure how much of a range of emulation memory is accessed by your program by using the **Memory Report** command. To use this command, make sure that the emulator is executing in the monitor or reset, then reset the coverage memory. Select:

Processor Break

Memory Report Reset

All coverage memory bits are reset to a non-accessed state. Determine the amount of memory accessed by the program by selecting:

Memory Report Accessed

Enter the address range you want to check (for example, **INIT..PROCESS_COMM**). The percentage accessed should show 0. Because the emulator is not running the user program, no memory has been accessed. Start the emulation processor executing the example program (or one of yours) so that you can make a coverage measurement.

Processor Go Address

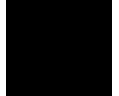
Enter the start address of your program (**INIT** for the sample program). Then, select:

Memory Report Accessed

Enter the address range. The percentage accessed is shown in the emulation window. To list the addresses not accessed during the measurement, select:

```
Memory Report Nonaccessed
```

Again, you should enter the address range of the block you want to check. The percentage of locations not accessed is displayed in the emulation window.



Using Software Breakpoints

The Z80 emulator replaces instructions with a LD B,B instruction sequence to implement a software breakpoint.

When a software breakpoint executes, the breakpoint sequence is replaced by the original opcode. A subsequent run or step command will execute from this address.

Defining a Software Breakpoint

To define a breakpoint at the address of the **COMMAND_A** label of the sample program, select:

```
Breakpoints Add
```

Enter the symbol **COMMAND_A**. After the breakpoint is added, the emulation window becomes active and shows that the breakpoint is set.

You can add multiple breakpoints in a single command by separating them with a semicolon. For example, you could type **CLEAR_LOOP;PROCESS_COMM;COMMAND_A** to set three breakpoints.

Run the program by selecting:

```
Processor Go Pc
```

The status line shows that the emulator is running the user program. Modify memory to enter the A command by selecting:

```
Modify Memory Bytes
```

Then enter **CMD_INPUT="A"**. The following messages result:

```
ALERT: Software breakpoint: 02026
STATUS: Z80--Running in monitor
```

To continue program execution, select:

```
Processor Go Pc
```



Displaying Software Breakpoints

To view the status of the breakpoint, select:

```
Breakpoints Display
```

The resulting display shows that the breakpoint was cleared.

Setting a Software Breakpoint

When a breakpoint is hit, it is disabled. To reenable the software breakpoint, you can select:

```
Breakpoints Set Single
```

Specify the address or symbol of the breakpoint to be enabled. As with the “Breakpoints Add” command, the breakpoint window becomes active and shows that the breakpoint is set.

Clearing a Software Breakpoint

If you want to clear a software breakpoint that is pending, you can select:

```
Breakpoints Clear Single
```

Specify the address or symbol of the breakpoint to be cleared.

Using the Analyzer

The Z80 emulation analyzer has trace signals that monitor internal emulation lines (address, data, and status lines). Optionally, you may have an additional 16 trace signals, which monitor external input lines. The analyzer collects data at each pulse of a clock signal, and saves the data (a trace state) if it meets a “storage qualification” condition.

Note



Emulators which do not have the optional external analyzer will not display the **I**nternal option shown in the examples. Enter the first part of the command to execute the example.

Resetting the Analysis Specification

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

```
Analysis Trace Reset Internal
```

Specifying a Simple Trigger

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

```
Analysis Trace Modify Internal
```

The analyzer trigger specification is shown. Use the arrow keys to move to the "Trigger on" field. Type in "a" and press **Enter**.

Now you enter the analyzer pattern specification form. Use the arrow keys to move the cursor to the field under the heading "addr" next to the row marked "a=." Type in the address of the command input byte, using the global symbol **CMD_INPUT**, and press **Enter**.

The "Data" field is now highlighted. Type in 42 and press **Enter**. 42H is the value of the "B" command. (Or, you can enter "B".)

Now the "Status" field is highlighted. Use the **Tab** key to view the status qualifiers that may be entered.

The status qualifiers are defined as follows.

Qualifier	Status Bits (46..36)	Description
busack	0xxxx1111y	Bus acknowledge cycle.
dataread	0xxxx0100y	Memory data read cycle.
datawrite	0xxxx0101y	Memory data write cycle.
grd	11xx0x0xy	Guarded memory access.
illegal	0xxxx0010y	Illegal opcode.
im0	0xxxx1000y	Maskable interrupt mode 0.
im1	0xxxx1001y	Maskable interrupt mode 1.
im2	0xxxx1010y	Maskable interrupt mode 2.
input	0xxxx0110y	I/O read cycle.
monitor	0xxxxxxxxy	Emulation monitor cycle.
nmi	0xxxx1011y	Nonmaskable interrupt request cycle.
opcode	0xxxx0000y	Opcode fetch cycle.
operand	0xxxx0001y	Operand fetch cycle.
output	0xxxx0111y	I/O write cycle.
refresh	0xxxx1100y	Memory refresh cycle.
user	1xxxxxxxxy	Operating in memory mapped to target.
wrrom	1x1x0101y	Write to ROM.

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

Figure 2-1 shows the resulting analysis pattern specification. To save the new specification, use the **End Enter** key sequence to exit from the form.

Figure 2-2 shows the analysis trigger specification. To save the trigger specification, press **End**, then **Enter**.

2-24 Use the Emulator

Internal State Trace Specification

Range (r)	Label	addr	=	Set 1	thru	xbits
Pat		addr		data	stat	
a =		CMD_INPUT		42	dataread	
b =						
c =						
d =						
Set 2						
e =						
f =						
g =						
h =						
arm						

Expression

Expressions have the form: <set1> and/or <set2>. Where set1 consists of <a, b, c, d, r, !r> and set2 consists of <e, f, g, h, arm>. Patterns within a set can be joined with !(or) or ~(nor), but not both. Example: !r ~ a or e ! f ! g ! h
 Pattern Expression: a

STATUS: Z80--Running user program Emulation trace halted

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

Figure 2-1. Analyzer Pattern Specification

Internal State Trace Specification

1 While storing any state
 Trigger on a 1 times

2 Store any state

Branches off Count time Prestore off Trigger position start of 512

←↑↓ : Interfield movement Ctrl ↔ : Field editing TAB : Scroll choices

STATUS: Z80--Running user program Emulation trace halted

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

Figure 2-2. Analyzer Trigger Specification

Starting the Trace

To start the trace, select:

```
Analysis Begin Internal
```

Start the processor running with the command:

```
Processor Go Pc
```

A message on the status line shows that the trace is running. You do not expect the trigger to be found because no commands were entered. Modify the command input byte to “B” by selecting:

```
Memory Modify Bytes
```

Enter `CMD_INPUT=42` or `CMD_INPUT="B"`. The status line now shows that the trace is complete.

Displaying the Trace

To display the trace, select:

```
Analysis Display Internal
```

Note



If you choose to dump a complete trace into the trace buffer, it will take a minute or so to display the trace.

You now have two fields in which to specify the states to display. Use the right arrow key to move the cursor to the “Ending state to display” field. Type 260 into the ending state field, press **Enter**, and use `<CTRL>z` to zoom the analysis window. Press the **Home** key to move to the top of the trace list.

When symbol transfer is supported, the PC Interface will show the states available for display and the default disassembly mode. By specifying the disassembly mode, you can display addresses, symbols, or both addresses and symbols in the trace list. The default is “Both,” indicating that addresses and symbols will be displayed. To display only symbols, select “Symbols” as the disassembly mode. To display only addresses, select “Addresses” as the disassembly mode. Set the mode to **Both** for this example.

The resulting trace is similar to the trace shown in the following display.

Line	addr,H	Z80 Mnemonic,H	xbits,H	count,R	seq
0	NPUT	42 data read	0000	---	+
1	2009	CP 00	0000	0.320 uS	.
2	200a	00 operand	0000	0.560 uS	.
3	200b	JR Z,CMD_RDR.S:READ_INPUT	0000	0.320 uS	.
4	200c	fb operand	0000	0.560 uS	.
5	_OLD	PUSH AF	0000	0.320 uS	.
6	2fff	42 data write	0000	0.680 uS	.
7	2ffe	02 data write	0000	0.400 uS	.
8	200e	LD DE,OUTPUT_BUFFER	0000	0.280 uS	.
9	200f	00 operand	0000	0.560 uS	.
10	2010	25 operand	0000	0.400 uS	.
11	2011	LD BC,0020	0000	0.320 uS	.
12	2012	20 operand	0000	0.560 uS	.
13	2013	00 operand	0000	0.360 uS	.
14	2014	LD A,00	0000	0.320 uS	.
15	2015	00 operand	0000	0.560 uS	.
16	LOOP	LD (DE),A	0000	0.320 uS	.

STATUS: Z80--Running user program Emulation trace complete
 Window System Register Processor Breakpoints Memory Config Analysis
 Begin Halt CMB System Format Trace Display

Line 0 in the preceding trace list shows the state that triggered the analyzer. The trigger state is always on line 0. The other states show the exit from the **READ_INPUT** loop, and the **CLEAR_OLD** instructions.

Press the **End** key, then the **PgUp** key to see more lines of the trace.

The following trace list shows the jump to **OUTPUT** and the beginning of the LDIR instruction.

Analysis						
225	2034	JR	CMD_RDR.S:OUTPUT	0000	0.320 uS	.
226	2035		06 operand	0000	0.560 uS	.
227	TPUT	LD	DE,OUTPUT_BUFFER	0000	0.920 uS	.
228	203d		00 operand	0000	0.600 uS	.
229	203e		25 operand	0000	0.360 uS	.
230	203f	LDIR		0000	0.320 uS	.
231	2040		b0 operand	0000	0.480 uS	.
232	GE_B		54 data read	0000	0.560 uS	.
233	FFER		54 data write	0000	0.400 uS	.
234	203f	LDIR		0000	1.160 uS	.
235	2040		b0 operand	0000	0.520 uS	.
236	2612		68 data read	0000	0.560 uS	.
237	2501		68 data write	0000	0.360 uS	.
238	203f	LDIR		0000	1.200 uS	.
239	2040		b0 operand	0000	0.480 uS	.
240	2613		69 data read	0000	0.600 uS	.
241	2502		69 data write	0000	0.360 uS	.
242	203f	LDIR		0000	1.200 uS	.
243	2040		b0 operand	0000	0.480 uS	.

STATUS: Z80--Running user program Emulation trace complete
 Window System Register Processor Breakpoints Memory Config Analysis
 Begin Halt CMB System Format Trace Display

Changing the Trace Format

You can modify the trace list format to suit your needs. You can:

- Widen the address column to show longer symbol names.
- Change the port base; to octal, for example.
- Change the count from relative to absolute.

The default trace display format for the Z80 emulator includes:

- Trace line number (which is always displayed)
- Hexadecimal address
- Z80 mnemonic
- External Analyzer inputs (if installed)
- Relative count

You can make the trace format more useful for this example by changing the address field width, adding a data character, and removing the external analyzer bits.

Analysis						
Line	addr,H	Z80 Mnemonic,H	data,A	count,R	seq	
0	CMD_INPUT	42 data read	B	---		+
1	2009	CP 00	.	0.320 uS	.	.
2	200a	00 operand	.	0.560 uS	.	.
3	200b	JR Z,CMD_RDR.S:READ_INPUT	(0.280 uS	.	.
4	200c	fb operand	.	0.560 uS	.	.
5	.S:CLEAR_OLD	PUSH AF	.	0.320 uS	.	.
6	2fff	42 data write	B	0.680 uS	.	.
7	2ffe	02 data write	.	0.400 uS	.	.
8	200e	LD DE,OUTPUT_BUFFER	.	0.320 uS	.	.
9	200f	00 operand	.	0.560 uS	.	.
10	2010	25 operand	%	0.360 uS	.	.
11	2011	LD BC,0020	.	0.320 uS	.	.
12	2012	20 operand	.	0.560 uS	.	.
13	2013	00 operand	.	0.360 uS	.	.
14	2014	LD A,00	>	0.320 uS	.	.
15	2015	00 operand	.	0.560 uS	.	.
16	S:CLEAR_LOOP	LD (DE),A	.	0.320 uS	.	.

STATUS: Z80--Running user program Emulation trace complete
Window System Register Processor Breakpoints Memory Config Analysis
Begin Halt CMB System Format Trace Display

Note



Notice that symbols are displayed in the trace list. If your emulator does not have symbol-handling capability, the symbols won't be displayed.

To see the bus activity for moving the message to the destination location, press the **End** key, then the **Page Up** key. You'll see the display that follows.

In-Circuit Emulation

In-circuit emulation is the process of using the emulator while the emulator probe is installed in the microprocessor socket of your target system. This lets you use the emulator to debug your target system hardware and software.

Installing the Emulator Probe

Precautions

If you install the emulator probe on a densely populated circuit board, there may not be enough room for the plastic shoulders of the probe connector. If this occurs, a female-to-male pin extender may be stacked onto the existing pin extender. But, adding length to the connector may degrade the signal quality in some target system designs.

Caution



Possible damage to the emulator probe. The emulation probe contains devices that are susceptible to damage by static discharge. You should take precautions before handling the probe, to avoid damaging the internal components of the probe with static electricity.

Caution



Possible damage to the emulator. Make sure both target system and emulator power are OFF before installing the emulator probe into the target system.

Caution



The emulator probe will be damaged if incorrectly installed. Make sure to align pin 1 of the probe connector with pin 1 of the socket.

Installation

To install the emulator probe:

1. Power down the target system and emulator.
2. Remove any tape, foam, or plastic from the end of the emulator probe.
3. Examine all pins on the protective socket connected to the emulation probe to verify that they are straight.
4. If any of the pins on the probe are bent, carefully straighten them. If a pin breaks on the socket, replace the socket.
5. Remove the Z80 microprocessor from the target system socket. Store the microprocessor in a protected environment (such as antistatic foam).
6. Locate pin 1 on the emulator probe. It is at the tip of the probe.
7. Locate pin 1 on your target system microprocessor socket.
8. Carefully match the pins on the emulator probe with the target system microprocessor socket.
9. Depending on what type of socket is in your target system, either press down on the emulator probe, or push the lever on the socket down. You must make a stable connection between the emulator probe and target system Z80 microprocessor socket.
10. Power up the emulator.
11. Power up the target system.

Real-Time versus Non-Real-Time Operation

Some target systems require real-time operation. Target systems that process interrupts and/or depend on a real-time clock for operation may not function correctly when you use certain emulation features.

If real-time performance of your target system is important, you must use care in using the emulation commands. Certain commands force monitor intervention or steal bus cycles from the processor to perform emulation functions.

Any emulator command that requires you to break to the monitor or forces a monitor break will interrupt real-time execution. These include register display and modify operations and display or modification of target system memory. Any operation that resets the emulator from the emulation controller also interferes with real-time operation.



Emulation Memory and Target System Memory

The use of emulation memory and/or target system memory can significantly affect the operation of the emulator. Ideally, emulation of a microprocessor should be done with as much of the final target system hardware as possible. The Z80 emulator provides for emulation memory to replace target system memory at the beginning of project development.

The final target system memory may not have the same specifications as emulation memory. Therefore, you should use the external target system clock to assure synchronous operation between the emulator and your target system.

Because of the inherent delays in the path between the emulation processor and the target system microprocessor socket, the emulator timing may not match the microprocessor timing specifications. Make sure that your target system specifications are within the minimum and maximum timing specification limits at the end of the emulator probe user cable. See the *Z80 Terminal Interface User's Guide* for information on emulator timing differences.

High-Speed CMOS Target System Interface

The HP 64753 Z80 emulator is designed for both CMOS capability, and operation at clock speeds up to 10 MHz. To meet these requirements, high speed CMOS buffer circuitry drives the address, data, and status signals on the emulator cable to the target system.

These CMOS buffers have extremely fast rise and fall times. Some Z80 target systems may exhibit erratic operation with the Z80 emulator installed, such as target systems which do not have power and ground layers on their printed circuit boards. This may occur with a prototype that is wire-wrapped.

To minimize the effects of this problem, Hewlett-Packard suggests that you add the circuit shown in figure 2-4. Figure 2-5 shows one possible implementation of the circuit.

Consult your local Hewlett-Packard Sales and Service Office for more information.

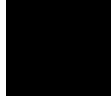
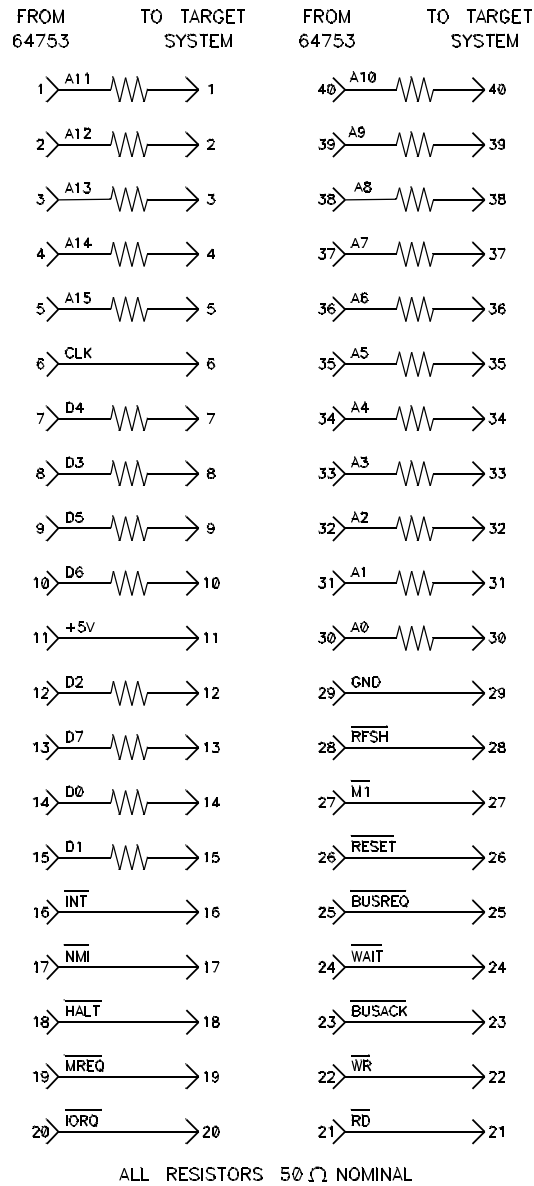


Figure 2-4. High-Speed CMOS Interface Circuit

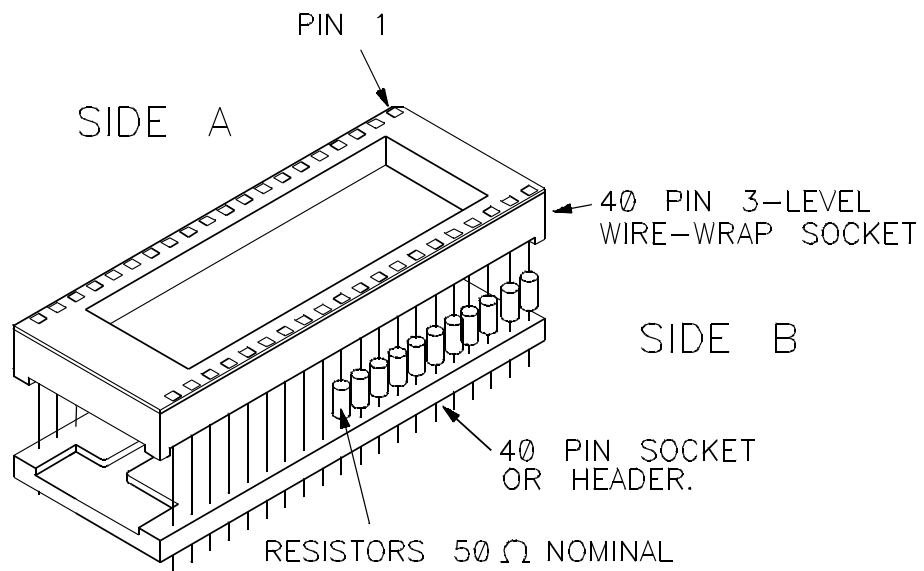
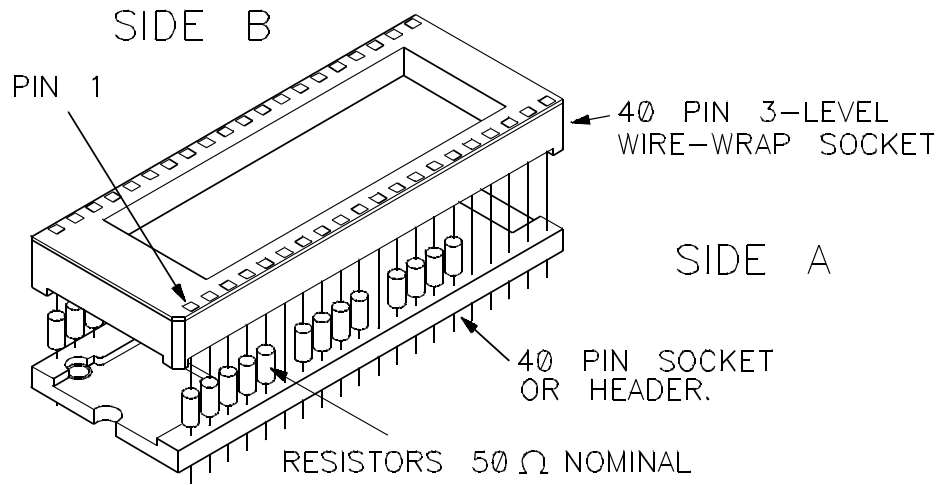


Figure 2-5. CMOS Interface Circuit Implementation

Run from Reset

You can cause the emulation processor to execute when a target system reset occurs. When you enter a run from reset (**Processor Go Reset**) command, the emulation processor will wait for a reset condition to occur in the target system before it continues running.

The emulator will recognize a **Processor Go Reset** command either while it is running a user program, running in the monitor, or is in the reset state.

Emulation Monitor Description

The HP 64753 Z80 emulator uses a background monitor. The monitor program is the interface between the emulation system controller and the target system. The emulation system controller contains a microprocessor that accepts and executes emulation, system, and analysis commands. The emulation processor executes the background monitor program.

The monitor program allows emulation commands to access target system resources. Access to the target system can be done only through the emulation processor. For example, if you enter a command to modify target system memory, the monitor program will execute instructions that write the new value into target system memory.

When the emulation system controller recognizes that an emulation command needs to access target system resources, it writes a command code to the communications area. Then it breaks into the monitor. The monitor reads the command code (and any associated parameters), then executes the appropriate Z80 instructions to access target system resources.

Background Operation

A background monitor executes independently from a user program. It has its own memory space. When executing in background, the emulator appears suspended to the target system (unless you request that monitor cycles are driven to the target; see chapter 3).



Modifying Operation Of The Monitor Program

When the emulator breaks from the user program it begins running the monitor program. The monitor allows access to internal Z80 registers and provides a way to access memory and input/output ports in the target system.

When the emulator stops running the user program, any support that is provided by the user program to the target system also will stop, including support of interrupts. This may be a problem, particularly with target systems that have real-time requirements.

Make Bus Cycles Visible

There are several ways that you can alter the operation of the monitor program to satisfy some specific requirements of your target system. The first step is to make the monitor program bus cycles visible to the target system by setting the **Enable monitor cycles to target?** configuration item to **y**.

Define Addresses. Your target system will require this if it has dynamic memories that must be refreshed by the Z80. You can define the range of addresses driven to the target system by setting the **Monitor block** configuration item.

For example, if you type in **7**, the probe will output addresses in the range 7000H through 7FFFH while running in the monitor.

Reduce Monitor Program Access Time

You can reduce the amount of time that the monitor program runs when a temporary break occurs. A temporary break occurs when the emulator breaks to the monitor to display registers or to display target memory. An automatic return to the user program follows the break. By setting the **Enable quick-break algorithm?** configuration item to **y**, the amount of time spent in the monitor for a temporary break can be reduced from about 6 milliseconds to about 200 microseconds. This depends on processor clock speed.

Even though interrupts are not serviced while the emulator is running in the monitor, it may be possible to return the emulator to the user program in time to service interrupts satisfactorily.

Note



This mode of breaking affects CMB operation. Other emulators on the CMB will not break to the monitor when this emulator breaks in the quick mode.

Tailoring The Monitor For Target System Interaction

Some target systems require constant interaction, and cannot tolerate the suspended state imposed by the monitor. The Z80 emulator allows you to add system-specific subroutines to the monitor to handle target system interaction during monitor execution. For example, a target system might have a “watchdog timer” that will reset the target system if it is not written to periodically. Here, you could modify the monitor program to write to the timer while it is running, in the same way that the user program writes to the timer. Or, you could turn off the timer when a break to the monitor occurs, and turn it on again when the emulator returns to the user program. Both requirements can be satisfied by adding user code to the monitor program.

Loading the User Monitor Code. You supply the user code in four possible Z80 assembly language subroutines.

- Subroutine #1: Reset Entry To Monitor

The first subroutine will be called once when the monitor program is entered from reset. This will occur when a reset command (**rst**) is followed by a break (**b**), run (**r**), or step (**s**) command, or with a reset to monitor command (**rst -m**).

- Subroutine #2: Break Entry To Monitor

The second subroutine will be called once when the emulator breaks from the user program to the monitor.

- Subroutine #3: Monitor Loop

The third subroutine will be called once for each time the monitor program cycles through its main loop.

■ Subroutine #4: Exit From Monitor

The last subroutine is called once when the emulator exits the monitor and returns to the user program.

Note



You don't need to provide all four subroutines. Any subroutine that is not included in the loaded code will be defaulted to a "return from subroutine" (RET) when the load is performed.

You load the user code using the **Memory Load** command. Use the **Tab** and **Shift-Tab** keys to change the "Target memory type for memory load" to **monitor**. The following display shows an example.

```
Memory Load Configuration

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

File name
mon_code.l

<↑↓> :Interfield movement  Ctrl <=> :Field editing  TAB :Scroll choices
STATUS: Z80--Running user program      Emulation trace complete
Enter the name of an HP64000 linker symbol file (ex. test.L).
```

Restrictions

Code that you add to the monitor is restricted as follows:

User Code Separate From Monitor Code. Code that you add to the monitor is completely separate from your user program. The user monitor code does not reside in the user address space, nor can it call or jump to any part of your target system program.

Location of the Subroutines. The four subroutines must begin at the following addresses:

0300H - reset entry to monitor subroutine

0400H - break entry to monitor subroutine

0500H - monitor loop subroutine

0600H - exit from monitor subroutine

Note



Each subroutine must end with a return from subroutine instruction (RET). The entire user monitor code must reside within the address range 0300H through 06FFH. The user monitor code must not jump to or access any monitor locations outside this range.

Note



The addresses of the subroutines are not related to the setting of the **Monitor block** configuration item.

Communication With Target System I/O Ports. The user monitor code can communicate with target system memory and I/O (input/output) ports. The instructions shown in table 2-3 must be used. No other instructions will access the target system.

Table 2-3. Instructions Used With a Target System

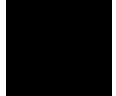
Instruction	Function
LD HL,(nnnn)	Loads HL with target memory locations nnnn and nnnn+1.
LD (nnnn),HL	Writes contents of HL to target memory locations nnnn and nnnn+1.
LD r,(HL)	Loads register with target memory address pointed to by HL. "r" indicates register A, B, C, D, E, H, or L.
LD (HL),r	Writes register content to target memory pointed to by HL.
IN r,(C)	Loads register with input port value pointed to by the C register.
OUT (C),r	Writes register content to I/O port pointed to by C.

You Cannot Use Some Instructions. Do not use the following instructions anywhere in the user monitor code. They are reserved for control of the Z80 emulator. *The emulator will operate erratically if you use any of these instructions:*

JP nnnn
RST n
LD A,(BC)
LD D,D
LDIR
LDDR

Registers Used by the User Monitor Code. The user monitor code may use only the following Z80 registers: A, flags, BC, DE, and HL.

The monitor does not maintain the contents of these registers between calls to the user routines. Any data storage required must share monitor memory addresses in the range 0300H through 06FFH with the user monitor code. Instructions used to access data stored in monitor memory must not use the instructions for target memory communication listed above.



A Stack is Provided. A stack is provided so that you may include subroutines in the user monitor code. The stack is used only for calling the user monitor subroutines, and for their operation. The stack size is limited to 64 bytes. The stack is always automatically initialized before a call is made to a user monitor subroutine.

No Access To Emulation Memory. The user monitor code cannot access emulation memory.

Caution



If the user monitor subroutines do not adhere to the restrictions listed above, the emulator may operate incorrectly.

Creating/Loading The User Monitor Subroutines

You can develop the user monitor subroutines with the same tools that are used for the user application program. Any or all of the subroutines can be included in the same file. You must **ORG** each subroutine at its specified address (refer to the appropriate *Assembler Manual* for details about the **ORG** command).

The subroutines are then loaded into the emulator with the monitor memory type option in the Memory Load form.

Note



The load will fail if the addresses specified in the file are not in the range 0300H through 06FFH.

Observe Monitor Operation

After the user monitor code has been loaded, the emulation analyzer can be used to monitor its operation. You can enable tracing of background monitor cycles by changing the qualify states field in the Analysis Format form to **both**.

Refer to the *Analyzer PC Interface User's Guide* for details.

Note



If the emulator is reinitialized, the user monitor code will be lost, and must then be reloaded.

Configure the Emulator

Topics Covered

Your Z80 emulator can be used in all stages of target system development. For instance, you can run the emulator out-of-circuit when developing target system software, or you can use the emulator in-circuit when integrating software with target system hardware. Emulation memory can be used for or with target system memory. You can use the emulator's internal clock or the target system clock. You can execute target programs in real-time. Or, divert emulator execution into the monitor when commands request access to target system resources (target system memory or I/O, or register contents).

The emulator is a versatile instrument and may be configured to suit your needs at any stage of the development process. This chapter shows you how to configure the HP 64753 Z80 emulator. Topics include:

- Z80 Configuration Items
- Mapping Memory
- Break Conditions
- Saving the Configuration
- Loading a Configuration
- Where to Find More Information

Prerequisites

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

Access Emulator Configuration Options

To access the Z80 emulator configuration options, select:

Config General

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

Note



You can use the System Terminal window to modify the emulator configuration. But, if you do this, some PC Interface features may no longer work properly. You should only modify the emulator configuration by using the options presented in the PC Interface.

You change items in the configuration screen by using the arrow keys to move to the selected item. Type in a value, or press the **Tab** and **Shift-Tab** keys to cycle through the various options. Press the **End** and **Enter** keys to exit from the configuration form and save your changes. Press the **Esc** key to exit from the configuration form and discard your changes.

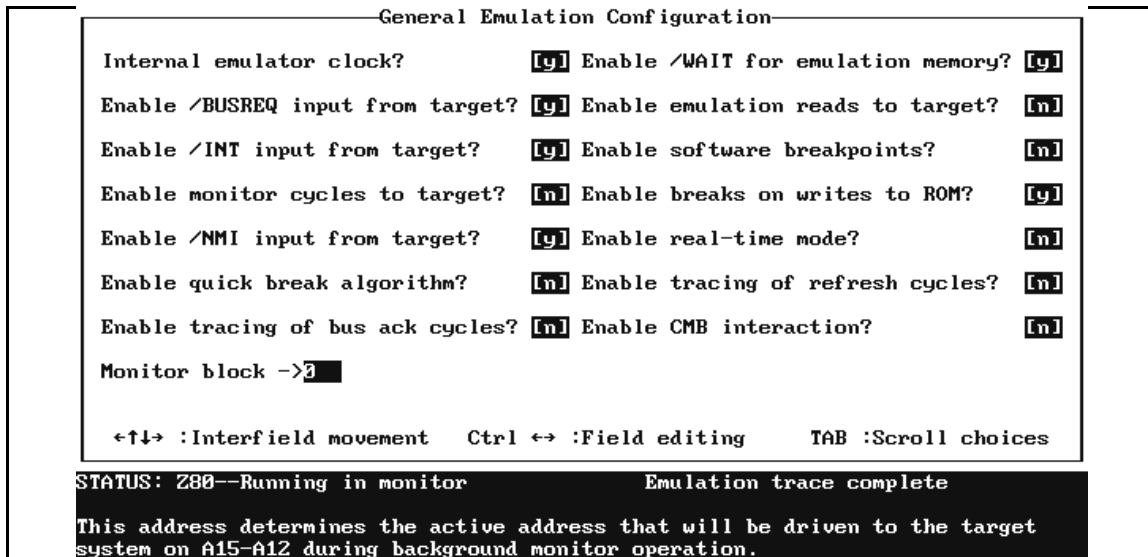


Figure 3-1. Emulator Configuration (Default)

Internal emulator clock?

- | | |
|---|---|
| y | Selects the 8 MHz clock source in the Z80 emulator. |
| n | Selects the clock source in the target system. External clock speeds up to a maximum of 10 MHz are supported. |
- When you use the internal clock, code execution time is relative to the internal clock speed. Select the internal clock specification when you perform out-of-circuit emulation (for example, while debugging software without a target system).

**Enable /WAIT input
from target?**

y

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.

n

Select “no” if you don’t want wait states inserted during emulation memory accesses.

**Enable /BUSREQ
input from target?**

y

When you select “yes” the emulator will respond to a BUSREQ input from the target system.

n

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

**Enable data output to
target?**

n

When you select “no” the emulator will drive the data bus only during memory write and output cycles.

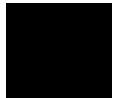
y

When you select “yes” the emulator will drive the data bus to the target system during all read cycles from emulation memory. The value driven is that read from emulation memory. Some target systems may need this so that peripheral devices can decode a RETI instruction that is in emulation memory. This may cause bus contention in the target system.

Enable /INT input from target?

y

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.



n

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

Enable software breakpoints?

n

The emulator does not respond to software breakpoints.

y

The emulator recognizes software breakpoints.

Note



The PC Interface automatically sets this configuration item to **y** when you use the **Breakpoints** command to add a breakpoint.

Enable monitor cycles to target?


n

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

y

When you select “yes,” while the emulator is executing in the monitor, the target system will recognize that the monitor program is running.

The **Monitor block** configuration item determines the address shown on lines A15-A12 when cycles are driven.



Enable breaks on writes to ROM?

- y 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.
- n Answering “no” keeps the emulator from breaking when a write to ROM occurs.

Enable /NMI input from target?

- y 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.
- n Select “no” if you want the emulation processor to ignore non-maskable interrupt requests by the target system.

Enable real-time mode?

- n 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.

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 use the **Processor Break** command to stop program execution and begin running in background.

Enable quick-break algorithm?

n

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.

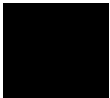
y

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 tracing of refresh cycles?

n

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).



Enable tracing of bus ack cycles?

y When you select “yes,” memory refresh cycles (both foreground and background) will appear in the analysis trace unless excluded by the trace specification.

n 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).

y When you select “yes,” bus acknowledge cycles (both foreground and background) will appear in the analysis trace unless excluded by the trace specification.

Monitor block

You can select the value that will be driven to the target system on address lines A15..A12 during background monitor operation. This value can be in the range 0 to 0FH. The default value is 0.

The entire 64-kilobyte address range is available for the user program. You should select a memory block where memory read operations will not cause undesirable interaction with the target system.

This configuration item has effect only when the **Enable monitor cycles to target?** configuration item is set to **y**.

Mapping Memory

The memory map allows you to define whether memory in the target system or emulator will respond to specific ranges of addresses. You can define map terms for program memory and data memory.

Note



When you modify the memory map, any existing software breakpoints are not cleared. Therefore you may want to delete all breakpoints before changing the memory map. To delete all breakpoints, use the command **Breakpoints Clear All**.

When mapping memory, you can define an address range as RAM or ROM. The emulator will break to the monitor if a write to ROM occurs, and a break on write to ROM is enabled (with the **Breaks on Write to ROM** configuration item).

You also can define an address range as guarded (grd). This is useful for detecting a read or write to an address where memory is not used, or is nonexistent. A break to the monitor will occur whenever guarded memory is accessed.

Which Memory Locations Should Be Mapped?

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

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

Display and Modify the Memory Map

You can display and modify the Z80 emulator memory map using the **Config Map Modify** command. Each memory map term will be a multiple of 256-byte blocks. If you specify a memory range that contains only part of a 256-byte block, the emulator will map a

complete 256-byte block that includes that range. You can specify a total of 16 memory map terms.

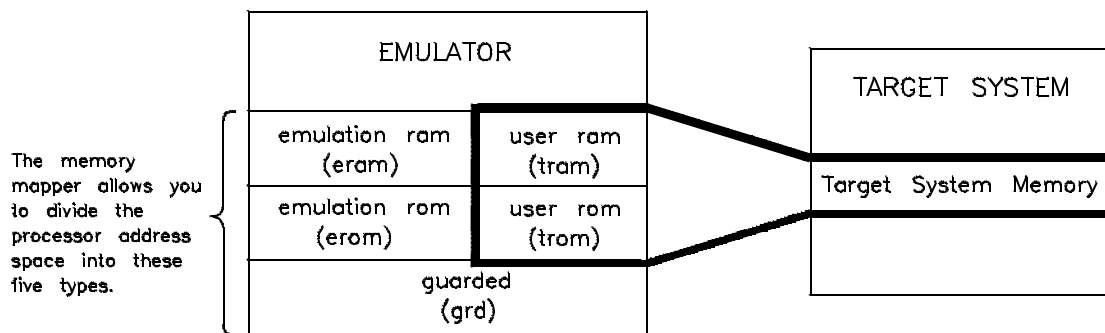
Specifying Memory Type

For each memory mapper term, you can specify one of these valid memory types:

- eram (emulation RAM)
- erom (emulation ROM)
- tram (target RAM)
- trom (target ROM)
- grd (guarded memory)

Z80 addresses not covered by your map term(s) are mapped to “other,” which is defined as emulation RAM (eram) at powerup.

You can map all memory to the target system, allowing you to do all your work in target system memory, such as copying programs from target system ROM to target system RAM to make changes and debug. You also can copy programs from target system ROM or RAM to emulation RAM so that they may be analyzed or altered.



Notes:

1. You can specify a maximum of 16 address ranges.
2. Unspecified addresses are mapped as grd if the default is set to guarded.
3. When emulation memory is mapped as tram or trom, target system memory is accessed during memory cycles.

Figure 3-2. Memory Map Types

3-10 Configure the Emulator

Modifying the Memory Map

To modify the memory map, enter:

`Config Map Modify`

You will see the entire memory map, resembling figure 3-3.

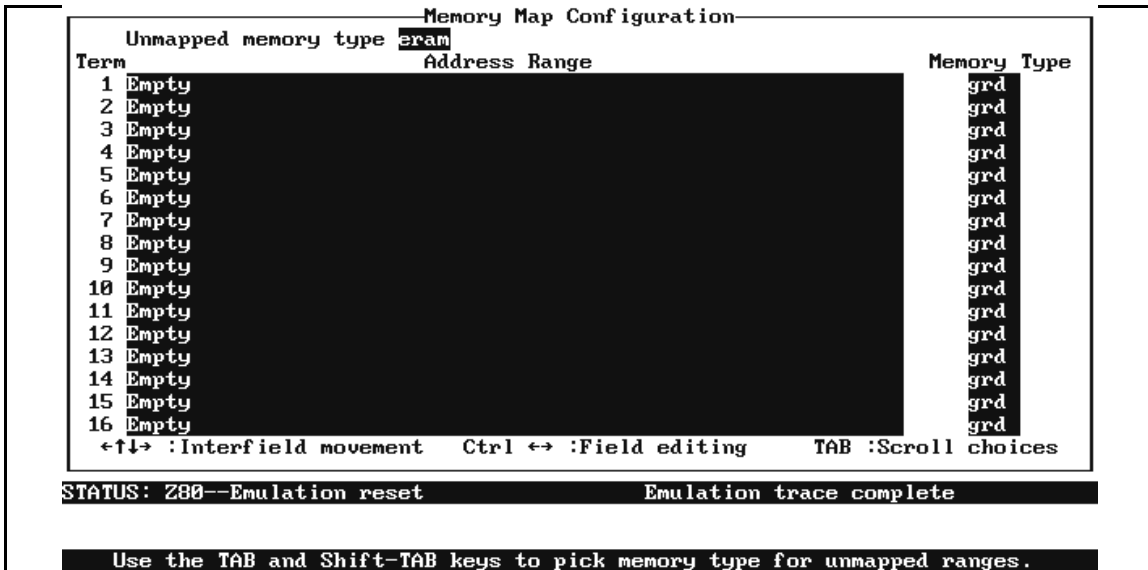


Figure 3-3. Default Memory Map

To define ranges of memory, press **Tab** first to choose a memory type for unmapped address ranges. The default is target RAM (tram).

Press the **down arrow** to move the cursor to any empty address range field. You also can use the **up arrow** to move the cursor up through the terms. Place the cursor at the Term 2 entry.

Type in a memory range, for example, **8000..800f**. The emulator will allocate a block of memory 256 bytes long (8000..80ff). You can't create mapper terms that cross a 256-byte boundary (such as 80f0..80ef). Press the **right arrow** or **Enter** to move to the Memory Type field.

Press **Tab** or **Shift Tab** to choose the memory type you want, or type in the memory type. Press **Enter**.

Note



You can put an expression in the address range field. For example, the expression **0.start+100/2-20** is a valid address range.

If you define multiple terms with empty terms between them, the intervening empty terms are removed when you save the map specification. For example, if you defined two terms in the memory map like the following diagram:

```
Memory Map Configuration
Unmapped memory type eram
Term  Address Range  Memory Type
1  00000..000ff  eram
2  Empty          grd
3  09000..090ff  eram
4  Empty          grd
5  Empty          grd
6  Empty          grd
7  Empty          grd
8  Empty          grd
9  Empty          grd
10 Empty          grd
11 Empty          grd
12 Empty          grd
13 Empty          grd
14 Empty          grd
15 Empty          grd
16 Empty          grd
←↑↓→ :Interfield movement  Ctrl ↔ :Field editing  TAB :Scroll choices
STATUS: Z80--Emulation reset  Emulation trace complete
Use the TAB and Shift-TAB keys to pick memory type for mapped range.
```

3-12 Configure the Emulator

When you save and exit the map, it will appear as:

```
Memory Map Configuration
Unmapped memory type eram
Term      Address Range      Memory Type
1 08000 .080ff      eram
2 09000 .090ff      eram
3 Empty
4 Empty
5 Empty
6 Empty
7 Empty
8 Empty
9 Empty
10 Empty
11 Empty
12 Empty
13 Empty
14 Empty
15 Empty
16 Empty
←↑↓→ :Interfield movement  Ctrl ↔ :Field editing  TAB :Scroll choices
STATUS: Z80--Emulation reset      Emulation trace complete
Use the TAB and Shift-TAB keys to pick memory type for unmapped ranges.
```

Notice that the original empty term between the two defined terms was removed. This is true for any terms that you define that are separated by empty terms.

Exit the Memory Map

When you want to save changes to the memory map, then exit, you can press **End** and then **Enter**. When the cursor is in the last field, just press **Enter** to exit the memory map and activate your definitions.

Note



To exit the memory mapper without saving changes, just press **Esc**. To delete a map term, just clear the address field by pressing the space bar.

Note



The memory mapper reassigns blocks of emulation memory after the insertion or deletion of mapper terms. For example, if you:

- modified the contents of term 1
- added a new term
- deleted term 1
- displayed locations in term 1

you would notice that the contents of those locations differ.

Reset the Memory Map

To reset the memory map, enter:

```
Config Map Reset
```

All the memory map parameters will now be set to the default values. By doing this, all unmapped memory is defined as emulation RAM, all address terms are empty, and all memory types are guarded.

Hardware Breakpoints

The analyzer may generate a break request to the emulation processor. To specify a break condition to execute on receiving an analyzer trigger, you use the **Config Trigger** form to modify the cross trigger configuration.

After you specify the trigger signal to be driven, and enable the trigger break condition, you can set up and execute the trace command. Then run the program. When the trigger condition is found, the emulator will break into the monitor. If the trigger condition is not found, the trigger will not occur, and the emulator will not break.

Modify the Cross Trigger Configuration

By modifying the cross trigger configuration, you select which devices drive and receive the trig1 and trig2 internal trigger signals. You may want to use either or both signals to arm the emulation analyzer.

To modify the cross trigger configuration, enter:

Config Trigger

The cross trigger configuration form is displayed (see figure 3-4).

When you position the cursor to any one of the fields, the last three lines on the bottom of the screen explain how that device handles the trig1 and trig2 lines.

The screenshot shows a terminal window titled "Cross Trigger Configuration". It is divided into two columns for "TRIG1" and "TRIG2". Each column has four rows: "BNC", "CMB", "Emulator", and "Analyzer". The "BNC" and "CMB" fields are set to "ignore". The "Emulator" field for TRIG1 is set to "<<-----" and for TRIG2 is set to "ignore". The "Analyzer" field for TRIG1 is set to "----->>" and for TRIG2 is set to "ignore". Below the configuration fields, there are three lines of status information: "<f1> : Interfield movement", "Ctrl <f2> : Field editing", and "TAB : Scroll choices". At the bottom, there is a status bar with "STATUS: Z80--Emulation reset" and "Emulation trace complete". A final line of text reads: "The internal analyzer may drive (----->>) , receive (<<-----) or ignore the TRIG1 and TRIG2 signals."

Figure 3-4. Cross Trigger Configuration

Use the **arrows** to position the cursor at any field. Press the **Tab** or **Shift Tab** keys to make a selection in each field.

Trigger Signal Interaction

Interaction with the trigger configuration by any or all devices can be ignored. This is the default.

The BNC trigger can drive, receive, or both drive and receive the trig1 and trig2 signals to CMB trigger or to the emulator.

The CMB trigger can drive, receive, or both drive and receive trig1 and trig2 to BNC trigger or to the emulator.

The analyzer can drive or receive the trig1 and trig2 signals to BNC trigger, CMB trigger, or to the emulator.

The emulator can receive the trig1 and trig2 signals from any of the other devices. The emulator can break into the monitor upon receiving either of these internal analyzer trigger signals. You must execute the correct commands to make this happen. Refer to the *CMB User's Guide* and the *Analyzer PC Interface User's Guide* for more information about the trig1 and trig2 signals and breaking the emulator on receiving these signals.

Exit the Cross Trigger Configuration

To activate your cross trigger definitions and exit the trigger configuration, you can press **Enter** when the cursor is in the last field. Otherwise you can press **End**, then **Enter** when the cursor is positioned elsewhere on the screen, and will exit with the changes activated.

Note



You can exit the cross trigger configuration without saving changes by pressing **Esc**.

Storing an Emulator Configuration

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

- General emulator configuration items.
- Memory map.

- Break conditions.
- Trigger configuration.
- Window specifications.
- Function key macros.

To store the current emulator configuration, select:

Config **S**tore

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

Loading an Emulator Configuration

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

Config **L**oad

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

Where to Find More Information

Due to the architecture of the HP 64700-Series Emulators, there are many items that affect how the emulator interacts with your system, controller, and other measuring instruments. If you need more information, refer to the following:

Analyzer PC Interface User's Guide

This manual describes how to use the analyzer in the PC Interface.

CMB User's Guide

This manual describes how to use the Coordinated Measurement Bus.

PC Interface Reference

This manual contains detailed descriptions for HP 64700-Series PC Interface commands. There is also more information on emulation configuration files.

Solve Problems

This chapter shows you how to solve some problems you may encounter while using the Z80 Emulator PC Interface.

Problems with Software Breakpoints

The Z80 emulator uses the LD B,B instruction to implement software breakpoints. If you use this instruction sequence in your programs when software breakpoints are enabled, the emulator will interpret the instruction as a software breakpoint.

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

Because software breakpoints are implemented by replacing opcodes with another instruction sequence, you cannot define software breakpoints in target ROM.

Do not add, set, remove, or disable software breakpoints while the emulator is running user code. If you enter any of these commands while the emulator is running user code in the area where the breakpoint is being modified, program execution may be unreliable.

Remove software breakpoints before altering the memory map or changing the monitor type. If you do not remove the breakpoints, the breakpoint instruction sequence will be left at unknown locations.

Insufficient Memory

In the PC Interface

Certain system configurations may consume enough DOS memory to cause problems when you use the PC Interface.

General Memory Problems

If you frequently encounter out of memory errors while using the PC Interface, you may have network utilities, device drivers, or TSRs (Terminate and Stay Resident) programs that are consuming memory. Check your system configuration and remove any programs that aren't really necessary or that you can do without while using the PC Interface.

If You Can't Load a Program (File Format Reader Won't Run)

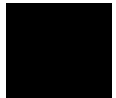
If your program is very large, the PC Interface may run out of memory while attempting to create the database file. You need to exit the PC Interface. Then execute the reader program at the MS-DOS command prompt to create the files.

Index

- A**
 - access to emulation memory not allowed, **2-43**
 - address range field, **3-12**
 - Analysis Begin command, **2-26**
 - Analysis Display command, **2-8, 2-26**
 - analysis specification
 - resetting the, **2-23**
 - saving, **2-24**
 - trigger condition, **2-23**
 - analyzer
 - using the, **2-22**
 - assembler symbol files, **2-8**
 - assemblers, **3-9**

- B**
 - BNC trigger, **3-16**
 - break conditions, **3-17**
 - break entry to monitor subroutine, **2-39**
 - breakpoints
 - software, **2-21**
 - Breakpoints Add command, **2-21**
 - Breakpoints Clear command, **3-9**
 - Breakpoints Display command, **2-22**
 - Breakpoints Set command, **2-22**

- C**
 - cautions
 - protect against static discharge, **2-32**
 - turn OFF power before installing emulator probe, **2-32**
 - verify pin 1 when installing emulator probe, **2-32**
 - clock, **2-34**
 - CMB (coordinated measurement bus)
 - execute signal while emulator is reset, **2-31**
 - trigger, **3-16**
 - commands
 - Analysis Begin, **2-26**
 - Analysis Display, **2-8, 2-26**
 - Breakpoints Add, **2-21**
 - Breakpoints Clear, **3-9**
 - Breakpoints Display, **2-22**



commands (cont'd)

- Breakpoints Set, **2-22**
- Config General, **1-5**
- Config Load, **3-17**
- Config Map, **1-6, 3-9, 3-11, 3-14**
- Config Store, **3-17**
- Config Trigger, **3-14 - 3-15**
- Memory Display, **2-8, 2-15**
- Memory Load, **1-7, 2-6**
- Memory Report, **2-20**
- Processor Break, **2-20**
- Processor Go, **2-19, 2-21, 2-37**
- Processor Reset, **2-31**
- Processor Step, **2-17 - 2-18**
- real-time operation, **2-33**
- selecting PC Interface, **1-4**
- start PC Interface, **1-3**
- System Exit, **1-8**
- System Symbols, **2-9 - 2-12**

communication with target system I/O ports, **2-41**

- Config General command, **1-5**
- Config Load command, **3-17**
- Config Map command, **1-6, 3-10 - 3-11, 3-14**
- Config Store command, **3-17**
- Config Trigger command, **3-14 - 3-15**

configuration

- accessing, **3-2**
- emulator, **1-5, 3-1**
- hardware, **1-1**
- loading, **3-17**
- storing, **3-16**

count

- Processor Step command, **2-19**

coverage, **2-20**

creating/loading user monitor subroutines, **2-43**

cross trigger specification, **3-14**

D data memory, **3-9**

default

- configuration items, **2-33**
- memory map, **3-11**

define addresses driven to target system, **2-38**

- defining memory map terms, **3-12**
- delays, **2-34**
- display
 - memory map, **3-9**
- displaying the trace, **2-26**
- E** emulation
 - in-circuit, **2-32**
 - memory, **2-34**
 - monitor, **2-37**emulator
 - configuration, **1-5**
 - reset, **2-31**
 - status, **1-4**emulator probe
 - installation, **2-32 - 2-33**
 - pin extender, **2-32**
 - precautions, **2-32**equation in address range field, **3-12**
- executing programs, **2-19**
- exit
 - cross trigger configuration, **3-16**
 - memory map, **3-13**
 - PC Interface, **1-8**exit from monitor subroutine, **2-40**
- F** files
 - assembler symbol, **2-8**
- G** getting started, **1-1, 2-1**
- global symbols, **2-8**
- guarded memory, **3-9**
- guidelines for in-circuit emulation, **2-34**
- H** hardware
 - installation, **1-1**
 - high-speed CMOS target system interface, **2-34**
 - HP64000 Reader
 - command (RHP64000.EXE), **2-5**
 - using with PC Interface, **2-5**
- I** IEEE-695 reader command (RIEEE695.EXE), **2-5**
- in-circuit emulation, **2-32**

installation
 emulator probe, **2-32**
 hardware, **1-1**
instructions for accessing target system, **2-41**
instructions not to be used, **2-42**
internal analyzer trigger signals, **3-14**

L line numbers, **2-27**
linkers, **3-9**
load -g command, **2-40**
load map, **3-9**
local symbols, **2-4, 2-10**
location of the subroutines, **2-41**
locked, PC Interface exit option, **1-8**

M make file, **2-2**
mapping memory, **1-5, 3-9**
 adding terms, **3-10**
 other, **3-10**
 types, **3-10**
memory
 accessed during program execution, **2-20**
 displaying in mnemonic format, **2-15**
 emulation, **2-34**
 guarded, **3-9**
 mapping, **1-5, 3-9 - 3-10**
 RAM, **3-9**
 reassignment of emulation memory blocks, **3-14**
 ROM, **3-9**
 target system, **2-34**
 types, **3-10**
Memory Display command, **2-8, 2-15**
Memory Load command, **1-7, 2-6**
Memory Report command, **2-20**
memory types, **3-10**
modify
 memory map, **3-9, 3-11**
monitor, **2-37**
monitor loop subroutine, **2-39**
monitor operation, **2-44**
monitor program, **2-38**
monitor program access time, **2-38**

- N** notes
 - date checking only in PC Interface, **2-7**
 - displaying complete traces, **2-26**
 - reassignment of emul. mem. blocks by mapper, **3-14**
 - terminal window to modify emul. config., **3-2**
 - use required options to include symbols, **2-3**

- O** out-of-circuit operation, **1-5**

- P** PC Interface
 - exiting the, **1-8**
 - HP64000 Reader, **2-5**
 - selecting commands, **1-4**PC Interface startup command, **1-3**
 - pin extender, **2-32**
 - prerequisites, **1-1**
 - Processor Break command, **2-20, 2-31**
 - Processor Go command, **2-19, 2-21, 2-37**
 - Processor Reset command, **2-31**
 - Processor Step command, **2-17 - 2-18**programs
 - coverage, **2-20**
 - memory, **3-9**

- Q** qualifiers, analyzer status, **2-24**

- R** RAM, **3-9**
 - real-time
 - commands which affect, **2-33**
 - performance, **2-33**
 - reduce monitor program access time, **2-38**
 - registers used by the user monitor code, **2-42**
 - relocatable files, **3-9**
 - removing symbols, **2-12**
 - reset
 - emulator, **2-31**
 - memory map, **3-14**
 - run from, **2-37**
 - reset entry to monitor subroutine, **2-39**
 - resetting the analyzer specifications, **2-23**
 - restrictions on adding monitor code, **2-41**

- ROM, **3-9**

- run
 - from reset, **2-37**
 - running programs, **2-19**
- S**
 - saving analysis specifications, **2-24**
 - selecting PC Interface commands, **1-4**
 - simple trigger, specifying, **2-23**
 - software breakpoints, **2-21**
 - clearing, **2-22**
 - defining (adding), **2-21**
 - displaying, **2-22**
 - setting, **2-22**
 - specifications, **2-34**
 - See also analysis specification
 - timing, **2-34**
 - stack provided for including subroutines, **2-43**
 - starting the trace, **2-26**
 - startup command for PC Interface, **1-3**
 - static discharge, protecting the emulator probe against, **2-32**
 - status (analyzer) qualifiers, **2-24**
 - status line, **1-4**
 - step, **2-17**
 - count specification, **2-19**
 - subroutines for the monitor, **2-39**
 - symbol file (<file>.hps), **2-2**
 - symbols
 - .HPS file format, **2-3**
 - local, **2-2**
 - removing from the emulator, **2-12**
 - transferring to the emulator, **2-11**
 - System Exit command, **1-8**
 - System Symbols command, **2-9 - 2-12**
- T**
 - tailoring the monitor, **2-39**
 - target system
 - clock, **2-34**
 - memory, **2-34**
 - memory map, **3-10**
 - monitor access, **2-37**
 - real-time operation, **2-33**
 - target system interface, **2-34**
 - terms in memory map, **3-10**

- timing
 - specifications, **2-34**
- trace
 - analyzer signals, **2-22**
 - description of listing, **2-27**
 - displaying the, **2-26**
 - starting the, **2-26**
- transferring symbols, **2-11**
- trig1 and trig2 lines, **3-15**
- trigger, **2-23**
 - configuration, **3-14**
 - signal interaction, **3-15**
 - specifying a simple, **2-23**
- U** unlocked, PC Interface exit option, **1-8**
- unmapped memory, **3-14**
- user code separate from monitor code, **2-41**
- Z** zoom, window, **2-9, 2-15**



Notes

