HP 64700-Series Emulators

PC Interface

Reference

HP Part No. 64740-97005
Printed in U.S.A.
May, 1990

Edition 3

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.

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	64740-90906, November 1987 E1187
Edition 2	64740-90906, September 1988 E0988
Edition 3	**64740-97005, May 1990**

Using this Manual

Topics Covered

This manual, the *HP 64700 Series Emulators PC Interface User's Reference*, explains how the PC Interface operates on HP Vectra and IBM compatible PCs. It covers installation and use of the PC Interface, including:

- An Introduction - Chapter 1
- Installation - Chapter 2
- Getting Started - Chapter 3
- Using the Windows - Chapter 4
- Using System Features - Chapter 5
- Controlling Emulators - Chapter 6
- Creating and using Command Files - Chapter 7
- Creating Function Key Macros - Chapter 8
- Syntax Summary - Appendix A

The index contains terms and corresponding page numbers so that you can locate information quickly.

The *Analyzer PC Interface User's Guide* covers PC Interface analysis capabilities.

Understanding HP 64700 Terms

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

Using the Manuals

Use this manual with your *HP 64700-Series Emulator PC Interface User's Guide*. That manual contains information about operating the emulator specific PC Interface for that emulator.

Refer to the HP 64700-Series Manual Maps for directions for getting started with the appropriate manuals, the various interfaces, and with using your emulator/analyzer. You can find the maps in the package marked Read Me First.

Contents

1 Introducing the PC Interface

Features of the PC Interface 1-1
Where to Find More Information 1-3
About the HP 64700-Series Products 1-3
About Configurations 1-3
About MS-DOS/Vectra 1-3
About HP Support Services 1-3
About Analysis 1-3

2 Installation and Setup

Topics Covered 2-1
Before You Install the PC Interface 2-1
Install the PC Interface 2-2
Define the Emulator for the PC Interface 2-3
Define Shell Environment Variable Space 2-6
Define Break Checking Variable 2-7
If the Baud Rate is < = 1200 2-7
What to Do If Problems Occur 2-7
Problems with Coresident Programs 2-8
Memory Use 2-8
Interrupt Service Routines 2-8
Troubleshooting 2-8
If All Else Fails 2-9

3 Getting Started

Topics Covered 3-1
Before Getting Started 3-1
Conventions 3-2
What We Mean When We Say “Form” 3-2
Create a Configuration File 3-2
Function of this Configuration File 3-3
Start the PC Interface 3-3
If You Didn’t Include the /m Option 3-5
Options You Can Use 3-6

Understand the PC Interface Screen 3-7
Use Keys to Perform Various Functions 3-8
Escape Key Actions 3-9
Use the PC Interface Commands 3-10
Follow a Tutorial to Learn the PC Interface 3-12
Access the Host Computer System 3-13
Learn How to Create a Command File 3-14
Create a User-defined Window 3-22
Load Data Into Another Window 3-25
Learning Basic Emulation Features 3-26
Execute the Command File 3-27
Exit the PC Interface 3-27

4 Using Windows

Topics Covered 4-1
Window Functions and Features 4-1
Functions of the System Windows 4-2
User-defined Windows 4-4
Window Attributes 4-4
Window Features 4-5
Accessing the Window Functions 4-5
Open a Window 4-5
How Many Windows You Can Create 4-7
Delete a User-defined Window 4-7
View a Window 4-8
Hide a Window 4-8
Erase a Window 4-8
Activate a Window 4-9
Load a Window 4-9
Zoom a Window 4-9
Store a Window 4-10
Search a Window 4-10
Summary of Window Control Characters 4-11
Window Characteristics and Parameters 4-11
The Cursor Location is Retained 4-11
About the Window Characteristics 4-11
Color 4-12
Size 4-12
Name 4-12
Autoclear 4-13
Buffer Size 4-13

Display 4-13
Scroll 4-14
Window Tutorial 4-14
Delete the Window 4-16
How to Use the System Terminal Window 4-17
Commands You Shouldn't Execute 4-19
Specifics About the System Terminal Window 4-20
Printing Window Contents 4-20

5 Using System Features

Topics Covered 5-1
How to Execute System Level Commands 5-1
Execute a Single System Level Command 5-1
Execute multiple System Level Commands 5-3
Log Commands and Output to a File 5-4
Load and Display Symbols 5-4
Details About Symbols 5-5
Global Symbols 5-5
Local Symbols 5-6
Symbols defined as local and global 5-7
Emulator Symbol Capabilities 5-8
Global Symbol Options 5-8
Local Symbol Options 5-8
How to Store the PC Interface Configuration 5-9
PC Interface Configuration File Details 5-9
Window Configuration Information 5-10
Emulator-Specific Information 5-11
How to Load the PC Interface Configuration 5-13
Example Configuration Files 5-13
How to Exit the PC Interface 5-15

6 Controlling Emulators

Topics Covered 6-1
Modify the General Emulator Configuration 6-2
Microprocessor Execution 6-2
Run the Microprocessor 6-2
Step the Microprocessor 6-4
Break the Microprocessor 6-5
Reset the Microprocessor 6-5
Display I/O Port Addresses 6-6
Modify I/O Port Addresses 6-7

Start the Emulator When CMB Events Occur 6-8
The Memory Mapper 6-9
Modify the Memory Mapper 6-9
Storing the Memory Map 6-12
Reset the Memory Map 6-12
Controlling Memory 6-13
Display Memory 6-13
Modify Memory 6-14
Store Memory to a File 6-15
Load Memory from a File 6-15
Copy Memory 6-16
Find a Data Pattern in Memory 6-17
Entering Memory Ranges 6-17
Break Events 6-18
Configure Break Events 6-19
Control Software Breakpoints 6-19
Display Software Breakpoints 6-20
Add Software Breakpoints 6-20
Remove Software Breakpoints 6-21
Set Software Breakpoints 6-22
Clear Software Breakpoints 6-23
Emulator Registers 6-24
Display Registers 6-24
Modify Registers 6-24
Coverage Analysis 6-25
Reset the Coverage Hardware 6-26
Run the Processor 6-26
Check an Address Range 6-26
Make a Percentage Measurement 6-27
Using the Emulator While Connected to a Target System . . . 6-27
Where to Find More Information 6-28

7 Creating and Using Command Files

Topics Covered 7-1
What Are Command Files? 7-1
What You Can Do with Command Files 7-2
Commands Not to Include 7-2
Command File Specifics 7-3
Nesting Command Files 7-3
How to Create Command Files 7-3
Using an Editor 7-3

Using the System Log Feature 7-4
How to Use Command Files 7-7

8 Function Key Macros

Introduction 8-1
How to Define Function Key Macros 8-1
Creating a Macro 8-1
Nesting and Chaining Macros 8-3
Keystroke Representations 8-3
Editing Macros 8-4
Organizing Your Macros 8-4
Saving/Restoring Macros 8-5
Predefined Macros 8-5
How to Use Function Key Macros 8-5
Examples 8-6

A PC Interface Syntax Summary

Introduction A-1
Conventions Used A-1
Window Syntax Summary A-2
System Syntax Summary A-3
Registers Syntax Summary A-4
Processor Syntax Summary A-5
Breakpoints Syntax Summary A-6
Memory Syntax Summary A-7
Configuration Syntax Summary A-8
Analysis Syntax Summary A-9
Expressions A-10
Values A-10
Operators A-12
Using Expressions in Addressing and Analyzer Expressions A-16
Analyzer Pattern Expressions A-17
Syntax A-17
Description A-18
Pattern Labels and Ranges A-18
Sets A-18
Intraset Operations A-18
Interset Operations A-19
Combination A-19
DeMorgan's Theorem and Complex Expressions A-20

Illustrations

Figure 1-1. Where the PC Interface Operates 1-2
Figure 2-1. How the PC Interface locates an Emulator 2-3
Figure 2-2. Example Emulator Device Table File 2-5
Figure 2-3. How the PC Interface recognizes Emulators 2-6
Figure 3-1. Example Form (opening a Window) 3-2
Figure 3-2. Example Configuration File 3-3
Figure 3-3. The Default PC Interface Screen 3-4
Figure 3-4. PC Interface Screen with “newwindow” 3-7
Figure 3-5. How to choose an Option 3-11
Figure 3-6. Example Tutorial File 3-14
Figure 3-7. How the Log Command Operates 3-15
Figure 3-8. Displaying the File You Created 3-18
Figure 3-9. Example Command File “tutorial” 3-19
Figure 3-10. PC Interface and System Interaction 3-21
Figure 3-11. Responses for creating MYWINDOW 3-23
Figure 3-12. Your User-Defined Window 3-23
Figure 3-13. Methods for exiting the PC Interface 3-26
Figure 4-1. PC Interface Window Representations 4-2
Figure 4-2. More System Window Representations 4-3
Figure 4-3. How to open a Window 4-6
Figure 4-4. Your User-Defined Window (MYWINDOW) . . 4-16
Figure 4-5. System Terminal Window displayed 4-19
Figure 6-1. Default Memory Map 6-11
Figure 7-1. Using the Log Commands 7-6

1

Introducing the PC Interface

Welcome to the PC Interface. This is a program that allows you to interact with and control HP 64700-Series emulators, and communicate with your host computer. As you will see, the PC Interface provides ways of controlling your HP 64700-Series emulator that are more straightforward than using the emulator's built-in Terminal Interface.

The PC Interface allows you to monitor all emulation and analysis data. In addition, executing PC Interface commands can be done quickly because you need only type a single letter to perform a function.

Features of the PC Interface

The PC Interface allows you to:

- Organize the results of emulation commands in window-oriented displays.
- Define custom windows.
- Enter your data in easy-to-follow forms.
- Use control characters for quick command execution.
- Enter commands using the best technique for you.
- Control emulation and analysis activities.

- Save output to a file.
- Use symbolic addresses when running your program.
- Store and load absolute files.
- Store and load the PC Interface configuration.
- Create and use command files.
- Create and use function key macros.
- Temporarily access the MS-DOS system level at any time.
- Execute individual MS-DOS commands.

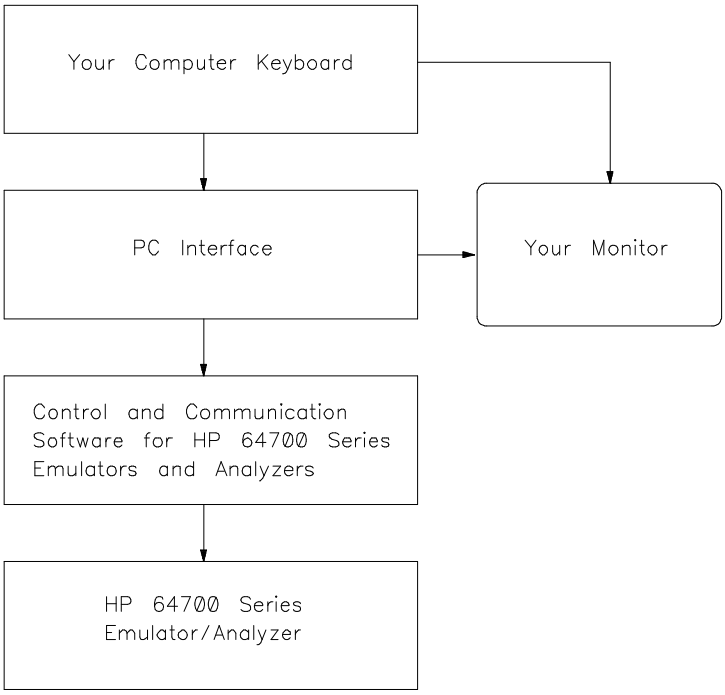


Figure 1-1. Where the PC Interface Operates

1-2 Introducing the PC Interface

Where to Find More Information

About the HP 64700-Series Products

Refer to the *HP 64700 Emulators System Overview* manual for details on the complete HP 64700-Series environment.

About Configurations

Refer to the *HP 64700 Emulators Hardware Installation And Configuration Manual* for information on configuring your HP 64700-Series system. The *PC Interface Emulator User's Guide*s contain details for configuring HP 64700-Series Emulators.

About MS-DOS/Vectra

You can find information on MS-DOS and the HP Vectra in your set of MS-DOS and HP Vectra Manuals.

About HP Support Services

Refer to the *HP 64700 Emulators Support Services* manual for information on the following:

- what to do if your HP 64700-Series emulator fails
- how to get your HP 64700-Series emulator fixed
- Software Materials Subscription
- Response Centers
- HP Sales/Service Offices

About Analysis

Refer to the *Analyzer PC Interface User's Guide* for details about analyzer expressions and using the emulation analyzer or external analyzer with the PC Interface.

Notes

2

Installation and Setup

Topics Covered

- Before You Install the PC Interface
- Install the PC Interface
- Define the Emulator for the PC Interface
- What To Do If Problems Occur

Before You Install the PC Interface

Before you begin installing the PC Interface:

1. Verify that your system has the required hardware (refer to the *Hardware Installation and Configuration Manual*).

2. Make sure your system has enough disk space to hold the PC Interface software. To operate the PC Interface, your system must have 640 Kbytes of standard memory. You will need approximately 600 Kbytes of storage on your hard disk for the standard PC Interface. The PC Interface version that supports integrated timing analysis requires approximately 750 Kbytes of disk space.

3. Make sure that your \config.sys file contains the statement "files = 20." In this statement, the number of files specified must be 20 or more for the PC Interface to work properly.

Install the PC Interface

Your PC Interface software comes on media that is compatible with your host computer.

You will find correct software installation instructions for the PC Interface in the MS-DOS software installation instructions you received.

Note

Be sure to install all the PC Interface software supplied on the media, including the emulator device table file. The file's default name is "64700tab" in the \hp64700\tables directory.

Define the Emulator for the PC Interface

You must define your emulator's environment in a file so that the PC Interface has the correct communication parameters for your emulator. The PC Interface reads the file on startup.

Once you do this, your emulator(s) will work with the PC Interface.

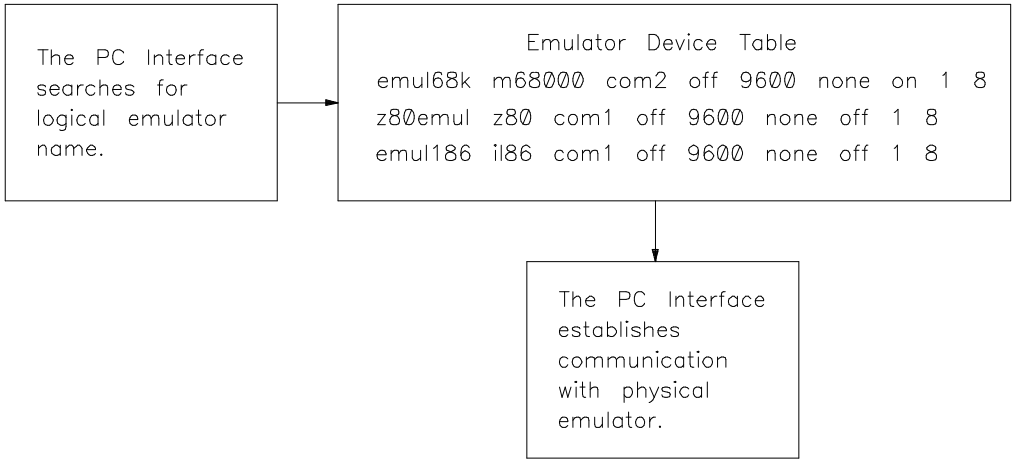


Figure 2-1. How the PC Interface locates an Emulator

Note

The emulator device table file contains emulator port communication information. It allows you to access the PC Interface with correct emulator communication parameters by specifying the emulator device name. The PC Interface reads the emulator device table file to determine which emulator is connected to each system port. If you are using multiple emulators, and have multiple environments defined in the device table file, the PC Interface will recognize each of them, and allow you to start any one.

To allow the PC Interface to recognize an emulator, follow these steps.

1. Use an editor to modify the existing emulator device table file to define the environment for your emulator(s). This file is on the media with the PC Interface, and will be located on your system as "\hp64700\tables\64700tab" by default.

2. You may relocate the emulator device table file, by setting the environment variable HPTABLES= "path of file" (for example, HPTABLES= \mydir\emulator). You may include the statement in the \autoexec.bat file, or execute it from the MS-DOS command line. Refer to the **set** command in your *MS-DOS Manual*.

Note

You can define multiple environments for multiple emulators in the emulator device table file. There is only one emulator device table file.

For example, the following emulator device table file provides communication between the PC and three emulators, including the 68000, Z80, and 80186. Refer to your *HP 64700 Emulator PC Interface User's Guide* for the correct device name for your emulator.

```
emul68k m68000 com2 off 9600 none on 1 8
z80emul z80 com1 off 9600 none off 1 8
emul186 i186 com1 off 9600 none off 1 8
```

Figure 2-2. Example Emulator Device Table File

In the second line of the example emulator device table file:

Parameter	Meaning
z80emul	logical emulator name
Z80	emulator name (defined by you)
com1	physical port name
off	transparency mode setting
9600	baud rate
none	parity
off	transmit/receive pacing
1	stop bits
8	dataword width in bits

Notice the emulator name (z80emul). You will use this in chapter 3 when starting up the PC Interface.

When you finish modifying the emulator device table file, save it in the current directory.

Note

If your emulator is not connected to port 1 (com1), be sure to include the correct port in the emulator device table. Otherwise, the PC Interface will not recognize the emulator.

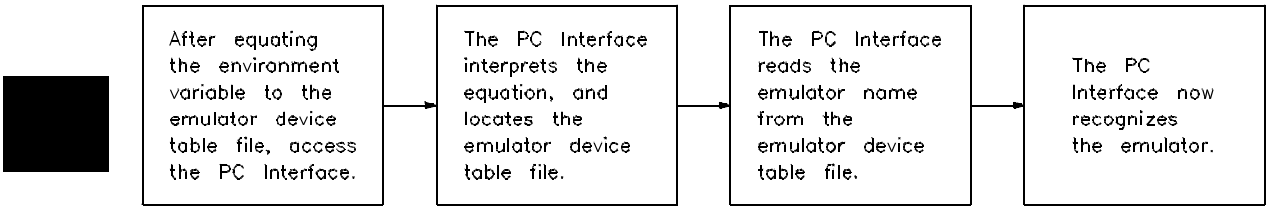


Figure 2-3. How the PC Interface recognizes Emulators

3. Equate the shell environment variable to the emulator device table file.

```
set  HPTABLES=c:\<directory_path>
```

   < directory_path> is the directory on your computer where the emulator device table file (64700tab) resides. The default directory is \hp64700\tables. You can put this environment variable in your \autoexec.bat file. Then, the environment variable will be set automatically every time you reboot your computer.

4. The PC Interface uses a file named **pcexpr.x** to install additional command features in the emulator. This file is installed in the directory **\hp64700\bin** by default. If you install it in a different directory or move it, you need to set the environment variable HPBIN as follows:

```
set HPBIN=c:\<directory_path>
```

Define Shell Environment Variable Space

If your system runs out of space for environment variables, you can redefine the number of allowable environment variables that your computer will handle. Depending on the version of your MS-DOS software, this may be defined differently.

For MS-DOS versions 3.0 and 3.1, the **shell** command defines these environment variables in a certain format. In the \config.sys file, you would add the command:

shell= command.com /e:xx

"xx" indicates the number of 16-byte memory blocks to reserve for your computer environment. This value can be from 10 to 62.

MS-DOS 3.2 and 3.3 use a similar scheme, but the "xx"indicates the number of bytes to reserve for the environment. Refer to your *MS-DOS Manual* for additional details.

Define Break Checking Variable

In the \config.sys file, you should add the **break= on** command to allow the system to check for two break conditions (**Ctrl-Break** and **Ctrl-c**).

If the Baud Rate is < = 1200

If you toggle the "xon" parameter using the **stty** command when running the emulator at 1200 baud or less, invalid characters will be displayed. For example, if the **stty** command shows:

```
stty A 1200 xon
```

and then you enter

```
stty=xon
```

invalid characters will appear on screen.

To avoid this problem, set switch S13 on the HP 64700 rear panel to "1." This prevents use of "xon", even if you enter the **stty** command shown above. The PC Interface will start properly.

What to Do If Problems Occur

If you encounter problems while installing the PC Interface:

1. Make sure that your system has the required 640 kilobytes of memory for operating the PC Interface.
2. Be sure to include the "files= 20" statement in your \config.sys file.

If your system still has problems, reboot the PC to clear any other system problems that may exist.

Problems with Coresident Programs

Coresident programs include TSR (terminate-and-stay-resident) utilities, device drivers, and operating system modifier programs (such as LAN shells). These programs may cause problems with PC Interface operation.

Memory Use

Once a coresident program loads, it may not leave enough free memory for the PC Interface to load and run correctly. Or, the coresident program may dynamically allocate memory during execution, causing problems when the PC Interface needs memory to perform its functions.

Interrupt Service Routines

In certain situations, the PC Interface cannot service incoming data from the serial port (sent by the emulator) because of interrupt interference by coresident programs. The results are lost data and erroneous measurement displays. The problems caused by coresident programs may be one of the following:

- The coresident program disables interrupts while performing its actions. This includes certain disk caching programs.
- The coresident program has an interrupt service routine of higher priority than the PC Interface, and the routine takes an excessive amount of time. Thus, more than one character is received by the serial interface during the service routine, causing data loss. (The standard PC serial port is unable to accept more than one character at a time.) Programs that cause this problem include HP ARPA Services for the PC, and may include others.

Troubleshooting

- If you think a coresident program may be causing problems, remove each program (device driver, TSR, LAN shell, and so on) from your system until the problem is resolved.

- The HP RS-422 communications interface (HP 64037A) uses a buffer for incoming data, which may help prevent data loss for all but exceptionally long service routines.

If All Else Fails

If all else fails, refer to the *HP 64700-Series Emulator Support Services* manual. This manual will lead you in the right direction for solving problems.

Your emulator and the PC Interface are ready to communicate. Proceed to the next chapter, which shows you how to get started.

Notes

3

Getting Started

Topics Covered

This chapter teaches you to use the PC Interface. You'll get the most benefit from this chapter if you do the examples on your system using the PC Interface. You will learn to:

- Create a Configuration File
- Access the PC Interface
- Understand the PC Interface Screen
- Use Keys to Perform Various Functions
- Use the PC Interface Commands
- Follow a Tutorial to learn the PC Interface
- Exit the PC Interface

Before Getting Started

Make sure that you are working in the same directory where the PC Interface resides, or that you have added that directory name to your PATH variable. For MS-DOS computers, modify the **\autoexec.bat** file to add the directory to your PATH statement. Refer to the appropriate *Emulator PC Interface User's Guide* if necessary.

Also make sure that you have added you emulator's environmental specifications to the emulator device table file. See chapter 2 for more information.

Conventions

This manual uses the following conventions:

Bold text indicates commands that you type.

< > (angle brackets) enclose variables that you type.

^ (up carat) indicates the keyboard Ctrl (control) key.

^ z means press and hold the **Ctrl** key and then press **z**.

What We Mean When We Say "Form"

A form is a part of the PC Interface display that you use to enter data. It differs slightly between the various PC Interface options. Forms allow you to modify current settings, set parameters for the first time, load files, create user-defined windows, and so on. You will learn more about forms through examples in the rest of this manual. An example form for creating a user-defined window appears like this:

```
Window Open -> ___________Top row ___Bottom row___ Autoclear? _
Buffer size ____       Left edge ___Right edge ___ Display? _
```

Figure 3-1. Example Form (opening a Window)

Create a Configuration File

Use an editor to create the example configuration file shown in figure 3-2. Chapter 4 explains the parameters in the file, so don't worry about their meanings now. But, if you are using a monochrome monitor, change "color" to "mono" in the $misc section of the configuration file shown below.

Name the configuration file "myconfig." This file should be in the directory where you are now working. Take time now to look at the list of files in the current directory, and make sure that the configuration file you just created (myconfig) is there. Then, verify that the content of "myconfig" appears exactly like the example configuration file shown above.

```
$misc
color  white  blue
$misc
$userwin
newwindow  T  T  F  0,0  10,35  30
$userwin
```

Figure 3-2. Example Configuration File

Function of this Configuration File

This file configures your PC display monitor when you access the PC Interface. It also creates a user-defined window. Configuration files can perform many other functions (see chapter 6 for details).

Start the PC Interface

There are several ways to start the PC Interface.

Note

If you are using a monochrome monitor, add the **/m** option in each command that shows how to access the PC Interface. If you don't use the **/m** option, fields in the forms will not have borders because the borders are "white" on "white." Using **/m** avoids this situation. For example, for the next command you would type: **pcz80 /m** < **z80emul**> if you are using the HP 64753 Z80 emulator.

The first method is to enter the PC Interface using default parameters.

For example, you enter:

```
pc<product>  <emulator_name>
```

The term < product> refers to your type of emulator. For example, if you are using the Z80 emulator, you would type **pcz80**. For the HP 64742 68000 emulator, you would specify **pcm68k** for

< product> . For the 80186/88 emulator (HP 64764/65), you would use **pci18x**. Your *Emulator PC Interface User's Guide* has the details.

The name we gave to the Z80 emulator is "z80emul," and is included in the emulator device table file explained in chapter 2. If you included something other than **z80emul** in your emulator device table file, be sure to use that term here also.

This command displays three windows by default:

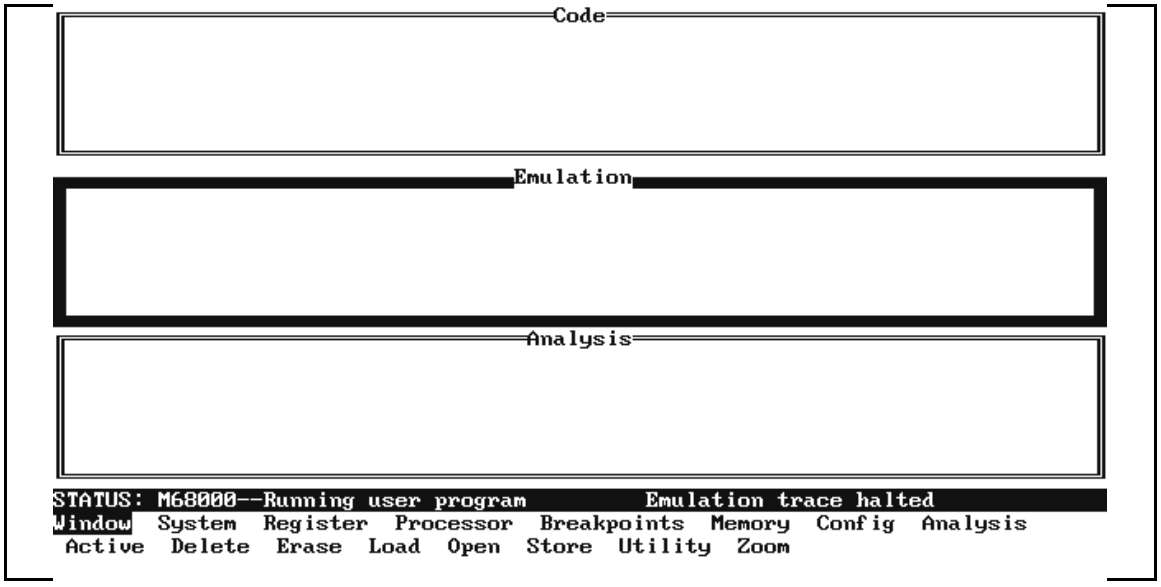


Figure 3-3. The Default PC Interface Screen

Exit the PC Interface now by entering:

```
System Exit Locked
```

This will return control to the MS-DOS operating system.

```
pc<product> /w myconfig <emulator_name>
```

Note

If you used something other than "z80emul" in the emulator device table file, be sure to use that same term here.

You typed in a command that initiated a PC Interface session, indicated that a configuration file would be used (**/w**), specified the name of the configuration file that customizes the PC Interface window and emulator environment (myconfig), and supplied the emulator name (z80emul). If you used the **/m** option, the PC Interface recognized your monitor as a monochrome monitor.

The PC Interface activates after several moments. The display shows four windows on screen: the three default windows, and a user-defined window (newwindow). The status line and PC Interface key descriptions appear at the bottom of the screen.

Note

As the PC Interface starts, you may see the message "Loading host commands" flashing at the bottom of the screen for a few seconds. The PC Interface is loading extensions to the HP 64700-Series Terminal Interface command set (from the file **pcexpr.x**—see chapter 2 for more details). The new command is called **_pc** and is used by the PC Interface for expression validation. Do not use this command within the System Terminal window.

If You Didn't Include the /m Option

When you start the PC Interface, the default monitor type is color. Don't worry if you didn't include the **/m** option when starting up the PC Interface while using a monochrome monitor. At first you will notice that the screen is difficult to follow, because some borders are invisible. To view the screen as you would with the **/m** option, change the monitor type within the PC Interface. To do this, enter:

```
Window Utility Color
```

A form is displayed that contains data for the window color and monitor type parameters. Press:

```
<Tab>
```

Notice that the monitor type changed to “mono.” When you save the data in the form (by pressing **End Enter**), you will notice that the PC Interface screen is much easier to read.

Options You Can Use

You can use the following options when accessing the PC Interface:

/m tells the PC Interface that you are using a monochrome monitor. For the PC Interface, the default monitor type is color. Because the default monitor type is color, the commands in this chapter do not contain the **/m** option. Remember to include this option when using a monochrome monitor, so that the PC Interface forms and windows display correctly.

/w tells the PC Interface to load a configuration file, which will customize the PC Interface window and emulator environment. The specifications you define in the configuration file will be interpreted by the PC Interface for the customization process.

/c instructs the PC Interface to take input from a specified command file. Within a command file you can include various commands to perform functions automatically when accessing the PC Interface (see chapter 7 for details on command files).

/? displays the options available from the command line when starting the PC Interface.

The complete syntax for invoking the PC Interface is:

```
pc<product>  [/m]  [/w config_file]  [/c command_file]  <emulname>
```

Understand the PC Interface Screen

Four windows should be displayed on the screen. The upper left window should be labeled “newwindow”. It was created when you entered the PC Interface using the configuration file named “myconfig.”

The PC Interface screen should now resemble:

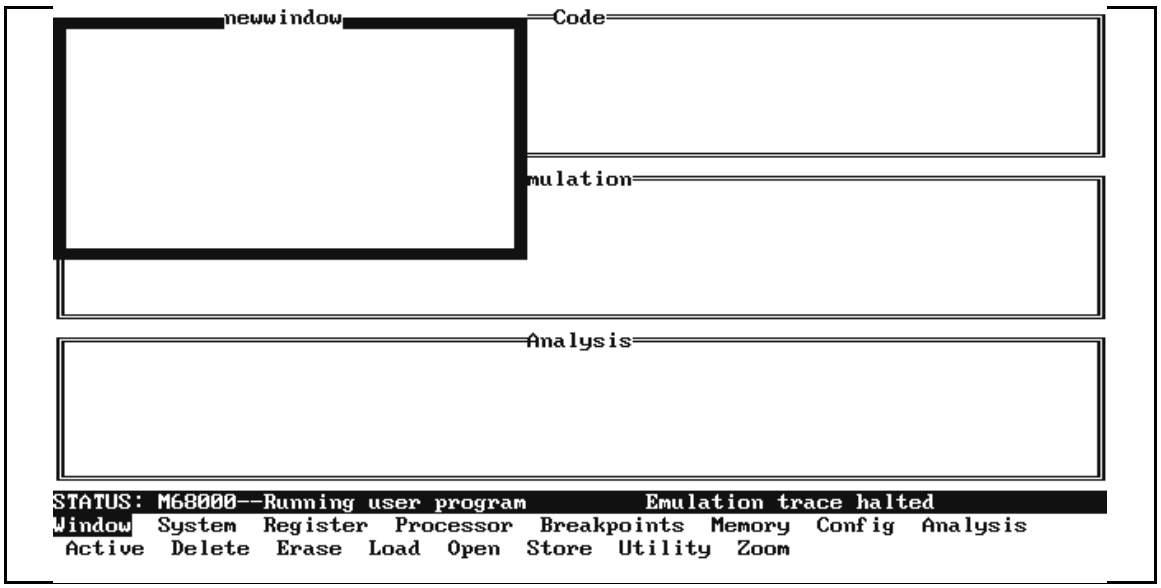


Figure 3-4. PC Interface Screen with "newwindow"

The PC Interface screen is 80 columns wide, with the left column defined as 0 and the right column defined as 79. The screen height is a total of 25 rows. The top row is row 0, and the bottom row is row 24.

The PC Interface always uses four of these 25 rows.

The functions of these bottom four rows are:

- 1 Status line displays emulation and PC Interface status.
- 2 Data Entry lines allow you to enter data.

- 1 Command/Message line keeps you informed.

When you need to be alerted of some activity, the line above the Status line will show “ALERT:” or “ERROR:” and include a message. ALERT messages do not always report errors.

Use Keys to Perform Various Functions

You will be using many keyboard keys. Below are the ones that you will use most often.

Note 

The functions of the following keys depend on whether you use them to enter commands or to enter data in a form.

Key	Function
Backspace	Deletes characters from right to left.
Ctrl [	Returns to the previous PC Interface level (same as ESC key), and aborts some PC Interface operations. See the list following this table.
**Ctrl **	Aborts System Terminal operation.
Ctrl Break	Always interrupts data transfers between the emulator and the host that were initiated by the present command (thus canceling the command).
Ctrl c	Usually interrupts data transfers between the emulator and the host that were initiated by the present command (thus canceling the command).
Ctrl Left Arrow	Moves through text in forms without overwriting text in fields.
Ctrl Right Arrow	Moves through text in forms without overwriting text in fields.

Key	Function
Ctrl ScrLck	Same as **Ctrl Break**.
DEL	Deletes a character at the cursor location.
End	Shows bottom of text in windows. Also saves information in a form when followed by pressing **Enter**.
Enter	Enters commands into PC Interface and emulation/analysis system.
ESC	Same as **Ctrl [**. Exits or aborts various PC Interface operations. See the list following this table.
Home	Shows top of text in the active window.
Ins	Allows insertion of characters at the cursor location.
Left/Right Arrows	Moves cursor to previous/next location.
Pg Dn	Shows the next page of text or data in a window.
Pg Up	Shows the previous page of text or data in a window.
Shift Tab	Selects the previous valid option in a form field.
Tab	Selects the next valid option in a form field.
Up/Down Arrows	Select windows in forms and can scroll window text.
^ z	Zooms a window to full screen size, or back to original size.
^ e	Erases the currently active window.
^ a	Activates another window automatically.
^ r	Recalls the last command you executed.

Escape Key Actions

The escape key (**Esc** or **Ctrl-[**) has the following actions:

- Exit from any data form without completing the operation.
- Aborts command file execution, command recall (^ R), or a key macro.

- Aborts a window search operation.
- Aborts a window store operation.
- Exits Memory Display Repetitively mode.
- Exits timing waveforms displayed in Analysis Display External.

Use the PC Interface Commands

You can quickly execute PC Interface commands by pressing the first letter of an option that is displayed at the bottom of the PC Interface screen. Each time you choose an option, the "next level" options are immediately displayed at the bottom of the screen. This can help you choose the next selection.

If you prefer, you can select PC Interface commands by pressing the left and right arrows to highlight a command. Then press <ENTER> to accept the command.

Command lookahead allows you to see the next options. These are displayed on the bottom row of the display, below the current command selections.

Now the PC Interface should be at the main level. Notice the labels displayed at the bottom of the screen. These labels represent the main PC Interface options:

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

If you have not yet accessed the PC Interface, type: **pc< product> < emulname>** . Remember to use the **/m** option if you are using a monochrome monitor. The PC Interface labels should now be present at the bottom of your screen.

Just to become familiar with how the PC Interface operates, access the **System** option. Enter:

```
System
```

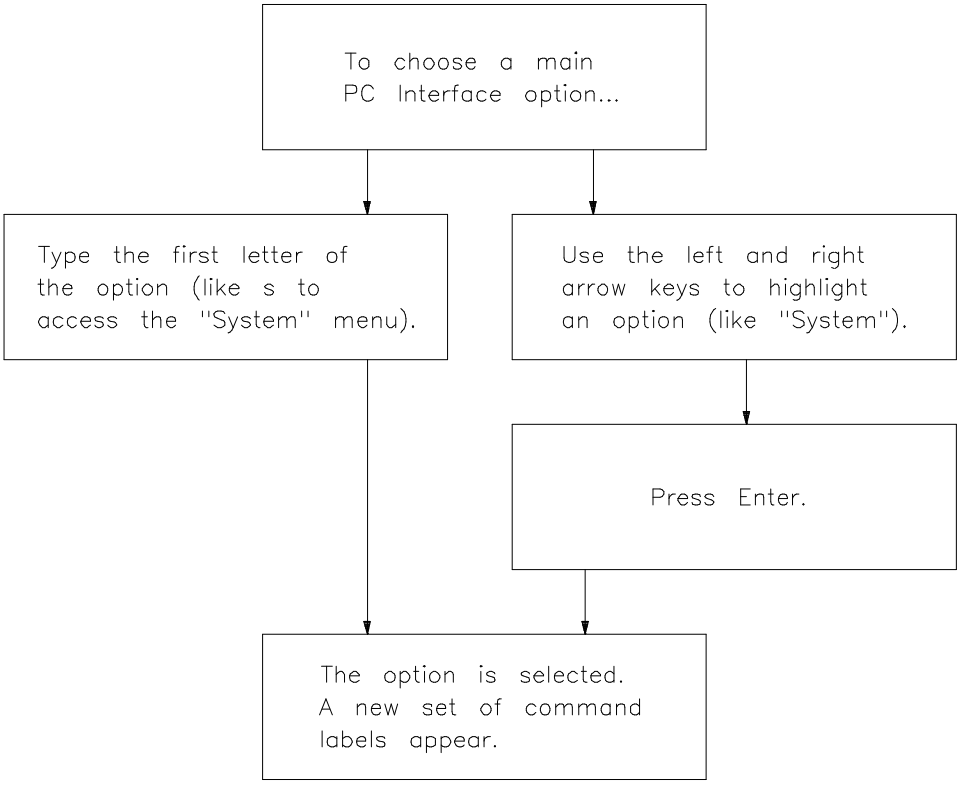


Figure 3-5. How to choose an Option

Did the screen display a new set of labels? If not, try pressing **S**ystem again. (To return to the previous PC Interface level from any level, press ESC.) Enter:

```
MS-DOS Fork
```

See how direct the method is? You have accessed the system fork feature. This provides access to the MS-DOS host computer system level, so that you can perform system functions temporarily from within the PC Interface. We will discuss more about this feature in chapter 5. For now, use some of your favorite commands at this system level. When you are ready, at the system prompt, enter:

```
exit <ENTER>
```

This ends operation at the host computer system level and returns control of the system to the PC Interface.

If you access the system level with the **S**ystem **M**S-DOS **F**ork command, then forget that you're there, and try to access the PC Interface again, the message "Program too big to fit in memory" will appear. Just type **exit** to get back to the PC Interface. Any program which has large memory requirements will not run in the system fork mode. You can see how much free memory is available for programs while at the system level using the MS-DOS **CHKDSK** command.

Follow a Tutorial to Learn the PC Interface

In this tutorial you will type the first letter of a PC Interface option to select the option. For example, we will tell you to type **w** for window. You could instead use the arrows to highlight the window option, then press **Enter** to select that option.

When we say "main level," we are referring to the PC Interface commands you see when you first start it.

If you press a key that is not the first letter of a displayed PC Interface options, the PC will beep. To correct this, choose a valid letter.

In this tutorial you will:

- Learn to use the system features.
- Create a command file.
- Learn to use the window features.
- Learn basic HP 64700-Series emulation features.
- Execute the command file.

Follow the tutorial, working the examples on your system. Remember to include the **/m** option if your monitor is monochrome. To begin, enter

```
pc<product>  <emulator_name>
```

< emulname> is the name you assigned to your emulator in chapter 2. This command starts the PC Interface. If you have any questions about how to start the PC Interface, see chapter 2.

The PC Interface status line should show that your emulator is reset.

Access the Host Computer System

Enter:

```
System MS-DOS Fork
```

You have accessed the host computer system. Try executing some of the commands you typically use. For example:

```
dir <ENTER>
```

Note

Do not run any programs that access the communication port connected to your emulator. Doing this could cause your emulator to stop operating properly.

Files in the current directory are displayed. At this level you can run various applications that you normally use, and can return to the PC Interface with one command. You should be able to print files at this point. Go ahead and execute any other system commands you desire.

When you finish, start your favorite editor. Create the following file:

```
In this tutorial I learned how to:
.  use system features
.  create a command file
.  use the window features
.  use basic emulation features
.  execute a command file
Good Job!    To continue, press any key.
```

Figure 3-6. Example Tutorial File

Save this file in the current directory (which is \hp64000\bin if you started the PC Interface from there), and name it "ILEARNED." Now type:

```
exit <ENTER>
```

This returns control of the system to the PC Interface.

Note

In the commands that follow, watch the PC Interface options at the bottom of the screen change as you press each letter.

Learn How to Create a Command File

Enter:

```
System Log Both Enable
```

You will be prompted for a log file name. Type:

```
tutorial  <ENTER>
```

All commands and results of the commands that you type will be recorded in the file named "tutorial." This is the process of logging commands. This type of file is called a *log file*.

You can log either the input (the commands that you enter) or output (the results of the commands that you entered), or both. For this example, you will log both the input and output.

The PC Interface automatically returns to the main level. We will return to “tutorial” later.

Note

When creating log files, you may specify a path for the log file if you want it located in another directory.

```
System Wait Time 10 <Enter>
```

You have instructed the PC Interface to idle (wait) for 10 seconds. Watch the counter at the bottom of the screen decrement until the 10 seconds tick away. You can press any key during this time to stop the wait process. Enter:

```
System MS-DOS Command dir <ENTER>
```

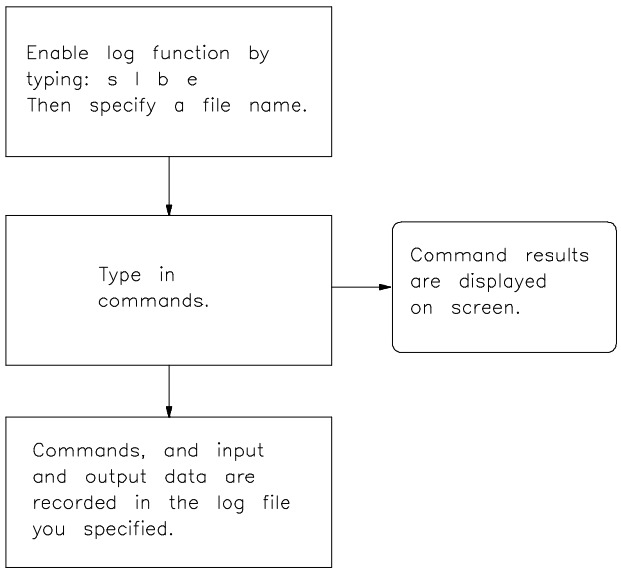


Figure 3-7. How the Log Command Operates

You have temporarily accessed the host system to execute an MS-DOS command. Files in the current directory are displayed. Press any key to return to the PC Interface main level. Enter:

```
System Wait Time 5 <ENTER>
```

"Wait" periods can be useful in command files to allow time to observe the results of processes. When this 5 second wait period ends, enter:

```
Window Active
PRESS:  <Tab> until "Emulation" appears in
the form, then press  <ENTER>
```

The Emulation window now has a highlighted border indicating that it is the active window.

Note 

If you specify the name of a window which is not visible, the message "< Window> is not visible at this point" will appear. Type in the name of a valid window.

```
Window Zoom <ENTER>
```

The Emulation window enlarges (zooms) to the full screen size. You can zoom any of the PC Interface windows. The currently active window is zoomed. Enter:

```
Window Zoom <ENTER>
```

The Emulation window reduces to its default size.

Break the emulation processor to the monitor.

```
Processor Break
```

Now, display the processor's registers:

```
Registers Display <ENTER>
```

Options for displaying your emulator registers will appear. Select an option, and watch what occurs. Refer to your *Emulator PC Interface User's Guide* to learn how your emulator's registers are displayed.

Note

The system windows will always display specific data resulting from an operation. For example, register content will always be directed to the Emulation window, trace results to the Analysis window, symbols information to the Symbols window, and so on.

When you execute the command **R**egister **D**isplay for your emulator, your emulator's registers will automatically be displayed in the Emulation window. Execute those commands and observe the results.

Note

When you enter a command that affects a system window, that window automatically becomes active. That window is then automatically selected for any other window commands.

Now look at that file you created and named "ILEARNED." First, enter:

```
Window Active  <ENTER>
```

Then enter:

```
Code  <ENTER>
```

The system window named "Code" should be displayed at the top of the screen. This window is provided specifically for you to load your programs. Let's load the file you created. Enter:

```
Window Load  <ENTER>
ILEARNED  <ENTER>
```

Notice that the file you created is displayed in the window named "Code." Does it look familiar? You probably can't see the entire file, so enter:

```
^z
```

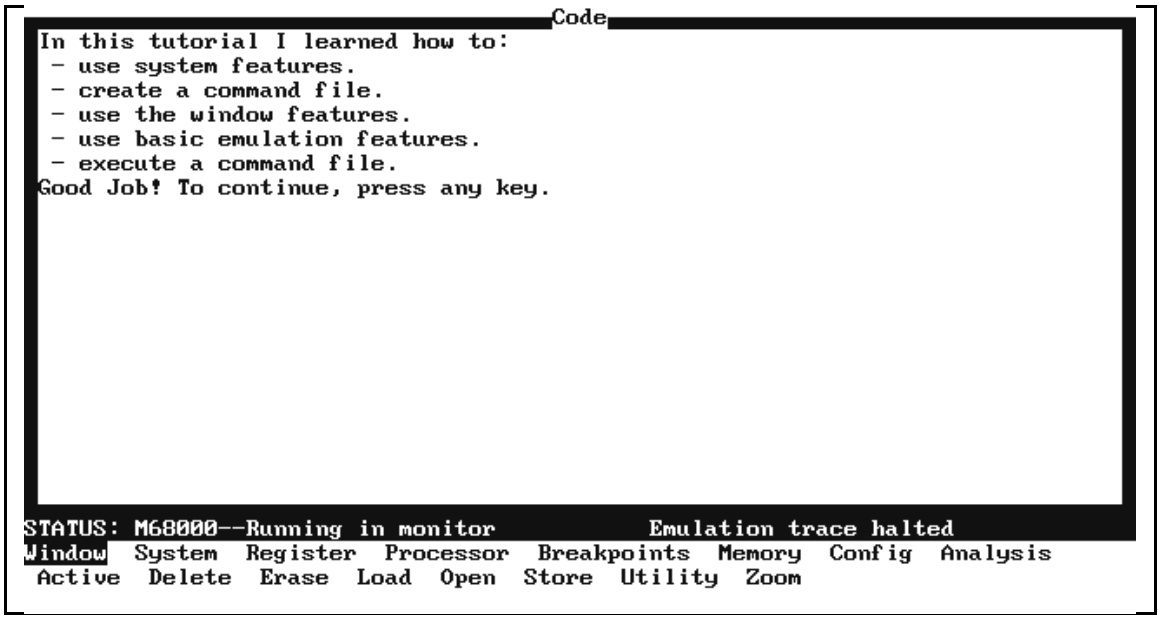


Figure 3-8. Displaying the File You Created

Note

When we say press ^ **z**, this means you should press and hold the **Ctrl** key and then press **z**.

This zooms (enlarges) the window to the full screen size. You should see the entire file. It should look familiar now. Enter:

```
System Wait Key  <ENTER>
```

This feature allows you to delay PC Interface execution until you press a key to continue. When you are ready, enter:

```
any key you desire
```

Now let's turn off logging of commands to the file "tutorial." Enter:

```
System Log Both Disable
```

Commands will no longer be recorded in the file “tutorial.” Look at the contents of “tutorial.” Enter:

```
Window Load Code  <ENTER>
tutorial  <ENTER>
```

This is the command file you created. To observe the entire file, enter:

```
^z
```

```
swt
@10
smc
@dir
swt
@5
wa
@Emulation
wz
@Emulation
wz
@Emulation
pb
rda
# pc = 00000000  st = 2700     ssp = 000006fa usp = 00000000
# a0 = 00000000  a1 = 00000000  a2 = 00000000  a3 = 00000000
# a4 = 00000000  a5 = 00000000  a6 = 00000000  a7 = 000006fa
# d0 = 00000000  d1 = 00000000  d2 = 00000000  d3 = 00000000
# d4 = 00000000  d5 = 00000000  d6 = 00000000  d7 = 00000000
#

wa
@Code
wl
@Code
@ilearned
swk
```

Figure 3-9. Example Command File "tutorial"

You also can use the cursor keys to scroll through the file.

Note

Press **End** to see the end of any file in a window. Press **Home** to see the top of any file in a window. You also can use **Pg Up** and **Pg Dn** to look at one window-sized page of text at a time.

Notice the lines in this file. The commands that you typed were recorded just as you typed them. Data that you entered is preceded with the "@" symbol. The results of PC Interface operations are preceded with the "# " symbol. Use the up and down arrows to scroll through the lines.

Enter ^ **z** to zoom back out of the Code window.

Note

You do not need to edit command files before executing them. The recorded results are "commented out" due to the "# " in the first column, and the PC Interface reads the "@" as data input.

```
Window Store Emulation 1  <ENTER>
```

This stores the Emulation window content to a file. All the lines in the window will be stored, because you specified that the "From line" is "1," the first line in the window. You could store a range of lines by specifying different numbers from the "From line" and "To line" fields in the form. You still need to specify a file name (which you will do in the next step).

Note

Whenever the name of the window you request appears in a form, press **Enter** to accept it. This is true for data in any field of a form.

```
reglist  <ENTER>
```

The content of the Emulation window is stored in a file named "reglist." This file now exists in the current working directory (\hp64000\bin) on your host computer. The PC Interface remains at the main level.

Let's make sure that the file "reglist" was created.

```
System MS-DOS Fork
```

You should now be observing the host system prompt.

```
dir  <ENTER>
```

Check for the file named “reglist.” It should exist in the current listing of files. When you find it, enter:

```
type  reglist  <ENTER>
```

Does this file contain a listing of your emulator’s registers? If it does, proceed to the next example. Otherwise, try executing the commands again, starting with “**W**indow **S**tore **Re**gister **1** < ENTER> < ENTER> .”

Note

To make sure the registers are displayed correctly for your emulator, refer to your *Emulator PC Interface User’s Guide.*

Then continue in the tutorial. Don’t worry if you miss a step. It won’t affect the remainder of the tutorial very much. Type:

```
exit  <ENTER>
```

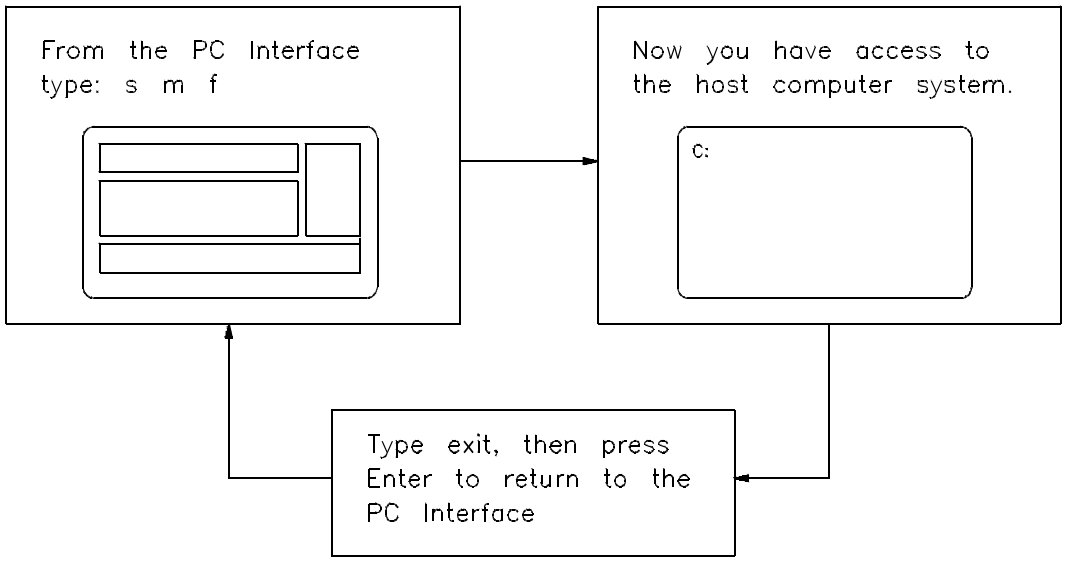


Figure 3-10. PC Interface and System Interaction

This returns control of the system to the PC Interface. Now, enter:

```
Window Store <ENTER> <ENTER> <ENTER> <ENTER>
```

This will resave the register contents in the file "reglist." Enter:

```
System MS-DOS Fork
```

Again, you have accessed the host system using the **System Fork** feature. Notice that the data was appended to the file. Enter:

```
type reglist <ENTER>
```

Notice that "reglist" now contains two listings of the registers, separated by a blank line. The last stored entry is appended to the end of the file.

Note

Appending to files, rather than overwriting them, can be reassuring if you accidentally specify an existing file name to contain the data. With this feature you don't have to worry about losing data already contained in previously stored files.

```
exit <ENTER>
```

This returns control to the PC Interface.

Create a User-defined Window

Enter:

```
Window Open
```

Enter the following terms shown below in bold for each field in the form that is displayed.

Note

If you pass a field in the form, press the left arrow key to return to the previous field. To advance through the fields without changing the data in them, press < ENTER> , or use the right arrow.

```
Window Open ->  MYWINDOW  Enter
Top row ->  9  Enter
Bottom row ->  Enter
Autoclear? ->  Enter
Buffer size ->  50  Enter
Left edge ->  45  Enter
Right edge ->  Enter
Display? ->  Enter
```

Figure 3-11. Responses for creating MYWINDOW

This creates a user-defined window in the lower right section of your display. Notice that this window has a highlighted border. It is active because you just created it.

Your screen should now resemble:

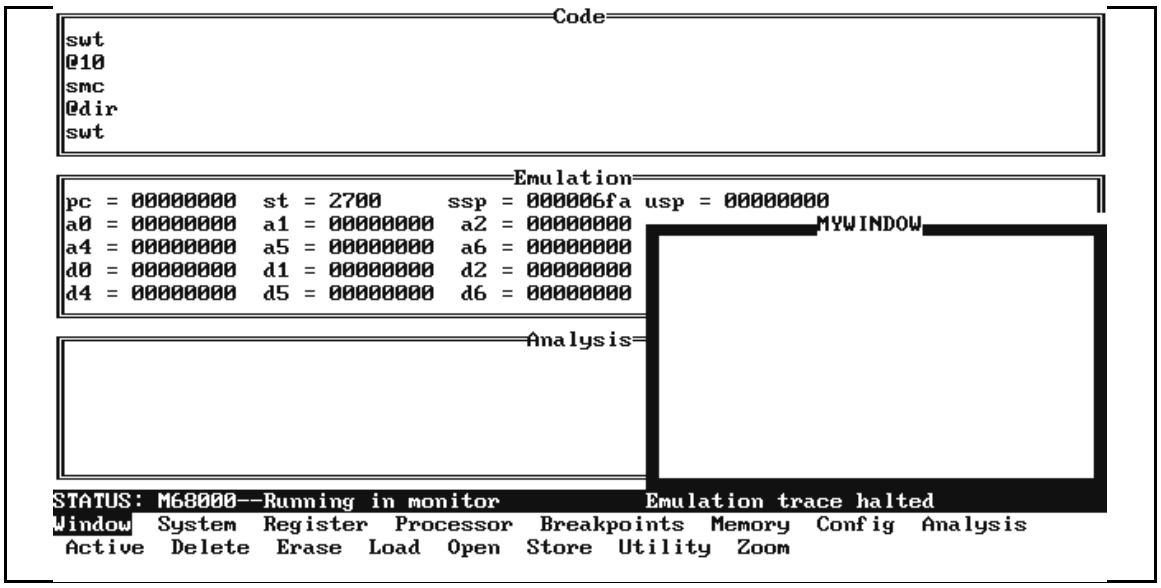


Figure 3-12. Your User-Defined Window

Press:

^z

This is a quick way to zoom a window. Notice that MYWINDOW is now the full screen size.

Let's load a file into your new window. Because MYWINDOW is already active, enter:

Window **L**oad <ENTER>

Note 

You can always use the < Tab> and < Shift> < Tab> keys to select other window names within a form.

Enter:

reglist <ENTER>

This gives the filename to load into the window. The contents of the file "reglist" will be displayed in the user-defined window named MYWINDOW. Enter:

Window **E**rase <ENTER>

This erases your user-defined window contents. You will be prompted for a "y" response to confirm the erasure. Type:

y <ENTER>

The content of MYWINDOW is erased. Now, enter:

Window **A**ctive **Code** <ENTER>

This activates the Code window. MYWINDOW automatically reduces to its original size.

Load Data Into Another Window

You can view registers in a window other than the Emulation window. Here's how:

Earlier you stored the register content in a file named "reglist." Now that you have activated the Code window, you can view the register content in that window by loading it with the file "reglist." Enter:

```
Window Load  <ENTER>
```

The file name "reglist" already exists in the form from the previous window load. Press:

```
<ENTER>
```

You have done what we described earlier:

1. displayed registers
2. saved the register listing in a file
3. loaded that file into a window other than the Emulation window

Note

You could choose any window for viewing the file by pressing the **Tab** or **Shift -Tab** keys until the window name you want appears in the form. Then press **Enter**. You also could type a window name into the form, then press **Enter**.

The content of the file "reglist" is displayed in the Code window. You cannot see the entire file because the Code window buffer size is too small. To observe the entire file, enter:

```
^z
```

The Code window enlarges to the full screen size. It will stay enlarged until you reduce it, or until you perform an action on another window. Press:

```
^z
```

Notice that the Emulation window still contains the original listing of registers. The Code window should contain two listings of the register contents.

Just for fun, enter:

```
^a
```

Watch the active window deactivate and another window become active. This is a handy feature you can use to quickly cycle through the windows to select each in turn. Continue pressing ^ **a** and watch each window become active when you press it. Notice that the active window will overwrite another window if the areas overlap. Press ^ **a** until MYWINDOW is the active window. Press:

```
^z
```

This returns the PC Interface screen to its original size. All previously displayed windows should appear.

Erase the other windows. Enter:

```
Window Erase Code  <ENTER>
y  <ENTER>
```

The content of the Code window is erased. Enter:

```
Window Erase Emulation  <ENTER>
y  <ENTER>
```

The content of the Emulation window is erased. Now enter:

```
Window Delete  <ENTER>
```

< Tab> until MYWINDOW is selected for deletion. Then enter:

```
y  <ENTER>
```

MYWINDOW should disappear from the screen. Your screen should now appear as if you just started the PC Interface.

Learning Basic Emulation Features

Each HP 64700-Series emulator has a unique way of handling registers, memory, and I/O locations. That is why you have a separate manual for the emulator that describes how to use the emulator with the PC Interface.

You should review the rest of this manual to gain a general understanding of the PC Interface. Perform all the examples and

tutorials in this manual. Then proceed to your Emulator User's Guide. Chapter 6 in this manual covers details on how the PC Interface controls the emulators.

Execute the Command File

Now execute the command file "tutorial."

```
System Command tutorial  <ENTER>
```

Watch your command file execute. You can press ESC to terminate the command file execution. To learn more about command files, see chapter 7.

Exit the PC Interface

To exit the PC Interface, return to the main PC Interface level by pressing < **ESC**> until "System" appears. Choose **S**ystem and **E**xit.

Now, choose **L**ocked to retain the current configuration, **U**nlocked to start the PC Interface later with the reset configuration, or **N**o_Save to start the PC Interface later without saving the current configuration.

See the section titled "How to Exit the PC Interface" in chapter 5 for more information on exit options.

Notes

4

Using Windows

Topics Covered

This chapter shows you how to use the PC Interface windows. The topics covered include:

- Window Functions and Features
- Accessing the Window Functions
- Window Characteristics and Parameters
- Window Tutorial
- How to Use the System Terminal Window
- Printing Window Contents

Window Functions and Features

A window is a view of your input data and measurement results. It is displayed on screen and allows you to observe emulation and analysis operations.

The PC Interface windows allow you to perform multiple operations simultaneously, and help you organize your tasks. You can execute one function in one window and a different function in another window. When you are through with this chapter you should be able to use PC Interface windows to perform emulator and analyzer functions with ease.

There are two types of windows in the PC Interface: system windows and user-defined windows.

Functions of the System Windows

The system windows allow you to:

- Display emulation processor registers (Emulation window).
- Display emulation and target system memory (Emulation window).
- Trace emulator operation with the built-in analyzer (Analysis window).
- Display currently defined breakpoints (Emulation window).
- Display emulation processor I/O locations (Emulation window).

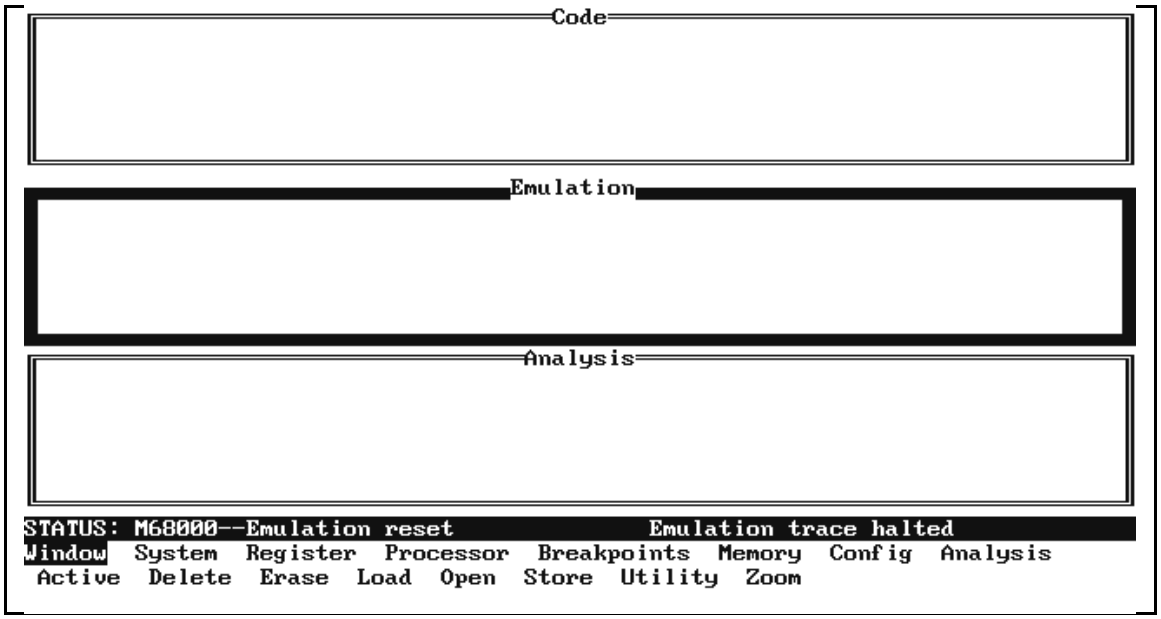


Figure 4-1. PC Interface Window Representations

- Observe status and error messages (Error_Log window).
- Operate your emulator using the Terminal Interface (Terminal window).
- Display your programs (Code and user-defined windows).
- Display symbols associated with your programs (Symbols window).

The system windows and their names are predefined. You cannot accidentally delete any of the system windows. You can "hide" them. You also can change their size and location on the screen.

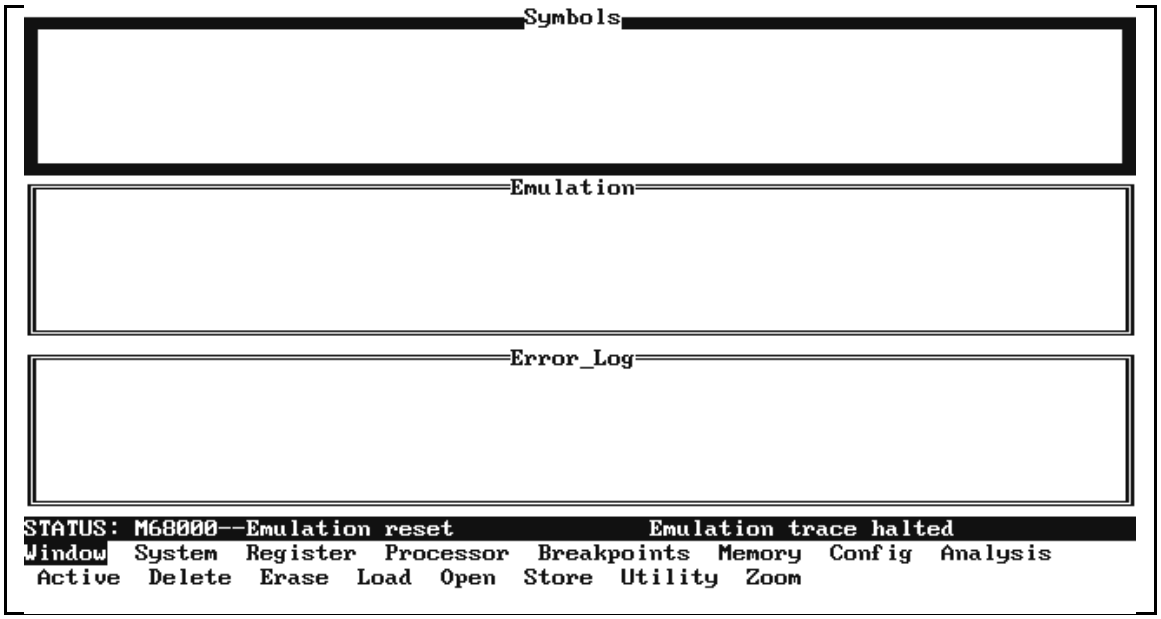


Figure 4-2. More System Window Representations

User-defined Windows

The user-defined windows:

- are created and named by you
- can show the content of files
- initially have default size, location and attributes
- can be deleted

Window Attributes

One of the windows displayed on screen is always active. The currently active window has a solid highlighted border. All window commands act on the currently active window by default. When you specify another window for an operation, that window automatically becomes active.

To browse through a window, you can use these keys:

- up and **down** arrows
- Ctrl-left arrow and **Ctrl-right arrow** to move left and right. If you have more than 78 columns of text, you can scroll to view the additional columns using these keys.
- Home and **End**
- Pg Up and **Pg Dn**
- space bar

All PC Interface windows are surrounded with a double-line border. The border on a "zoomed" window is solid because it is active. The status window (status lines at the bottom of the screen) does not have a visible border.

Note

The PC Interface status line is updated about every 300 milliseconds.

Each window has a memory buffer that contains data sent to it because of a command. You can configure each window's memory buffer size with PC Interface commands. The memory buffer size limit depends on the amount of free memory available on the host computer. This is because the memory buffer is allocated from the computer's pool of free memory.

Window Features

You can manipulate PC Interface windows using the following options:

- activate
- delete
- erase
- load
- open
- store
- color
- hide
- parms
- search
- view
- zoom

Note

You can delete user-defined windows. You cannot delete the system windows.

Accessing the Window Functions

After you have accessed the PC Interface, type **w** to access the "Window" menu.

You also can select the window feature by pressing < ENTER> , because the "Window" label is already highlighted.

Open a Window

Opening a window allows you to create a new user-defined window.

To open a window, enter:

```
Window Open
```

Specify a unique name of 12 characters or less for the window. When you open a window, the top, bottom, left edge, and right edge values appear as default values. If you do not change these values when opening a new window, the size defaults to the full length and width of the screen.

To select the default values, just press < ENTER> for each field, and watch the new window appear after you have completed the form.

A memory buffer is created for each new window opened, and is automatically set to 20 rows. You cannot set the buffer size to anything less than 20. The autoclear feature is turned on. If you are using a color monitor, the default colors are white text on a blue background. For monochrome monitors, the default colors are white text on a black background.

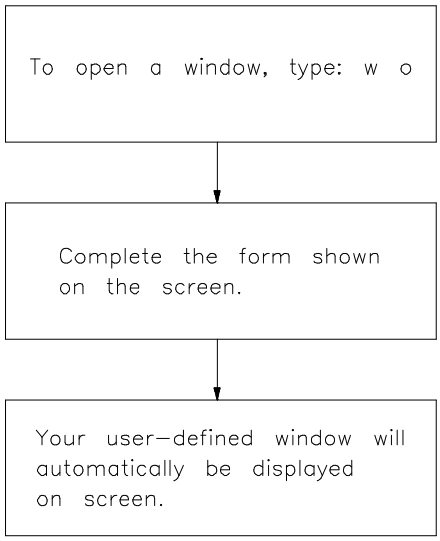


Figure 4-3. How to open a Window

If the size of the window is smaller than the window name, the window name will be truncated on screen. The complete name will appear in the forms.

How Many Windows You Can Create

You can create four user-defined windows.

Note

You should delete any user-defined windows that you are not using because each user-defined window uses memory on your host computer. The amount of memory each window consumes depends on the buffer size you specified, and on the data loaded into the buffer. A buffer of 20 can consume as much as 2 kilobytes of memory.

Delete a User-defined Window

You can delete a user-defined window when you have finished with it or if you opened it by mistake. When you delete a window, the memory it consumed is returned for reuse by the PC Interface. It is not available to the system until you exit.

To delete a window, enter:

```
Window Delete
```

Specify the window name in the form that appears. You are asked to confirm the deletion. If you type anything other than **y**, the window will not be deleted.

Note

If you delete a window without specifying a window name, the currently active user-defined window will be selected for the deletion. Be careful when deleting your windows!

Even if a user-defined window is not visible on screen, you can still delete it.

View a Window

Viewing a window allows you to display a window that was previously hidden with the **hide** command, or one that is hidden by default when you enter the PC Interface. To view a window, you need only specify the window name.

To view a window, enter:

```
Window Utility View
```

and specify the window to view in the form that appears. You can use **Tab** and **Shift Tab** keys to choose the window. Watch the window appear on screen as the view command completes.

Hide a Window

You can hide a window to temporarily remove it from the screen. The window definition remains on the system. When you hide a window, you can still write data to it just as if it is being displayed. If you choose to hide a window, but do not specify a window name, the currently active window will be hidden.

To hide a window, enter:

```
Window Utility Hide
```

and specify the window name in the form that appears. Watch the window disappear from the screen. To verify that the window is hidden, you can use the **W**indow **U**tility **V**iew command. Only hidden windows will be listed, not those which are inactive and merely obscured by other windows.

Erase a Window

Erasing a window removes the contents of that window.

To erase a window, enter:

```
Window Erase
```

Specify the window name. If you choose to erase a window, but do not specify the window name, the currently active window is erased. You are asked to confirm the erasure. If you type anything other than **y**, the window will not be erased. You also can press **^ e** to erase a window, but you are not asked to confirm the erasure.

Note

When a window contains important data, be sure to use the PC Interface **Window Store** feature (described later in this chapter) to save the data in a file before you erase the window. If you have not saved the data, it will be lost when you erase the window.

Activate a Window

Activate a window when you want to perform a function using that window. When a window becomes active, you can use the **up, down, Ctrl-left arrow,** and **Ctrl-right arrow**, with the **Home, End, Pg Up**, and **Pg Dn** keys, and the space bar to view the content of the window. To activate a window, you need only to specify the window name. You also can press **^ a** to change the currently active window.

To activate a window, enter:

```
Window Active
```

and specify the window name in the form that appears. Watch the window's border change to a solid line as you activate it.

Load a Window

You can load an ASCII file into a window. This allows you to view your file while running other commands.

To load a file into a window, enter:

```
Window Load
```

Specify the window name, the file name and any applicable directory information. If you choose to load a window, but do not specify a window name, the active window will be loaded. If the window was inactive, it will become active when you start the load process. Files with a length exceeding the window buffer size will be truncated, and a message will be displayed indicating "File is too big for window memory buffer...Lines may be truncated."

Zoom a Window

You can zoom a window to expand its boundaries to the full screen.

To zoom a window, enter:

```
Window Zoom
```

Specify the window name in the form that appears. When you want to reduce a zoomed window to its original size, just choose zoom again. If you choose to zoom a window, but do not specify a window name, the currently active window will be zoomed. You also can press **^ z** to zoom the currently active window.

Store a Window

You can store part or all of a window's contents in an ASCII file.

To store a window, enter:

```
Window Store
```

Specify the window name, the range of lines to store, and the destination file where the window content will be stored in the form that appears.

If you do not specify the window name, the active window will be stored.

Specify a range of line numbers. The value in the "From line" field gives the line number of the current cursor location in the window. You can use the up and down arrows to position the cursor on any line in the window. To return to the form, just press either the **right** or **left arrow**, or press < ENTER> . The default is to store all lines in the window.

You can include any valid directory information if you want to store the file in another directory. You also can add an extension to the file (for example yourfile.ext).

Note

The PC Interface will not let you store an empty window into a file.

You can use the **System MS-DOS Fork** feature to verify that the window content was stored. Chapter 3 tells how to access the MS-DOS system level with the "fork" feature.

Search a Window

The window search feature allows you to locate the first occurrence of a string within the specified window. To find a user-defined

string, specify the window name, string to be found, and a range of line numbers to be searched.

To search a window, enter:

```
Window Utility Search
```

and specify the window to be searched, the range of lines in which to search for the string, and the string to be located. The cursor will then be placed immediately after the string, and the string will be displayed at the top of the window.

Summary of Window Control Characters

Keystroke	Action
^ a	Activates the currently active window.
^ e	Erases the currently active window.
^ z	Zooms the currently active window.

Window Characteristics and Parameters

The Cursor Location is Retained

When you work within multiple windows, the PC Interface retains the cursor location in each of those windows. The PC Interface will retain the cursor location for each window no matter how often you switch between the windows, or even if you remove a window from the screen (hide it) and then redisplay (view) it.

About the Window Characteristics

All the PC Interface windows have characteristics of color, size, name, autoclear and buffer size. You can change these characteristics to suit your needs by using the **W**indow **P**arameter command, and the **W**indow **U**tility **C**olor command.

Before you modify a window's characteristics, you must specify that window's name. Otherwise, the active window is modified.

Color

You can set the foreground and background colors. Foreground color applies to the text in a window. Background color applies to the area on which the text is displayed. When you use a color monitor, the available foreground and background colors are black, blue, green, cyan, red, magenta and white. This applies to all the windows (user-defined and system windows). The default colors (for a color monitor) are white text on blue background.

Changing colors affects all the windows. You cannot assign different colors to individual windows.

Note 

The PC Interface will not let you set the foreground and background colors the same.

Size

Using the size parameter, you can determine the size and location of each window on the screen. To change the size and/or location of a window, specify new values for the window's length and height. The length can be defined within the range of 0 to 79 columns. The height can be defined within the range of 0 to 20 rows. The windows must be located in the first 20 rows, because the bottom 4 rows are reserved for status messages and command entry. The largest possible window can be defined with all 80 columns and all 20 rows.

The smallest possible window size is 5 columns by 5 rows. This is because the window must have a name. With a 5 x 5 window, there is enough room for a name that has a single letter.

Name

Each window name is an ASCII string of 12 alphanumeric characters or less. The window name appears in the upper left

center of the window. You cannot change any of the window names. But, when you create a user-defined window, you do assign a name to it. Every new window must have a unique name. If you try to duplicate a window name, the message “Another window exists by this name” will appear at the bottom of the screen. Then you must select a unique name.

If you specify a window name that is larger than the window itself, the name will be truncated. When you zoom the window, the entire name will appear. Also, when you perform an action on that window using a form, the entire window name will appear in the form when you press the **Tab** key. If you plan to type in the window name, you must specify the entire name, or the PC Interface will not accept the name.

Autoclear

This parameter determines whether each window should be cleared before writing data to it. With this parameter set to “y,” new data overwrites previous data displayed in that window.

With this parameter set to “n,” each time you display data in that window, it is appended to the previous display until the window buffer fills. The data then scrolls upward. You cannot recover any data on lines that are scrolled off the top of the window memory buffer.

Buffer Size

The window buffer size is defined by a number of rows. The buffer, which is allocated from the pool of free memory on the host computer, stores the data displayed by the window. You can change the buffer size of any of the windows. Whenever you change a window’s buffer size, all data currently stored in that buffer is lost. A buffer size of 20 can consume as much as 2 kilobytes of memory.

The minimum buffer size is 20. The maximum buffer size is 1030.

Display

This parameter determines whether the window should be viewed or hidden. With the parameter set to “y,” the window is visible after ending the command. When set to “n,” the window is hidden

and can only be seen when made visible with the **W**indow **U**tility **V**iew command.

The display parameter is only available when you are defining a new window. It is not available through the Window Utility Parameters menu. If you want to view a window, use **W**indow **U**tiltity **V**iew. To hide it, use **W**indow **U**tility **H**ide.

Scroll

This parameter determines the scrolling method for the window. It is only available through the **W**indow **U**tiltity **P**arameters menu.

If you choose "y," data is written to the window one line at a time as it is received from the emulator or the host system. This can be useful when you want to display large source files or trace listings.

If you choose "n," the window is not updated until the PC Interface has all the requested data available. All the information is written in a single operation. This method is faster overall, but you may have a noticeable wait period while the PC Interface collects the data.

Window Tutorial

You may want to define windows for your use. These can be useful for reviewing your programs, and viewing command or configuration files. You probably can think of other ways to use them. This tutorial shows you how to create and delete windows.

Follow the example, using your computer and the PC Interface to create a window.

1. From the main level enter **W**indow then **O**pen. A form will be displayed at the bottom of the screen:

```
Window Open ____________ Top Row 0__ Bottom Row 20_ Autoclear? y
Buffer size 20         Left edge 0   Right edge 79  Display? y
```

2. Specify a name for the **W**indow **O**pen process by typing **MYWINDOW**. You can use upper and lower case letters in the name. They are matched in the window display.

3. Use the **right arrow** or < ENTER> key to move the cursor to the next field.

4. Specify a row number for the location of the top of the window by typing **0**.

5. Move the cursor to the "Bottom Row" field and specify the bottom location for the window by typing **20**. This value cannot be larger than 20.

6. Move the cursor to the "Autoclear" field. The default is "y"es to clear the window. Let's leave it as is. Press < ENTER> . Each time you display data in this window, old data will be overwritten. (To change the field to append to any data that was previously stored, you would just type **n**.)

Note

Whenever you encounter a "yes" or "no" option within a form, you can type an uppercase letter if you prefer.

7. Specify a buffer size by typing **100**. This value can be as large as 1030 rows.

8. Specify the window's left edge by typing **40**. This value can be from 0 to 79.

9. Specify the window's right edge by typing **79**. This value can be from 0 to 79.

10. Move the cursor to the "Display" field. The default is "y"es. Let's leave it as is. Press < ENTER> . The window will be displayed after you create it. (To create a window, but not display it, you would just type **n**. Then you would use **W**indow **U**tility **V**iew to make it visible.)

Your window should now be displayed on the right half of the screen. You can see that it's yours because the name you assigned is at the top (MYWINDOW).

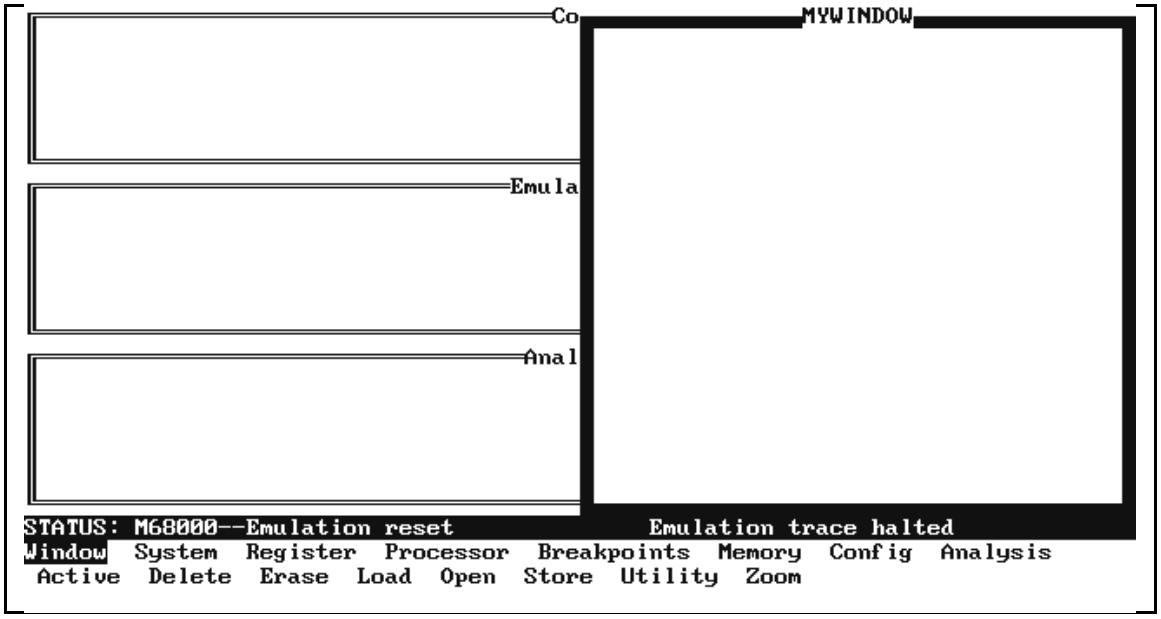


Figure 4-4. Your User-Defined Window (MYWINDOW)

Delete the Window

To delete the user-defined window you just created (named MYWINDOW):

1. Make sure you are at the main PC Interface level. If not, press the ESC key until the main PC Interface labels appear.

2. Type **w** then **d**.

3. The form at the bottom of the screen should display "MYWINDOW." If you created any additional windows in previous steps, the name of the last window you created will be displayed in the form. Press < ENTER> if the name of the window you want to delete appears in the form. Otherwise, press Tab until the name of the window you want to delete appears in the form, then press < ENTER> .

4. You are asked to confirm the deletion. Type **y** and press < ENTER> again to delete MYWINDOW. (If you don't want to delete the window, type n (which is the default answer.)

Note

You can scroll through the list of possible windows to delete by pressing the **Tab** and **Shift Tab** keys. If there is only one user-defined window, the name in the form will not change when you press **Tab** or **Shift Tab**.

If you haven't created any user-defined windows, you will not be able to delete any windows. If you try it anyway, the PC Interface will return to the main level, and the message "No user-defined windows have been created" will appear.

You can delete a user-defined window even if it isn't visible on the screen.

How to Use the System Terminal Window

To access the System Terminal window:

1. Return to the main PC Interface level by pressing **ESC**.

2. Press **S**ystem **T**erminal.

Notice that the window named Terminal is active. You can tell that it is active because its border is highlighted.

If you press < ENTER> now, you should see an emulator prompt appear at the top of the window. This feature allows you to access and operate your emulator using the Terminal Interface. Don't change the emulator configuration while working in the Terminal Interface. Otherwise, some features may no longer work properly when you exit the System Terminal.

The default size of the System Terminal window is 80 characters wide, and 20 rows deep.

To exit the system terminal window, press < CTRL> \.

The Terminal window has a 48 line buffer. When you exit the Terminal window, all information stored in that buffer is retained. You can then reenter the Terminal window, and find the same information displayed. In addition, if you "hide" the Terminal window, and then view it, all information stored in the window buffer is retained.

Refer to the *Terminal Interface Reference* for information on Terminal Interface commands.

With the System Terminal window displayed, your screen should resemble:

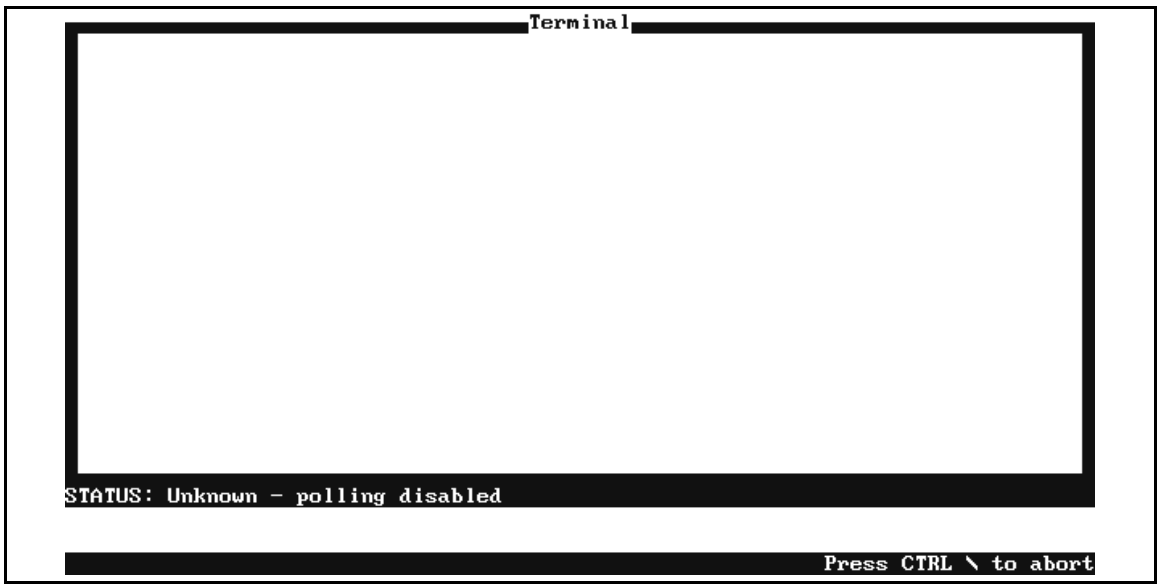


Figure 4-5. System Terminal Window displayed

Note

In the System Terminal window, you can use ^ **r** to recall commands from the previous command. You also can use ^ **b** to recall commands from the first command you entered. To recall multiple commands sequentially, continue pressing either ^ **r** or ^ **b**.

Commands You Shouldn't Execute

While working in the Terminal window, don't execute the following commands, because emulator operation may be suspended:

```
<CTRL> s followed by <CTRL> \
po
stty
```

```
init -pv
xp
echo
mac
wait
pv
t
```

Specifics About the System Terminal Window

When you start the System Terminal window, everything you type at the keyboard is sent to that window, which corresponds directly to that communication port on the host computer. The PC Interface does not recognize the commands you type, because you are accessing the emulator directly through the Terminal Interface, not the PC Interface.

You can perform most of the functions on the System Terminal window that you perform on the other windows. One thing that you cannot change is whether the System Terminal window is displayed. When you access the System Terminal window, it is automatically displayed. You can then use the **W**indow **U**tility **H**ide feature to hide the System Terminal window.

You can put System Terminal window parameters in a configuration file that you execute when initiating the PC Interface. This file could include details about the size and location of the System Terminal window on the screen.

Printing Window Contents

You can print the contents of a window. To do this, enter:

```
Window Store prn
```

This tells the PC Interface that the file is to go directly to the device named “prn.” If your printer has a different name, specify its name in place of “prn.”

5

Using System Features

Topics Covered

- How to Execute System Level Commands
- Log Commands and Output to a File
- Load and Display Symbols
- How to Store the PC Interface Configuration
- How to Load the PC Interface Configuration
- How to Exit the PC Interface

How to Execute System Level Commands

System level commands are those you typically execute at the MS-DOS system level, like **dir, chkdsk**, and **copy**.

There are two ways to execute system level commands from within the PC Interface. You can use the single command execution feature, or you can temporarily access the system level to execute multiple system level commands. The following paragraphs describe both methods.

Execute a Single System Level Command

This feature allows you to execute a system level command, such as **dir**, by completing a form. The form requires that you type in the single command (chapter 3 describes forms).

You can redirect the output of one command into the input of another command. For example, you could type **dir | more** to view the list of files in the current directory screen by screen, if supported by your PC.

You also could use an editor to modify or create a file. When you save the file, control of the system automatically returns to the PC Interface.

Note

When you use this method, the system level command does not interact with the PC Interface windows. While the command is executing, the PC Interface is temporarily suspended.

With the single system level command, you can redirect the output of the command into a file. For example, you could type **dir > filelist** to store the list of files in the current directory into a file named "filelist." You could include directory information if you want the resulting file stored in another directory.

Note

All single system level commands will act on the current directory, unless you specify otherwise by including directory information.

To access the system level and display all files in the current directory, enter:

```
System MS-DOS Command dir
```

This command informed the PC Interface that an MS-DOS system level command will be executed, and **dir** displayed a listing of the files in the current directory.

To access the system level and format a flexible disk on drive a, enter:

```
System MS-DOS Command format a:
```

You must then insert the flexible disk into the drive, and wait for the format to complete. When you finish formatting diskettes, press any key to return to the PC Interface.

Try executing other system level commands using this method.

Execute multiple System Level Commands

This feature gives you temporary access to the MS-DOS system level. Once you have accessed the system level, you can execute as many commands as you want before returning to the PC Interface.

You cannot execute the command to access the PC Interface. If you forget that you've accessed the system level, and then try to access the PC Interface again, the message "Program too big to fit in memory" will appear.

To access the MS-DOS system level, enter:

```
System MS-DOS Fork
```

This indicated to the PC Interface that you temporarily transferred control to the MS-DOS host operating system. At this point you can use your host computer for many typical system functions.

Try this feature. Then, when you are ready, enter:

```
exit
```

Now press < ENTER> to return to the PC Interface.

Note

Do not run any programs that modify the RS-232 ports. Also, you should not run any programs that consume memory and without returning it to the host computer. Some examples include: operating HP AdvanceLink, starting a LAN (Local Area Network) connection, and copying files from another computer system. Also, you will not be able to execute any commands that require large amounts of system memory, such as compilers, complex text editors, and so on. You can use the MS-DOS CHKDSK command to determine the amount of free memory available after you fork the system.

Log Commands and Output to a File

The PC Interface "log" feature allows you to store commands and/or the results of those commands in a file. You can use it later as a command file to perform tasks automatically.

Command and log files are PC Interface system features. Chapter 7 describes how to create and use them.

Load and Display Symbols

The PC Interface allows you to access a symbol database on the host computer. Using this feature, you can connect to a symbol database, display global symbols, and display local symbols. For each of these processes, a form will appear on screen. You must make your choices in this form.

To load a symbol database, enter:

```
System Symbols Global Load <filename.L>
```

The ".L" suffix specifies a linker symbol file produced by an HP 64000 software development tool. The proper filename for you tools depends on the file formats you use. See the appendices of your ***Emulator PC Interface User's Guide*** for more information.

To display global symbols in the database you just loaded, enter:

```
System Symbols Global Display
```

The global symbols in the database will automatically be displayed in the Symbols window.

Note

You must load a symbol database before displaying symbols. Loading a program module (using the **M**emory **L**oad command) automatically loads the symbol database for most file formats. Refer to your *Emulator PC Interface User's Guide*.

To display local symbols from the symbol database, enter:

```
System Symbols Local <filename.S>
```

The ".S" suffix specifies a source module for an HP 64000 software developent tool. Again, the filename and extension you need depends on the file formats you use. Refer to the appendices of your *Emulator PC Interface User's Guide* for more information.

Note

The top of the global symbols display lists the modules which can be referenced for local symbols. You can enter any of the names listed when displaying local symbols. Though the MS-DOS operating system is not case sensitive, you must enter these names in the same case as their listing in the global symbols display. This supports situations where code development is done on systems which are case-sensitive, such as HP-UX.

Details About Symbols

Symbol files are created when generating absolute files with the HP 64000-PC Cross Assembler/Linkers. When you assemble a source file, an assembler symbol file (with the same base name as the source file but with an extension of ".a") is created. The assembler symbol file contains local symbol information.

When you link relocatable assembly modules, a linker symbol file with the same base name as the absolute file but with 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. The file reader for the HP64000 format collects the information from the ".L" and ".A" (assembler symbol) files to construct the symbol database. Readers for other file formats may use different methods to build a symbol database.

Global Symbols

When any files supported by a file format reader are loaded into the emulator (using the **M**emory **L**oad command), the PC Interface connects to the corresponding symbol database. You also can connect to the symbol database with the **S**ystem **S**ymbols **G**lobal

Load command. Use this command in situations where you are loading multiple absolute files into the emulator. It allows you to connect to symbol databases corresponding to the various absolute files. When you specify a new symbol database, information contained in a previous symbol database is no longer accessible (only one symbol database can be loaded at a time).

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

Local Symbols

If you display local symbols with the **S**ystem **S**ymbols **L**ocal command, you also can enter local symbols as they appear in the source file or local symbol display.

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

```
module_name:[scope.]*symbol
```

In other words, you specify a module name, followed by zero or more optional scope specifiers (separated by periods), followed by the symbol name. For example, suppose you have a C module named **proj.c**, with a function called **test** which defines a **static int index**. You refer to this variable as:

```
proj.c:test.index
```

Note

The symbol database does not contain information about symbols within functions that refer to variables on the stack.

You also can specify ranges using symbols using the syntax:

```
<lower_bound>..<upper_bound>
```

Suppose you had a table of messages defined in your program and wanted to display the memory where the messages were stored.

```
Memory Display Byte
cmd_rds.S:Cmd_A..cmd_rds.S:Cmd_I
```

This also works across modules, as long as both modules referenced are part of the same symbol database (that is, both module names are listed when you display global symbols). For example, suppose **Cmd_I** is in **inv_proc.S**, which links with **cmd_rds.S** to form the program whose symbol database is loaded:

```
Memory Display Bytes
cmd_rds.S:Cmd_A..inv_proc.S:Cmd_I
```

When you include the name of a local symbol module with a local symbol, that module becomes the default local symbol module, as with the **S**ystem **S**ymbols **L**ocal command. 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 module exists and contains information).

The only valid module names are those listed in the current global symbols display. The module names are displayed at the beginning of the global symbols list. You must match the case shown in the global symbols listing when entering module names. For example, if the global symbols list shows a module named CMD_RDR.S, then cmd_rdr.s and Cmd_Rdr.S will be rejected as module names. Only CMD_RDR.S will be accepted.

Symbols defined as local and global

It is possible for a symbol to be local in one module and global in another, which may result in some confusion. For example, suppose symbol "XYZ" is defined as global in module A and 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 also refer to a global symbol explicitly using the syntax

```
:<symbol_name>
```

For example,

```
:XYZ
```

would refer to a global symbol with the name "XYZ", even if a local symbol exists with that name.

Emulator Symbol Capabilities

Certain emulators can internally store and reference local and global symbols. If your emulator has this capability, you will see additional options appear in the **S**ystem **S**ymbols **L**ocal and **S**ystem **S**ymbols **G**lobal commands.

Symbol handling within the emulator provides additional PC interface features. If you transfer global and local symbols to the emulator, you will see symbol displays in the trace, memory mnemonic, and single step displays.

Global Symbol Options

There are two additional options for global symbols.

```
System Symbols Global Transfer
```

This command moves the global symbol data for the specified absolute file into the emulator, where it can be used by the emulator's symbol handler.

```
System Symbols Global Remove
```

This command removes any global symbol data loaded into the emulator.

Local Symbol Options

There are additional options for local symbols.

```
System Symbols Local Transfer Group
  <module, module, ...>
```

This command moves the local symbol data for the specified modules into the emulator, where it can be used by the emulator's symbol handling tools. To transfer the local symbol data for all modules in the current symbol database, enter:

```
System Symbols Local Transfer All
```

The command

```
System Symbols Local Remove Group
  <module, module, ...>
```

removes any local symbol data loaded into the emulator for the specified modules only. To remove all local symbol data in the emulator, enter:

```
System Symbols Local Remove All
```

To find out which local symbols are loaded into the emulator, enter:

```
System Symbols Local Loaded
```

The PC Interface returns the module name(s) associated with that symbol set.

How to Store the PC Interface Configuration

The PC Interface configuration feature allows you to store and/or load all the PC Interface configuration data into a file that you can reload later.

The configuration store feature saves:

- miscellaneous information
- key macro definitions
- analyzer trace format and specification
- system and user-defined window information
- emulator-specific information

PC Interface Configuration File Details

The monitor color information is stored in the PC Interface configuration file, under the topic of “miscellaneous.”

An example of the monitor section of a PC Interface configuration file resembles:

```
$MISC
color white blue
$MISC
```

If you are using a monochrome monitor, the term “color” is replaced by “mono.”

Window Configuration Information

All the system and user-defined window characteristics are stored in the PC Interface configuration file.

Note

Data displayed in any of the windows is not stored in the PC Interface configuration file.

When loading the PC Interface configuration file, windows and their characteristics will be redisplayed as they were when the configuration was stored.

An example section of a PC Interface configuration file used to define windows resembles:

```
$SYSWIN
Code F F F 0,0 6,60 20
Symbols T T F 0,61 15,79 20
Trace T T F 16,0 20,79 20 T
Error_Log F F F 16,0 20,79 20
Emulation T F F 7,0 15,60 20
Terminal F F F 0,0 15,79 20
$SYSWIN
$USERWIN
MYWINDOW T F F 0,40 20,79 20
$USERWIN
```

System window characteristics are enclosed with $SYSWIN. User-defined window characteristics are enclosed with $USERWIN.

The user-defined window characteristics are:

Parameter	Meaning
MYWINDOW	user-defined window name
T	display the window after it is created (True)
F	clear the window before writing data to it (False)
0,40	Top,Left margins
20,79	Bottom,Right margins
20	Window buffer size

The system window characteristics also include these parameters.

Emulator-Specific Information

The emulator-specific characteristics stored in the PC Interface configuration file include:

- emulator display and access modes
- general emulator configuration
- break conditions
- emulator memory map terms
- CMB setup
- BNC trigger setup
- CMB trigger setup

Note

You can use the System Terminal feature to access the Terminal Interface to modify the emulator-specific configuration. When you return to the PC Interface, the new configuration parameters are in effect. If you change the emulator configuration in the Terminal Interface, some emulator features that worked before may not work properly when you exit the System Terminal window.

An example emulation section of a PC Interface configuration file (for the Z80 emulator):

```
$EMUL
mo -ab -db
cf clk=int
cf rrt=dis
cf qbrk=en
cf trfsh=dis
cf tbusack=dis
cf busreq=en
cf int=en
cf nmi=en
cf waitem=en
cf wrdata=dis
cf moncyc=dis
cf monbase=0000
bc -d bp #disable
bc -e rom #enable
bc -d bnct #disable
bc -d cmbt #disable
bc -d trig1 #disable
bc -d trig2 #disable
map  00000..07fff  eram   # term 1
map  08000..08fff  erom   # term 2
map  other eram
cmb -d  #cmb currently disabled
bnct -d none -r none
cmbt -d none -r none
tgout none
tarm always
$EMUL
```

Notice that all the emulation configuration items are included. Break conditions, the memory map, CMB, and trigger details are also included.

How to Load the PC Interface Configuration

You can load the PC Interface configuration file in two ways:

1. You can specify the configuration file name (with the **/w** option) when accessing the PC Interface.

2. You can use the **C**onfig **L**oad command from within the PC Interface.

You can modify an existing configuration file using an editor, then reload the configuration file. The parameters in the new version of the configuration file will immediately take effect.

You can change all or part of the PC Interface program configuration by loading a complete configuration file, and then loading subsequent partial configuration files.

Example Configuration Files

Suppose you have several configuration files, organized like this:

Configuration file # 1, named "entire.cfg," contains all the characteristics from a previous configuration.

Configuration file # 2, named "window.cfg," contains only user-defined window characteristics. You could have created this file manually, or you could have edited a previous configuration file.

Configuration file # 3, named "emul.cfg," contains all the characteristics for your emulator configuration.

Here's how you could use these files:

First, you could load the complete configuration file "entire.cfg" by specifying the file name when accessing the PC Interface. Next, you could automatically recreate any user-defined windows by loading the file "window.cfg" with the "**C**onfig **L**oad" command. Then, to perform specific tasks with the emulator, you could use the "**C**onfig **L**oad" command to load the configuration file "emul.cfg." This file may contain characteristics to change your emulator configuration to operate with a target system.

Using configuration files this manner can help you initialize your emulation system quickly and completely to perform a wide range of tasks.

How to Exit the PC Interface

There are three methods for exiting the PC Interface and returning to the host computer operating system:

To exit the PC Interface and keep it locked, enter:

```
System Exit Lock
```

To exit the PC Interface and leave it unlocked, enter:

```
System Exit Unlock
```

To exit the PC Interface without saving the current configuration, enter:

```
System Exit No_save
```

The following table summarizes these exit choices and their effects:

Action	System Exit		
	Locked	**No_save**	**Unlocked**
Saves current configuration?	Yes	No	No
Initialize emulator on next PC Interface entry?	No	No	Yes
Load emulation configuration items ($EMUL) on next PC Interface entry?	No	No	Yes

When you reenter the PC Interface, it always loads any configuration file specified with the /w command line parameter. (Although it may ignore certain configuration file parameters; see the table above.) If none is given, the default configuration file is loaded (if it exists). Otherwise, the emulator configuration is determined by the PC Interface defaults.

If you exit with the Unlocked option, the emulator is reinitialized to its powerup state the next time you enter the PC Interface. Then, any existing configuration files are loaded. The interface will take a little longer to start up, since it must wait for the initialization to complete.

Note

When executing a command file, the PC Interface will exit (return control to the host computer) only if the last command in the file is "**s e l**" (System Exit Locked), "**s e u**" (System Exit Unlock), or "**s e n**" (System Exit No_save).

Note

The PC Interface allows you to execute a command file to perform these functions for you. When restarting the PC Interface, you can execute a command file to reconfigure your system as it was previously configured. Chapter 7 contains the details.

6

Controlling Emulators

Topics Covered

This chapter explains how the PC Interface controls the HP 64700-Series emulator. The topics include:

- Modifying the General Emulator Configuration
- Microprocessor Execution
- The Memory Mapper
- Emulation Memory
- Break Events
- Emulator Registers
- Using the Emulator While Connected to a Target System

Continue with this chapter to learn how the PC Interface does this. The following pages describe each of these major topics and their subtopics.

Note

For additional information on using HP 64700-Series emulators with the PC Interface, refer to your *Emulator PC Interface User's Guide* or the *HP 64700 Emulator Terminal Interface User's Guide*. If you have questions about terms in this chapter, refer to the *HP 64700 Emulators Glossary Of Terms*.

Modify the General Emulator Configuration

Each HP 64700-Series emulator has its own specific configuration information. Refer to your *Emulator PC Interface User's Guide* for details about modifying your emulator general configuration.

Microprocessor Execution

Microprocessor features controlled by the PC Interface are:

- Run
- Break
- Reset
- I/O Display/Modify *
- Memory Display/Modify
- CMB Execution
- Single Step

* Refer to the appropriate *Emulator User's Guide* for details.

Run the Microprocessor

The **run** commands cause the emulator to execute from either:

- an address in emulation or target system memory
- the current program counter value
- the reset address of the emulator or target system

If an error occurs, the "Error_Log" window displays an appropriate error message. For example, if your program encounters a software breakpoint, the result is displayed in the Error_Log window.

Otherwise, the run commands do not have any output location for data. When you execute any of the run commands, the status line displays the current state of the emulator.

Let's suppose your emulator has a short program in emulation memory, starting at address 0.

1. To run from the start of this program, enter:

```
Processor Go Address
```

   A form appears for you to enter the address from which the processor should begin executing your program. Enter the starting address of your program. Notice that the status line shows a user program is running.

Note

The emulation processor might halt if there is no valid code to execute.

2. To run from the current program counter value, enter:

```
Processor Go PC
```

   Notice the status line says "Running user program."

3. To run the emulation processor from target system reset, enter:

```
Processor Go Reset
```

   Now you must reset the target system microprocessor. If you have a hard reset button in your target system, after you press it the target system program will continue to execute. (If your target system does not have a hard reset switch, the target program execution will continue when the emulation processor is reset.)

A key feature of HP 64700-Series emulators is the Coordinated Measurement Bus (CMB). You can use this to make synchronized measurements among multiple emulators/analyzers. To do this, you specify the CMB trigger setup using the "Config Trigger" menu, then instruct the emulation processor to execute on CMB Trigger. This is explained later in this chapter.

Step the Microprocessor

There are many ways you can single-step the emulation processor.

You can control the events which occur at the end of the single-step. You also control the data that is displayed. You can specify the number of instructions you want the microprocessor to step. The default is to step one instruction. The maximum number of instructions you can specify is 99.

Register contents are automatically displayed in the Emulation window. If any errors occur, the results will be displayed in the Error_Log window.

1. To step the emulation processor four instructions from the current program counter address, enter:

```
Processor Step PC 4
```

Note

When stepping the microprocessor, registers are automatically displayed in the Emulation window.

2. To step the emulation processor from a specified address, enter:

```
Processor Step Address
```

   A form prompts you to enter the number of instructions to single-step, and an address from which the emulation processor should begin stepping. Go ahead and enter a desired number (no larger than 99). Then enter the address from which the processor should begin stepping your program.

3. To do a conditional single-step, enter:

```
Processor Step Event
```

   A form lets you specify whether register contents and mnemonics should be displayed. Then you may specify the name of a command file to execute the actions you want when the step completes.

Break the Microprocessor

The break feature allows you to redirect the emulation processor execution from user program execution to the emulation monitor. There are no options to this command.

Note

If your HP 64700-Series emulator is part of a CMB measurement, the break command causes a break in all other HP 64700-Series emulators in the measurement.

To break your emulator into the monitor, enter:

```
Processor Break
```

The processor suspends execution of your program, and begins executing in the monitor. Notice that the status line displays the emulator's current state.

Reset the Microprocessor

The **reset** command stops execution of the current instruction and either begins execution in the monitor or remains reset, depending on which option you choose. No information is saved through the reset.

After the reset, the emulator's program counter is set to the powerup reset address. To run the emulator from the program counter value now would be the same as running from a powerup condition. Notice that the status line displays the emulator's current state.

1. To reset your emulator and start monitor execution, enter:

```
Processor Reset Monitor
```

2. To reset and hold your emulator in the reset state, enter:

```
Processor Reset Hold
```

The emulation processor is reset, and will remain in the reset state until you cause it to do something else. Notice the status line indicates "Emulation reset."

Note

To release the emulator from the halted state, enter **P**rocessor **B**reak at the main PC Interface level. The emulator will begin executing in the monitor.

Display I/O Port Addresses

The **I**/O **D**isplay command allows you to display the contents of the emulation processor or target system microprocessor I/O port addresses.

Note

This command is only valid for emulation processors that have separate I/O and memory address spaces. Refer to your *Emulator User's Guide* for details about your emulator's capabilities.

You can display emulation processor I/O port addresses by specifying the addresses either symbolically or with absolute values by entering your choices in the form that appears.

When you display I/O addresses, a momentary break of foreground execution occurs. After the display is complete, foreground execution resumes.

The Emulation window contains the results of this command. You may have to use the **W**indow **Z**oom command to see the entire window content.

To display your emulator's I/O port address 0, enter:

```
Processor I/O Display 0
```

The result in the I/O window is:

```
Address         Value
-------         -----
0000H      =    FF
```

The data in your emulation processor port addresses may be different that the values shown here.

Note

You can enter multiple I/O port addresses by separating them with a semicolon (;).

Modify I/O Port Addresses

The **I**/O **M**odify command allows you to modify data at emulator I/O port addresses.

Note

This command is only valid for emulators that have separate I/O and memory address spaces. Refer to your *Emulator User's Guide* for details about your emulator's I/O capabilities.

You can specify the I/O addresses either symbolically or with absolute values.

When you modify I/O addresses, a momentary break of foreground execution occurs. After the modify is complete, foreground execution resumes.

To modify data at your emulator's I/O port address 0 to "1," enter:

```
Processor I/O Modify 0=1
```

You indicated that data at I/O port address 0 would be changed to "1." With this command, you must equate a port address to data that you want stored at that address.

You can write multiple data values to the same I/O address by equating the address to multiple terms separated by a comma (,). For example, you could enter:

```
Processor I/O Modify 0=1,2,3,4
```

I/O port address 0 would then contain 1, then 2, then 3, and then 4.

You can also enter ASCII strings. To send the null-terminated string "ABCD" to I/O port zero, enter:

```
Processor I/O Modify 0="ABCD",0
```

Start the Emulator When CMB Events Occur

The **P**rocessor **C**MB command asserts the Coordinated Measurement Bus (CMB) EXECUTE signal.

The CMB synchronizes starts and stops among multiple HP 64700-Series emulators. You can enable CMB interaction through the emulation configuration command. Each emulator has a unique CMB interaction question in the emulator configuration. When you enable CMB, you can use the **P**rocessor **C**MB **G**o command to start the emulators executing at a specified address when the CMB EXECUTE signal becomes valid. All HP 64700-Series emulators waiting for the CMB EXECUTE signal will start executing when the CMB EXECUTE signal goes true.

A line above the status line shows "ALERT: CMB execute; run started." The same result also will be displayed in the Error_Log window.

Note

Refer to the *CMB User's Guide* for details on CMB Operation.

To run the emulators from the current PC when CMB trigger goes valid, enter:

```
Processor CMB Go PC
```

The processor begins executing your program from the current program counter when CMB trigger is driven. Notice that after you enter this command, the emulation processor will not begin executing until you issue a **P**rocessor **C**MB **E**xecute command on one of the emulators participating in the measurement.

To run the emulators from a specific address when the CMB trigger goes valid, enter:

```
Processor CMB Go Address
```

The processor begins executing your program from the address you specified (required parameter) when CMB trigger is driven. Notice that after you enter this command, the emulation processor will

not begin executing until you issue a **P**rocessor **C**MB **E**xecute command.

To begin execution in all emulators connected to the CMB, enter:

```
Processor CMB Execute
```

All emulators connected to the CMB will begin executing in foreground once the CMB trigger signal becomes valid. Notice that the message "ALERT: CMB execute; run started" is displayed above the status line. The same information is also displayed in the Error_Log window.

The Memory Mapper

Each HP 64700-Series emulator provides emulation memory that the emulation processor can use. The emulator must control access to this memory and the memory in your target system. The HP 64700-Series emulators use a memory mapper to divide the microprocessor's logical address space into regions of memory that reside either in your target system, in the emulator, or as guarded (inaccessible) space. Target system memory also can be mapped.

You can program your emulator's memory mapper to resemble the memory configuration of your target system by specifying individual address ranges and memory types. The number of memory mapper terms you can define varies between emulator types. Refer to your *Emulator User's Guide* for details about your emulator's memory mapper.

The PC Interface allows you to modify and/or reset the emulation memory mapper features.

Modify the Memory Mapper

This feature allows you to modify the memory map terms. Memory map terms are definitions that you specify in the memory map menu.

Note

Each emulator restricts the blocksizes you can define to some value, such as 256, 512 or 1024 bytes. For example, the Z80 emulator requires that memory ranges you define be specified in multiples of 256-byte blocks. If you specify a range that is not a multiple of 256-byte blocks, the memory map automatically adjusts to make it a multiple of 256-byte blocks.

Emulators also restrict the starting range of blocks to a multiple of the blocksize value.

Refer to your *Emulator User's Guide* for details about your emulator's memory mapper.

Each memory map term has an address range and a memory type. The memory types are:

- eram (emulator RAM)
- erom (emulator ROM)
- tram (target system RAM)
- trom (target system ROM)
- grd (guarded/inaccessible)

When modifying the memory map configuration, you must type your choices in the form that appears. You must specify the address range for each term that you add or modify.

To modify the memory map, enter:

```
Config Map Modify
```

The memory map will automatically be displayed. It should resemble:

```
Memory Map Configuration
Unmapped memory type eram
Term       Address Range          Memory Type
1                                 Empty  grd
2                                 Empty  grd
3Empty                            grd
4Empty                            grd
5Empty                            grd
6Empty                            grd
7Empty                            grd
8Empty                            grd
9Empty                            grd
10Empty                           grd
11Empty                           grd
12Empty                           grd
13Empty                           grd
14Empty                           grd
15Empty                           grd
16Empty                           grd
```

Figure 6-1. Default Memory Map

You enter mapper terms by typing in the address range you want to map, and selecting the memory type. For example, to define a new term 1, you might type **0..3ffh** and use **Tab** to select erom in the Memory Type field. A shorthand entry method is to type only the starting address of the range, followed by two periods (..). This gives you one block.

Note

The Terminal Interface command **map addr..addr+ 7fh < type>** is sent to the emulator when you use this shorthand method. If this specification spans more than one block in the mapper, the mapper will allocate two memory blocks. Suppose your emulator maps memory in 1 Kbyte blocks. If you specify a memory block as **1000h..**, the emulator will map a block of memory from 1000h to 13ffh. But, if you specify the block as **13f0h..**, the emulator will map a block of memory from 1000h thru 17ffh (effectively two contiguous blocks).

When you have finished editing the memory mapper terms and exit the form, the new memory mapper configuration is in effect.

To learn more about defining ranges of memory for your emulator as either eram, erom, tram, trom, or grd, refer to your *Emulator PC Interface User's Guide.*

Storing the Memory Map

You can save the current memory map in a file using the PC Interface **C**onfig **S**tore command. If you want to load that memory map later, use the **C**onfig **L**oad command.

Reset the Memory Map

This feature allows you to delete all current memory map term definitions. All of emulation memory will become the default memory type. Refer to your *Emulator PC Interface User's Guide* for details about the default memory type for your emulator.

To reset the emulation memory map, enter:

```
Config Map Reset
```

Any previously defined memory map terms are deleted, and the memory map is set to the default state.

Note

Be sure that you want to reset the memory map before using this command. If you are not sure that you want to delete all the terms, first store the PC Interface configuration to a file. Then you can retrieve any of the terms you need without having to recreate them manually.

Controlling Memory

The PC Interface controls these emulation memory and target system memory features:

- Display
- Modify
- Load
- Store
- Copy
- Find

Display Memory

This feature allows you to display the emulation processor address space.

When the microprocessor is executing in foreground, and you display memory addresses that map to your target system, a momentary break of foreground execution occurs. After the memory display is complete, foreground execution continues.

The Emulation window contains the output from all memory display commands.

To display memory, enter:

```
Memory Display
```

Now you must specify how you want the memory display formatted. Depending on your emulator's capabilities, you may specify the data associated with these ranges in bytes, words, mnemonics, long words, or various floating point types. Refer to your *Emulator PC Interface User's Guide* for details.

A form will appear asking you to enter one or more address ranges. You can specify as many address locations or ranges as the form will allow by separating each range with a semicolon (;).

Note 

To display multiple memory ranges with a single command, separate the ranges with a semicolon (;).

To view the entire content of the Memory window, use the **Home** and **End** keys. To save more lines in the window, increase the window's buffer size using the **W**indow **U**tility **P**arameter command.

You can use the **M**emory **D**isplay **R**epetitively command to constantly update the memory display. The display format and range is that selected for the last **M**emory **D**isplay command. Use the **Esc** key to terminate the command.

Modify Memory

This feature allows you to modify the values of your emulation processor memory or target system memory locations. You can specify as many address locations or ranges as the form will allow by separating each range with a semicolon (;).

When the microprocessor is executing in foreground, and you modify memory addresses that map to your target system, a momentary break of foreground execution occurs. After the memory modify is complete, foreground execution continues.

To modify multiple memory ranges with a single command, separate the ranges with a semicolon (;).

Note

Depending on your emulator's capabilities, you may specify the data associated with these ranges in bytes, words, mnemonics, long words, or various floating point types. Refer to your *Emulator PC Interface User's Guide* for details.

Store Memory to a File

The memory store feature allows you to copy a block of emulation memory or target system memory to an absolute file on your host computer.

To store memory to a file, enter your choices in the form that appears on screen. Specify the file format, the address range to store, and the file name where the data will be stored. You can include any valid directory name if you want to store the data in a file in another directory.

The output from this command is the absolute file. If there are unmapped addresses in the specified address range, the command will terminate when the process encounters an unmapped address. Whenever an error occurs, no absolute file is created.

To store memory to a file, enter:

```
Memory Store
```

Now you must specify a name for the file, the absolute file type, and an address range.

Note

This command will automatically overwrite any existing file with the specified name. Therefore, be careful that you don't destroy valuable information by storing the data to an existing file.

Load Memory from a File

This feature allows you to load an absolute file into emulation memory or target system memory.

The command results are not automatically displayed in any window. You must inspect memory with the **M**emory **D**isplay

command to display the program. If an error occurs, the error is displayed in the Error_Log window, and the process is terminated. Any memory that was loaded before the error will remain altered.

To begin loading emulation memory with an absolute file, enter:

```
Memory Load
```

You need to specify the name of an absolute file and its type. Enter your choices in the form provided on screen. Specify the absolute file (including any applicable path information) and the file's format (HP64000, Intel_Hex, Motorola_Hex, or EXT_Tek_Hex). The HP 64000 format includes an 8-bit binary mode that uses the HP transfer protocol. Then specify whether you want to load addresses mapped as target, emulation, or both.

Each emulator also comes with file format readers that support special file formats for that processor. Refer to the *Emulator PC Interface User's Guide* for your emulator for further information.

The contents of the absolute file will be loaded into the same locations they occupied before you saved them in a file. With the **Both** option specified in the form, any memory mapped as target and emulation memory will be loaded, if both are found in the range you specify to load.

Copy Memory

This command allows you to copy data from one area of memory to another.

To copy a range of memory, enter your choices in the form that appears on screen. Specify the source range (starting and ending addresses) and destination address. The content of the source memory range is then copied to a destination range beginning with the address you specify.

Note

All destination addresses included in the memory copy process must be previously mapped as emulation RAM (eram) or ROM (erom) or target system RAM (tram) or ROM (trom).

To copy a range of emulation memory from 0 through 0ffh to 100h, enter:

```
Memory Copy
```

Now you must specify the source range (0..0ffh) and the destination starting address (100h).

You can verify the copy process by displaying memory locations 100h through 1ffh.

Find a Data Pattern in Memory

This feature allows you to search an address range for the occurrence of a specific data pattern. The data pattern can be 8 bytes or less.

To find a data pattern in emulation memory, enter:

```
Memory Find
```

Now you must specify a memory range to search (0..0fffh, for example), and a data pattern (1,2,3,4,5,6,7,8, for example). Enter your choices in the form that appears on screen. Specify the address range and the data pattern to be located. The pattern can be up to 8 hexadecimal bytes. After you enter the data, the PC Interface searches the address range until all occurrences of the data pattern are located. Then it displays all addresses where the pattern is located.

When the data pattern is located, the address of the pattern's first byte is written to the Emulation window with the pattern. The search then terminates. If the data pattern is not found, the message "Unable to find pattern: < pattern> " is displayed in the Emulation window.

Entering Memory Ranges

You can use expressions in memory ranges, including symbols (assuming that global or local symbols are loaded). For example, you can specify a memory range as:

```
Cmd_Input..Cmd_Input+4*2
```

If Cmd_Input is at address 400h, then the above expression is equivalent to specifying the address range 400h..408h.

You can also use a shorthand method to display a range of 128 locations. To do this, simply type two periods (..) after the lower range specifier, and leave the upper range blank. For example:

```
Cmd_Input..
```

Is equivalent to:

```
Cmd_Input..Cmd_Input+7fh
```

Which is equivalent to:

```
400h..47fh
```

Break Events

Break events stop execution of the user program and begin emulation monitor execution. Break events occur when:

1. the emulation processor tries to write to a memory address mapped as erom or trom
2. a pulse is received by the emulator external Trigger line
3. generated by the emulation analyzer or external state analyzer

Software breakpoints are provided so that you can configure a break event to occur on a certain instruction execution. You can display, add, remove, set, and clear 32 or fewer breakpoints. You can set, clear, or remove all the currently defined break events as a group, or individually, using the break control commands described in the following paragraphs.

Note

Breakpoints will be displayed automatically in the Emulation window when you define them, even if the Emulation window is not visible on the screen. By default, the Emulation window is displayed in the center of the screen. You can use the window utilities to view the Emulation window and see the current breakpoints.

Configure Break Events

You can enable or disable the following emulator break event conditions.

These are part of the general emulator configuration (**C**onfig **G**eneral):

- write-to-rom
- software breakpoints

These break events are part of the cross trigger configuration (**C**onfig **T**rigger):

- external trigger
- emulation trace
- external trace

Refer to your *Emulator PC Interface User's Guide* for details on your emulator's break event capabilities. Refer to the *CMB User's Guide* for information on cross triggering and coordinated measurements.

To cause the emulator to break when it finds a breakpoint, enter your choices in the cross trigger configuration form provided on screen. If you do not load an emulation configuration file, the PC Interface uses the default values.

Control Software Breakpoints

You can define no more than 32 software breakpoints.

You can perform multiple functions on software breakpoints, including:

- Display
- Add
- Remove
- Set
- Clear

Note

If an error occurs anywhere while the breakpoint command is executing, the result is written to the Error_Log window.

Display Software Breakpoints

The Emulation window is displayed in the middle of the screen. You can use ^ **a** to select each window until the Emulation window is active.

To display all currently defined breakpoints, enter:

```
Breakpoint Display
```

All breakpoints are automatically displayed in the Emulation window. You can verify that all the software breakpoints are displayed by looking at the Emulation window.

Memory locations that contain the same instructions used by software breakpoints will not be displayed. Only the breakpoints you define using the **B**reakpoint **A**dd command will be displayed.

If no breakpoints are set, the Emulation window will indicate "No software breakpoints are currently defined."

If you display breakpoints, and they take up more than the entire Emulation window, increase the window buffer size or window size using the **W**indow **U**tility **P**arameter command until you can see them all.

Add Software Breakpoints

When you add a software breakpoint, the command you execute automatically modifies the content of that memory location to the appropriate software breakpoint code for your emulator. For example, for the Z80 emulator, the specified memory location would be modified to 40h, which is the LD B,B statement.

If you add a breakpoint, but the address you specify already contains the equivalent code for the breakpoint, the message "Breakpoint code already exists: < address> " appears in the Error_Log window.

To add a software breakpoint at address 3000h, enter:

```
Breakpoint Add 3000h
```

The breakpoint at address 3000h will automatically be displayed and shown as “Set” in the Emulation window. Other breakpoints you might add will be displayed at the end of the list.

The content of the Emulation window should resemble:

```
          Breakpoint
   Status          Address
   ------          -------
   Set               3000H
```

You can enter a single address for a single breakpoint, or multiple addresses for multiple breakpoints, each separated by a semicolon (;). Multiple breakpoints will all automatically be added to the list.

Software breakpoints are always displayed in order of their address. For example, if the next software breakpoint you define occurs at 1000h, it will be placed above the currently defined breakpoint (3000h) in the list. Then if you define another breakpoint at 2000h, it will be placed in the middle of the list, and the Emulation window will resemble:

```
        Breakpoint
   Status          Address
   ------          -------
   Set               1000H
   Set               2000H
   Set               3000H
```

Remove Software Breakpoints

You use this feature to remove one or more existing software breakpoints.

Note

Even if the general emulator configuration previously disabled software breakpoints, you can still remove them. Be careful, because you cannot recover any removed breakpoints unless you redefine them.

If you try to remove a nonexistent breakpoint, the message "ALERT: Specified breakpoint not in list: < breakpoint> " will appear above the status line. The same message is displayed in the Error_Log window.

To remove all specified breakpoints, enter:

```
Breakpoint Reset All
```

All existing breakpoints will be removed. The Emulation window then shows that no software breakpoints are defined.

To remove a single breakpoint, enter:

```
Breakpoint Reset Single
```

Now, enter the address of the individual breakpoint you wish to remove.

Set Software Breakpoints

You use this feature to set (activate) one or more previously defined breakpoints. You may want to set all breakpoints, or an individual breakpoint, if you want your program to stop executing when it encounters any or all of them.

Note

When an emulator encounters an active software breakpoint (one that has been set), it stops foreground execution and begins executing in the monitor.

To set any currently defined breakpoints that were cleared, enter:

```
Breakpoint Set All
```

The defined breakpoints will automatically be set. Any breakpoints that were cleared will be shown as "Set" in the Emulation window. You can verify that all software breakpoints are set by looking at the Emulation window.

To set a single breakpoint, enter:

```
Breakpoint Set Single
```

Now enter the address of the breakpoint to be set.

You can enter either a single address, or multiple addresses separated by a semicolon (;).

Note

If you try to set a breakpoint that is not already defined, the message "ALERT: Specified breakpoint not in list: < breakpoint> " is displayed above the status line. This message also is displayed in the Error_Log window.

Clear Software Breakpoints

You use this feature to clear (deactivate) one or more previously defined software breakpoints. You may want to clear selected breakpoints if you want your program to ignore a particular breakpoint when the program begins executing.

HP 64700-Series emulators ignore all software breakpoints that have been cleared.

You can verify that all software breakpoints are cleared by displaying breakpoints, then looking at the Emulation window. When a breakpoint is cleared by a program execution, its status is not automatically changed to "Clear" in the Emulation window. But, when you use the **B**reakpoint **C**lear command to clear them, the status automatically changes to "Clear" in the Emulation window.

To clear a single breakpoint, enter:

```
Breakpoint Clear Single
```

Now, enter the address of the breakpoint to be cleared. The status of the breakpoint is automatically changed to "Clear" in the Emulation window.

You can enter either a single address, or multiple addresses separated by a semicolon (;).

To clear all currently defined breakpoints, enter:

```
Breakpoint Clear All
```

The breakpoints will automatically be cleared. You can verify that all software breakpoints are cleared by looking at the Emulation

window. The status of the breakpoint is automatically changed to "Clear" in the Emulation window.

Note

If you try to clear an undefined breakpoint, the message "ALERT: Specified breakpoint not in list: < breakpoint> " is displayed above the status line. This message is also displayed in the Error_Log window.

Emulator Registers

Since a microprocessor's operation can be traced by its register contents, the registers can be a valuable resource for solving problems in microprocessor hardware and software. So, the PC Interface allows you to display and modify the emulation processor registers.

Display Registers

This feature allows you to display the current contents of the emulation processor registers.

To access the register display feature, enter:

```
Register Display
```

The Emulation window displays the results.

Displaying registers causes a break in foreground execution. After the display process is complete, foreground execution continues.

Note

Each type of emulator has a unique register set. The way those registers are displayed is also unique. Refer to your *Emulator PC Interface User's Guide* for details about your emulator's registers.

Modify Registers

This feature allows you to modify the content of any of the emulation processor registers.

To modify a register, enter:

```
Register Modify
```

Enter your choices in the form provided on screen. Specify the register name in the Modify Register field. Then specify a value in the field below the register name. When you have completed the form, the register will contain the new value. You can modify only one register at a time. Modifying registers temporarily halts foreground execution. After the modify is complete, foreground execution continues. The command results are not displayed in a window. You can verify that the register contents were changed by using the **R**egister **D**isplay command.

You can use the **Tab** and **Shift Tab** keys to select any of the registers. Then type a new value for the register.

Refer to your *Emulator PC Interface User's Guide* for details about how your specific emulator registers are modified.

Note

You can store the result of a mathematical equation in a register. For example, you can store 4 * (5 + 2) in any of the emulator registers. The register will then contain 1C hexadecimal (28 decimal). The register contents display is hexadecimal. See the Expressions syntax in appendix A for more information.

Coverage Analysis

HP 64700-Series emulators include hardware that allow you to analyze memory coverage resulting from code execution. This can help you determine whether specific procedures were executed, or if a data block is accessed. The PC Interface includes commands to control these coverage tools.

Note

Coverage measurements are made only on emulation memory.

The coverage hardware works by recording a hit on a memory location any time that location is accessed, whether for a read or a write. The hit is remembered until the coverage hardware is reset. Therefore, you should remember to reset the coverage hardware before making an entirely new measurement to avoid confusing results. (However, if your goal is to exercise a new procedure and note the resulting change in coverage, do not reset the coverage hardware.)

Reset the Coverage Hardware

You can reset the coverage hardware with the command:

```
Memory Report Reset
```

A coverage measurement made immediately thereafter will report 0% coverage.

Run the Processor

Before you use the coverage measurement commands (and after you have reset the coverage hardware), you need to execute the user program. Otherwise, no memory locations will be accessed, yielding meaningless coverage data.

Check an Address Range

To see which memory locations were accessed within an address range or ranges, use the command:

```
Memory Report Accessed <address range;
address range ...>
```

Specify one or more address ranges. If you specify multiple ranges, they must be separated with semicolons. The Emulation window will display a list of all memory ranges that were accessed within the supplied ranges.

You can also find out which locations were *not* accessed within a range. Use the command:

```
Memory Report Nonaccessd <address range;
  address range...>
```

You'll see a list of all ranges within the specified range or ranges of memory that were not accessed.

Make a Percentage Measurement

You may simply want a figure that represents the memory accessed divided by the amount of memory in the range. To make this measurement, use the command:

```
Memory Report Percent <address range;
  address range ...>
```

Using the Emulator While Connected to a Target System

If you are using the HP 64700 emulator while it is connected to a target system, you must do the following before the target system will respond to the emulator.

1. Reset the emulator by entering: **P**rocessor **R**eset **H**old
2. Access the memory map by entering: **C**onfig **M**ap **M**odify

Note

Refer to your *Emulator PC Interface User's Guide* for details on the next 3 steps, then return here.

3. Enter the memory map configuration and delete all currently defined memory mapper terms.
4. Map memory according to your system needs (you might have some memory mapped as emulation RAM and some mapped as target system RAM).
5. Map all other memory as target system RAM (tram).
6. Save the memory map.
7. Enter the general emulator configuration by typing: **C**onfiguration **G**eneral
8. Select the external (target system) clock by responding with **n** to the internal clock question.

9. Save the general emulator configuration.

Now your emulator is ready to work correctly with your target system.

Where to Find More Information

To find out more about HP 64700-Series emulators, refer to the *HP 64700-Series Emulators Terminal Interface Reference* and your *Emulator Terminal and PC Interface User's Guides.*

For details on CMB operation, refer to the *CMB User's Guide.*

The *HP 64700-Series Emulators Terminal Interface Reference* contains details about the CMB commands.

For information on PC Interface Analyzer operation, refer to the *Analyzer PC Interface User's Guide.*

See chapter 4 for details about the PC Interface windows.

7

Creating and Using Command Files

Topics Covered

- What Are Command Files?
- What You Can Do With Command Files
- How to Create Command Files
- How to Use Command Files

What Are Command Files?

Command files are a collection of commands that allow you to accomplish and duplicate activities without having to enter all the commands manually.

Suppose that every time you access the PC Interface you want to create some user-defined windows. You can save time and keystrokes by creating a command file. Then you use that command file each time you want those windows created when you enter the PC Interface. You only need to create the command file once.

What You Can Do with Command Files

With command files you can:

- Redefine monitor colors.
- Redefine system window parameters.
- Create user-defined windows.
- Modify and display emulation processor registers.
- Modify and display emulation memory.
- Execute the emulation processor.
- Define breakpoints.
- Load and/or store a configuration file.
- Make trace measurements.
- Emulator configuration
- Memory map
- CMB and BNC trigger specifications

Any PC Interface commands that you execute using single letters and/or numbers can be included in a command file.

Commands Not to Include

There are commands that you cannot include in command files. These include:

- Key Macro Definitions
- Analyzer Specification

These must be stored in a configuration file.

The PC Interface reads command files differently than configuration files. Therefore, the method you use to load, store, and execute command files differs from the method you use to load, store, and use configuration files. (See chapter 5 for details about loading and storing the PC Interface configuration).

Command File Specifics

In PC Interface command files:

- A left angle bracket (<) resets the command file to the top of the file.
- An exclamation point (!) indicates a wait period of 1 second.
- The @ symbol located in column 1 specifies data.
- The # symbol preceding text signifies a comment.
- Pressing **ESC** terminates a command file.

Nesting Command Files

You can nest a maximum of 8 levels of command files. Nesting command files means that one command file calls another.

How to Create Command Files

You can create command files by:

1. Typing commands into a file using an editor.
2. Using the PC Interface **S**ystem **L**og command to record commands that execute during operation.

Using an Editor

You can use any editor on your host computer to create a command file. Create the command file as you would any text file.

At the top of the next page is a listing of a command file that creates one new window. The lines with the # symbol in the first column are comment lines.

Create this command file using an editor. When you finish, name the file “cmdfile.”

```
#open a new window
wo
#window name
@WIN1
#top row number
@0
#bottom row number
@5
#set autoclear on
@Y
#buffer size
@20
#left edge
@0
#right edge
@15
#display window when done
@Y
```

If you wanted to create 3 new windows, you would repeat this text 2 more times within the same file. Then you need to make each window name unique, and change each window’s parameters to suit you.

Using the System Log Feature

Instead of using an editor to create a command file, you can start the **S**ystem **L**og option. This allows you to record all commands that you execute, and/or the resulting output of those commands, in a file.

Logging both commands and output can be helpful when executing a set of commands that you are not sure will produce the results you are seeking. By logging commands that you type, you can record everything you try. By logging the resulting output, you can see if the expected results occurred. When you finish logging commands, you can use the “log” file as a command file.

Note

You do not have to modify the log file to use it as a command file, because all commands and output are stored in a format that the PC Interface can read when you load the command file. Still, you may want to edit the log file to remove any unwanted commands or results, or to add commands or comments.

To log both commands and the results from those commands into a file, enter:

```
System Log Both Enable <log_file>
```

You can specify any valid directory information if you want the file stored in another directory. If you try to enable logging of input and output to a log file that is already enabled, the message "The logging of input and output has already been enabled" will be displayed at the bottom of the screen.

Note 

If a file exists in the current directory with the same name that you want to specify for the new file, this command will overwrite the existing file. Make sure that an existing file does not have the same name you want to use for the new file.

To log just the commands that you execute, enter:

```
System Log Input Enable <log_file>
```

Only the input (commands that you type) will be recorded in the log file. The results of the commands that you type will not be recorded in the log file.

Note 

If you try to enable logging of input to a log file that is already enabled, the message "The logging of input has already been enabled" will be displayed at the bottom of the screen.

To log only the results of the commands that you execute, enter:

```
System Log Output Enable <log_file>
```

Only the output (results of the commands that you type) will be recorded in the log file. The commands themselves will not be recorded in the log file.

Note

If you try to enable logging of output to a log file that is already enabled, the message "The logging of output has already been enabled" will be displayed at the bottom of the screen.

To disable logging of input and output, enter:

```
System Log Both Disable
```

The recording of commands and results of commands to the log file is automatically disabled.

You can disable either the logging of input or output to a log file that already has both features enabled. After you disable either of those features, you can enable them again later. If you disable logging of both input and output, all logging to the log file is disabled (see figure 7-1).

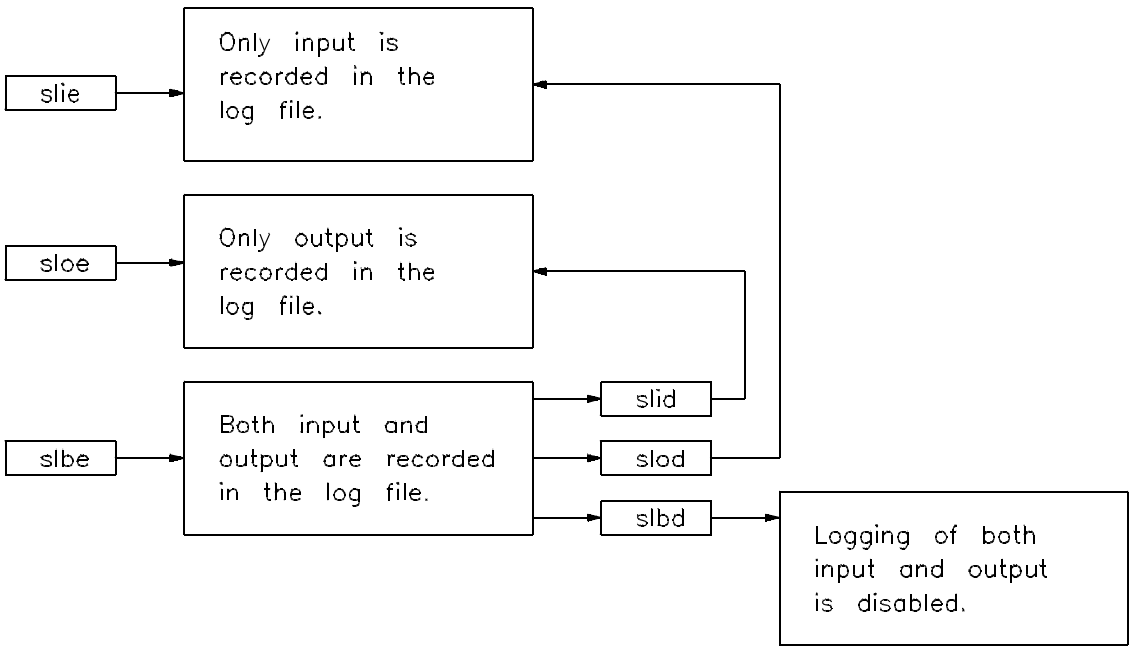


Figure 7-1. Using the Log Commands

How to Use Command Files

Suppose you want to use the command file you created in this chapter. You can use the command file in two ways:

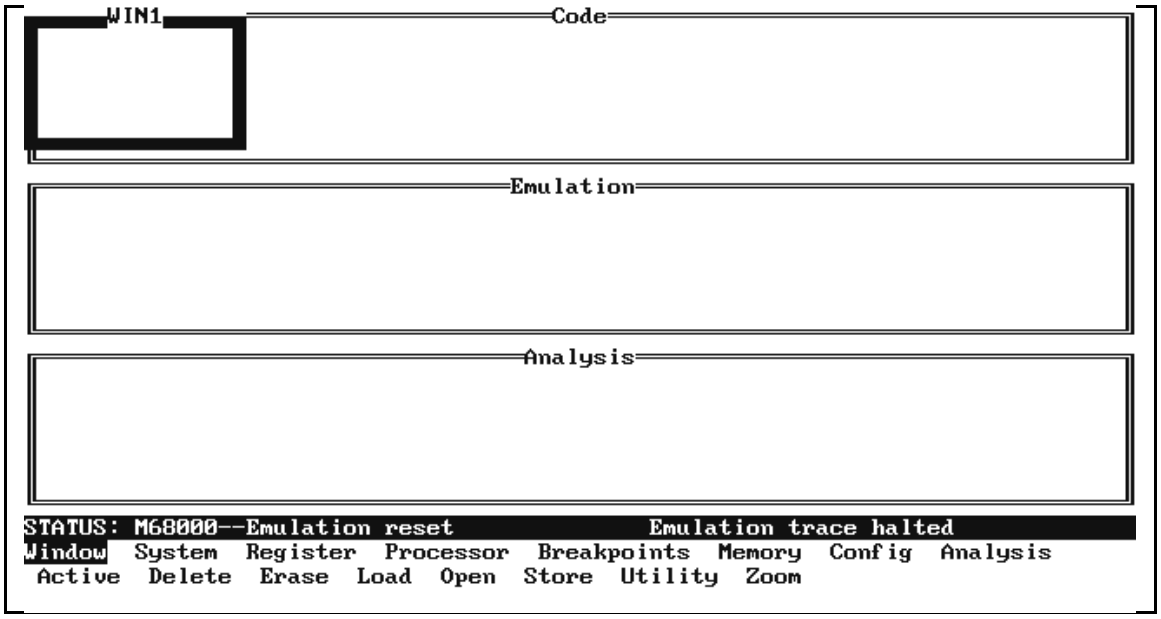
1. You can specify the command file name when starting up the PC Interface. For example, enter:

```
pc<product>  /c  cmdfile <emulname>
```

2. After the PC Interface is started, you can use **S**ystem **C**ommand_file to load the command file from the MS-DOS system level. For example, enter:

```
System  Command  cmdfile
```

In either case, your command file will produce the same results, shown below.



Notes

8

Function Key Macros

Introduction

Function Key Macros let you assign keystroke sequences to a function key or function key combination. Thus, you can reduce typing and simplify repeated measurements by assigning your most frequent command sequences to a function key macro.

How to Define Function Key Macros

There are forty possible function keys that may be assigned to macros.

- Function keys F1-F10
- Function keys F1-F10 in combination with **Shift**
- Function keys F1-F10 in combination with **Alt**
- Function keys F1-F10 in combination with **Ctrl**

Creating a Macro

Suppose you want to create a macro to load memory with a certain absolute file with a single keystroke. The file is CMD_RDR.L and is in HP64000 file format. It's in the directory \myproject\sources.

Enter the macro definition menu with the command:

```
System Config Key_Macro
```

The first field indicates the function key you want to define. Suppose you want this sequence assigned to **Ctrl-F5**. Use the **Tab**

and **Shift-Tab** keys until that sequence appears in the field. Press **Enter** to accept your choice.

You can also select the key sequence by typing in the sequence of characters that represent the sequence. For example:

If you want:	Type:
F5	< F5>
Shift-F5	< ShftF5>
Alt-F5	< AltF5>
Ctrl-F5	< CtrlF5>

The second field allows you to select the key that terminates the macro definition in the menu. It is predefined as **Esc**. For this example, leave it as is.

For some macros, you might need to change this definition. For example, you might need the **Esc** key within a macro to exit a configuration menu without changing it. To change the termination key, use the **Tab** and **Shift-Tab** keys to scroll through the available choices. Press **Enter** when you're finished.

The last field allows you to enter the keystroke sequence that will be assigned to the given function key. For this example, we want to load memory with an absolute file. Enter the key sequence:

```
MLHP64000EnterEnterEnterEnter
\myproject\sources\cmd_rdr.lEnterEsc
```

(When we say **Enter** in this section, we mean press the **Enter** key on your keyboard.)

The final **Esc** character terminates the macro definition and won't appear in the definition string. The string will look like the following (to see it, you have to reenter key_macro definition):

```
MLHP64000<^M><^M><^M><^M>
\myproject\sources\cmd_rdr.l<^M><^M>
```

Now, when you press **Ctrl-F5**, memory will be loaded with the absolute file **cmd_rdr.l**.

Nesting and Chaining Macros

You can nest macros to perform complex measurement sequences. You might have the following sequence assigned to **F3**:

```
key_seq1<ShftF5>key_seq2
```

Shift-F5 may then have another sequence. Nesting of macros is supported. For example, **Shift-F5** might be assigned the following key sequence:

```
key_seq3<F4>key_seq4
```

You are limited to 16 levels of nesting. Direct or indirect recursion of macros is not permitted, except as a chain. For example, suppose you have the following assigned to F3:

```
key_seq1<F3>key_seq2
```

This isn't a valid macro. But you can do the following:

```
key_seq1key_seq2<F3>
```

Keystroke Representations

When you selected the keystroke for activation of the macro, you could either use **Tab** and **Shift-Tab** to select the desired combination, or directly type in a character string to represent it. This works for all key macro definitions. In fact, you can create configuration files that define key macros by using a text editor , as long as you follow the rules for representing keystrokes. The rules are:

- All keystroke sequences that are not part of the standard ASCII character set must be enclosed in angle brackets. (Characters in the range 128..255 decimal are not part of the standard ASCII character set.) For example, function key 3 is represented as < F3> . Also, characters 0-31 decimal in the ASCII set must be enclosed in angle brackets (such as **Ctrl-I** (**Tab**) (^ I)).

- Keystroke sequences including the control key (**Ctrl**) are represented using the up caret symbol (^) when the following character is part of the ASCII character set, and by the string **Ctrl** when the following character is not part of the ASCII character set. For example, **Ctrl-M** (**Enter**) is shown as < ^ M> ; **Ctrl-F5** is shown as < CtrlF5> .

Normally, when you're using **C**onfig **K**ey_Macro to define a macro, it's easier to simply use a particular key rather than enter a string representing that key. But, when you're using a text editor to create a configuration file, pressing that key may have other consequences. For example, if you press **Enter** in your editor, a new line is started. So, you type < ^ M> in your editor file, which the PC Interface uses to represent the **Enter** key.

Editing Macros

The PC Interface provides intelligent editing of key_macros within the **C**onfig **K**ey_macro form. The following keys can help you when entering or changing the key_macro definition:

Delete deletes the next actual keystroke character. For example, if you have the cursor positioned on < ^ M> , pressing **Delete** will remove all four characters associated with that sequence (which is the Enter key).

Ctrl-left arrow and **Ctrl-right arrow** move the cursor left and right, respectively, one keystroke at a time. Suppose you have the following sequence:

```
<ShftF5>ml<^M><^M>
```

If you start with the cursor positioned at the beginning of < ShftF5> , and press **Ctrl-right arrow** four times, the cursor will be positioned at the beginning of the second < ^ M> . If you then press **Ctrl-left arrow** twice, the cursor will be positioned on the l.

Organizing Your Macros

Since you can define a large number of macros, you may want to arrange them to aid your memory. For example, you might assign all unshifted function keys to System functions, all Alt function keys to Analysis functions, and so on. Or, if you are using the emulator in a production testing environment, you might arrange the macros so that the key number corresponds to steps in a sequence.

For long keystroke sequences, it may be better to have the macro call a command file. This is particularly true when you are configuring the emulator for a measurement; for example, you want to change the configuration, map memory, and load an absolute file. Simplify by building command and configuration files that will be loaded with one macro.

Saving/Restoring Macros

Macro definitions are saved when you save the emulator configuration in a file, and restored when you load a configuration file. See chapter 5 for more information on configuration files. Remember that you can define key macros externally, using a text editor. See the earlier section on "Keystroke Representations."

Predefined Macros

When you start the PC Interface without a configuration file, or with the default configuration file, there are three predefined function keys.

- F1 is defined as

```
Processor Step Pc 1
```

- F2 is defined as **Ctrl ** (the system terminal abort sequence)

- F10 is defined as

```
System Exit Locked
```

You can redefine these three keys to any sequence you want. Be sure to save the new definitions by creating a new configuration file. See chapter 5 for more information on configuration files.

How to Use Function Key Macros

You use function key macros by simply pressing the key with the desired command sequence. For example, if you have **S**ystem **S**ymbols **G**lobal **D**isplay assigned to **Alt-F3**, press and hold the **Alt** key, then press **F3**. Global symbols for the current absolute file (if loaded) are displayed.

Press the **Esc** key to terminate the repetition of nested or chained macros.

Examples

Here are some example macros to help you understand how key macros work, and give you ideas for building your own macros. Each macro shows both the keystroke sequence and corresponding string representation of the macro. You enter the given keystroke sequence; the string will appear in the macro definition field. All the examples assume that **Esc** (< ESC>) terminates the macro definition; this key is not shown as part of the keystroke sequence.

Example 1

Build a macro to end modification of a PC Interface form and save its contents.

Macro activated by: < F5>

Keystroke sequence: **EndEnter**

Corresponding String: < End> < ^ M>

Example 2

Build a macro to create a user-defined window and load a source file into it. This example builds on the previous one by using **F5** as part of the macro sequence.

Macro activated by: < F6>

Keystroke sequence: **wosource1EnterEnterEnterEnter1000Enter 40F5wlEntercmd_rdr.SEnter**

Corresponding String: wosource1< ^ M> < ^ M> < ^ M> < ^ M> 1000< ^ M> 40< F5> wl< ^ M> cmd_rdr.S< ^ M>

Example 3

Although repetitive memory display is a feature in the PC Interface, you can use macro definitions to build a similar capability, which can be translated to other measurements. This example builds a macro to repetitively display 128 bytes of memory at a particular location. Two different macros are defined to support this feature.

Macro activated by: < F7>

Keystroke sequence: **Msg_Dest..Enter**

Corresponding string: Msg_Dest..< ^ M>

Macro activated by: < F8>

Keystroke sequence: **mdbF7F8**

Corresponding string: mdb< F7> < F8>

Now, when you press **F8**, the memory at Msg_Dest thru Msg_Dest+ 7fh will be repetitively displayed in byte format.

Example 4

This example builds a "trace continuous" feature. It requires three macros. (You can put all the keystrokes into one macro if you want.) It uses the previously defined **EndEnter** macro assigned to **F5**. These examples assume that the analyzer has external analysis capability. If your emulator doesn't, simply omit the i from the **Shift-F1** and **Shift-F2** macros.

Macro activated by: < ShftF1>

Keystroke sequence: **abi**

Corresponding string: abi

Macro activated by: < ShftF2>

Keystroke sequence: **adiF5**

Corresponding string: adi< F5>

Macro activated by: < ShftF3>

Keystroke sequence: **Shift-F1Shift-F2Shift-F3**

Corresponding string: < ShftF1> < ShftF2> < ShftF3>

Now, when you press **Shift-F3**, the analyzer repeats trace measurement and display. If there are no states to display, the PC Interface will beep. The analyzer will display the first 16 states by default. For your measurements, you may want to define other macros to set up the trace specification, and redefine the macro assigned to **Shift-F2** to select the states for display.

Note

To improve the effect of this macro, use the Window Utilities Parameters command to set the window parameter **autoclear** to **y** and **scroll** to **n**. Then, the entire analysis window will be updated at once.

A

PC Interface Syntax Summary

Introduction

The PC Interface command syntax summary covers:

- Windows
- System
- Registers
- Processor
- Breakpoints
- Memory
- Configuration
- Analysis
- Expressions
- Analyzer Pattern Expressions

Conventions Used

The conventions used in this chapter are:

- Angle brackets (< >) enclose variables that you type in or select with the **Tab** or **Shift Tab** keys.
- A vertical bar (|) separates “or” options (y| n, for example).
- Parentheses (( )) enclose results.
- Letters in the first column refer to main level PC Interface commands.

- Letters in the second column refer to options to the main level PC Interface commands.
- Letters in the third column refer to options to the second level PC Interface commands, and so on.

Window Syntax Summary

System windows and user-defined windows can be accessed and controlled with the following options.

```
W(indow) A(ctivate) <window>
         D(elete)   <window>     <y|n>*
         E(rase)    <window>     <y|n>
         L(oad)     <window>     <file>
         O(pen)     <new_window>      <top> <bottom> <autoclear>
                                      <buffersize> <left> <right> <display>
         S(tore)    <window> <from> <thru> <destination>
         U(tility)  c(olor)        <monitor_type> <foreground> <background>
                    H(ide)         <window>
                    P(arameter)    <window> <top> <bottom> <autoclear>
                                   <buffersize> <left> <right> <scroll>
                    S(earch)       <window> <from> <thru> <find_string>
                    V(iew)         <window>
         Z(oom)     <window>

* only valid if user-defined windows exist
```

System Syntax Summary

System features allow you to access the host computer system.

```
S(ystem) C(ommand_file) <command_file_name>
         W(ait)    k         <press_any_key>
                   T(ime)         <delay_in_seconds>
                   M(easurement)   I(nternal) ******
                                   E(xternal)
                                   B(oth)
         M(S-DOS)  F(ork)*
                   C(ommand) <MS-DOS_command>
         L(og)     I(nput)   E(nable)   <log_file_name> **
                             D(isable)
                   O(utput)  E(nable)   <log_file_name> **
                             D(isable)
                   B(oth)    E(nable)   <log_file_name> **
                             D(isable)
         T(erminal)***
         S(ymbols) G(lobal)  D(isplay)   <symbol_database_file>****
                             L(oad)      <symbol_database_file>
                             T(ransfer)*****
                             R(emove)*****
                   L(ocal)        <local_symbol_module>
                             D(isplay)      <local_symbol_module>*****
                             T(ransfer)     G(roup)  <module, module, ...>*****
                                            A(ll)*****
                             R(emove)       G(roup)  <module, module, ...>*****
                                            A(ll)*****
                             L(oaded)*****
         E(xit)    L(ock)
                   U(nlock)
                   N(o_save)

* accesses host system level...type exit to return to PC Interface
** only valid if a log file is not currently enabled
*** accesses Terminal Interface...type Ctrl backslash to return to PC Interface
**** only valid if a symbol file has been loaded
***** only valid if emulator has symbol handling capability
****** S(ystem) M(easurement) only valid if emulator equipped with external analyzer
```

Registers Syntax Summary

Registers vary from processor to processor. Refer to your *Emulator PC Interface User's Guide* for details about how the PC Interface controls your emulation processor's registers.

Note

This example describes the 68000 emulator registers.

```
R(egister)        D(isplay)     A(ll)
                                S(ingle)       <register_name>
                  M(odify)      <register_name>   <value>
```

Processor Syntax Summary

The following options are available for controlling the emulation processor.

```
P(rocessor)   G(o)      P(c)
                        A(ddress) <address>
                        R(eset) *
              B(reak)
              R(eset)   M(onitor)
                        H(old)
              I(/o)     D(isplay) <I/O address;...>***
                        M(odify)  <I/O address=value;...>***
              C(mb)     G(o)      P(c) **
                                  A(ddress)  <address> **
                        E(xecute)
              S(tep)         P(c)      <#instructions>
                             A(ddress) <#instructions> <address>
                             E(vents)  <displayreg><displaymnem><cmd_file>
                                       <displayonstepcntcomplete>
```

```
* only valid if a target system is connected and operating properly
** only valid after CMB trigger goes true
*** only valid for processors with separate memory and I/O
```

Breakpoints Syntax Summary

You manage software breakpoints with the following options.

Note

You must view the Breakpoint window to see the results of breakpoint processes.

```
B(reakpoint)  D(isplay)
              A(ddress) <address;address...>
              R(eset)*  A(ll)
                        S(ingle)  <address>
              S(et)*    A(ll)
                        S(ingle)  <address>
              C(lear)*  A(ll)
                        S(ingle)  <address>

* only valid after adding one or more breakpoints
```

Memory Syntax Summary

You can control emulation memory and target system memory with the following options.

```
M(emory) D(isplay) B(yte)          <address;address...>
                   W(ord)          <address;address...>
                   L(ongs)         <address;address...>
                   M(nemonic)      <address;address...>
                   R(epetitively)  (press ESC to abort)
         M(odify)  B(yte)          <address=valuelist;address=valuelist...>
                   W(ord)          <address=valuelist;address=valuelist...>
                   L(ongs)         <address=valuelist;address=valuelist...>
         L(oad)         <file_format>      <memory_type><file/processor options>
                                           <absolute_file_name>
         S(tore)        <file_format>      <memory_range><absolute_file_name>
         C(opy)         <source_range>     <destination>
         F(ind)         <memory_range>     <data_pattern>
         R(eport)     A(ccessed)     <address_range;address_range...>
                      N(onaccessed)  <address_range;address_range...>
                      P(ercent)      <address_range;address_range...>
                      R(eset)
```

Configuration Syntax Summary

You can configure all the emulator features using the following options.

```
C(onfig) L(oad)         <config_file_name>
         S(tore)        <config_file_name>
         G(eneral) *
         M(ap)          M(odify) **
                        R(eset)
         T(rigger) ***
         K(ey_macro)     <fkey_combination><termination_char><key_sequence>

* enters general emulator configuration; press End Enter to save, or ESC to cancel
** enters emulator memory map; press End Enter to save, or ESC to cancel
*** enters cross-trigger configuration; press End Enter to save, or ESC to cancel
```

Analysis Syntax Summary

The following command sequences set up, start and stop analyzer measurements.

```
A(nalysis)     B(egin)     I(nternal)
                           E(xternal)
                           B(oth)
               H(alt)      I(nternal)
                           E(xternal)
                           B(oth)
               C(mb)       B(egin)      I(nternal)
                                        E(xternal)
                                        B(oth)
                           E(xecute)    I(nternal)
                                        E(xternal)
                                        B(oth)
               S(ystem)    <external_pod_configuration>
               F(ormat)    I(nternal)*
                           E(xternal)**
               T(race)     M(odify)     I(nternal)***
                                        E(xternal)****
                           R(eset)      I(nternal)
                                        E(xternal)
                                        B(oth)
               D(isplay)   I(nternal)   <start_state> <end_state>
                           E(xternal)   <start_state> <end_state>
```

* Enters Internal State Format Specification, press END ENTER to save or ESC to cancel.

** Enters External State/Timing Format Specification.

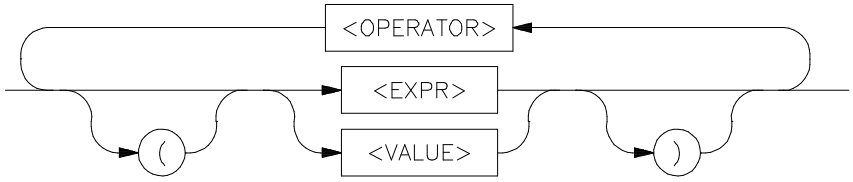
*** Enters Internal State Trace Specification, press END ENTER to save or ESC to cancel.

**** Enters External State/Timing Trace Specification.

Note: Internal, External, and Both options are only displayed if the emulator is equipped with the external analyzer option.

Expressions

Syntax



Description

Numeric expressions are the root of all HP 64700 PC Interface expression types, including analyzer expressions and address specifications.

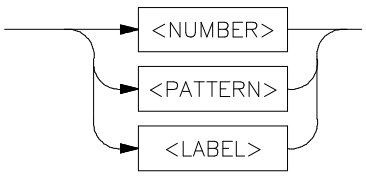
The expression capability in the PC Interface is very powerful. You may specify numbers in one of four different bases and use many different arithmetic and logical operators to form more complex expressions.

PC Interface expressions consist of other **expressions** and **values**, which may be modified by various **operators**. You may change the precedence of operators by enclosing expressions within parentheses.

Values

Values consist of **numbers** (in one of four bases), **patterns** (hexadecimal, octal, or binary numbers that also include don't care values), and **labels** (symbols which reference other numbers, from a symbol database).

<VALUE>



Numbers are in hexadecimal, decimal, octal, or binary. You specify the base as follows:

Y y	Binary (example: 10010y)
Q q O o	Octal (example: 377o or 377q)
T t	Decimal (example: 197T)
H h	Hexadecimal (example: 0A7fH) (Note that hexadecimal numbers starting with any one of the letter digits A-F must be prefixed with a zero; otherwise the system will return an error message)

If you do not specify a base, numbers default to hexadecimal or decimal, depending on the context.

All numbers used in address specification, analyzer expressions, and any other specification relating to a microprocessor address, data or status value defaults to hexadecimal.

Numbers used to specify repeat count values, such as in the analyzer trace specification or step count, default to decimal.

Floating point numbers are supported by some emulators. The only legal operations on these values are addition, subtraction, multiplication, and division. You represent a floating point value as a decimal in the form:

```
[+|-]X.X[E[+|-]X]
```

Where X's represent decimal values and E indicates a following exponent.

Patterns are hexadecimal, octal, or binary numbers which include don't care digits, specified by the letters **X** or **x**. The character **?** represents a pattern of all don't care digits. For example:

1011xx11y

0A7Xh (equivalent to 000010100111xxxxy)

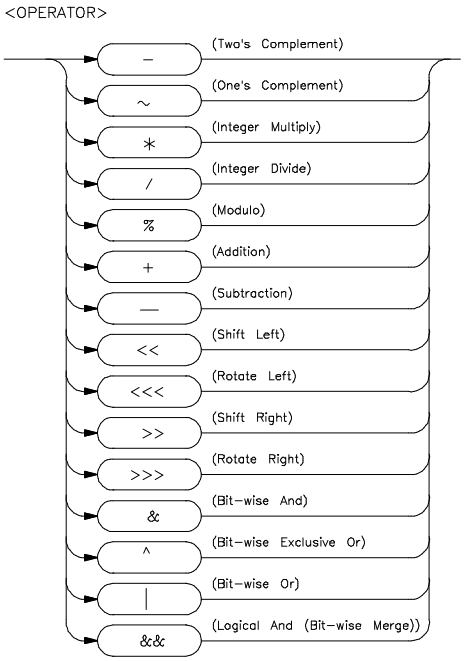
2x5Q (equivalent to 010xxx101y)

You will generally use patterns only in analyzer expressions.

Labels refer to names equated to numbers via the symbol database loaded for the current program or loaded using the **S**ystem **S**ymbols **G**lobal **L**oad command.

Operators

The expression capability includes a powerful set of operators, freeing you from the need to calculate expressions before entering them into other expressions. All operations are carried out on 32-bit two's complement signed integers (values which are not 32-bit will be padded out with zeros when expression evaluation occurs). For emulators that directly support floating point data types, addition, subtraction, multiplication and division are done using 64-bit floating point format.



The operators are listed in the following diagram and described in order of evaluation precedence. As mentioned above, you may use parentheses in the expression to change the order of evaluation.

Note

If your emulator supports symbols, and you are using a symbol in an expression, only the + and - operators are valid before and after the symbol. For example: 100h+ main-5

- ~	Unary two's complement, unary one's complement. Two's complement is not allowed on patterns containing don't care bits. This is the truth table for one's complement: 0 = > 1 1 = > 0 X = > X Examples: ~ 1x0y = 0x1Y -1101Y = 0011Y
* / %	Integer multiply, integer divide, integer modulo. These operations are not allowed on patterns containing don't care bits. Examples: 30afH*21 = 06468fH 23T%4T= 3 0fa6/2 = 07d3h

+ -	Addition, subtraction. Not allowed on patterns containing don't care bits.

Examples:

03dh+ 03fh = 07ch

1110Y-101Y = 1001Y

< < < < < < > > > > > >	Shift left, rotate left, shift right, rotate right (you must specify the number of locations to shift or rotate after the operator).

Examples:

1x0Y< < 1 = 1x00Y

1x0Y> > 1 = 01xY

1x01Y> > > 1 = 100000000000000000000000000001x0Y

0xxf0abcdH> > > 4 = 0dxxf0abcH

&	This symbol (&) represents a bit-wise AND operation. The truth table resembles:

&	**0**	**1**	**X**
0	0	0	0
1	0	1	X
X	0	X	X

For example:

10xxy&11x1Y = 10xxY

^ This symbol (^) represents a bit-wise exclusive OR operation. The truth table resembles:

^	**0**	**1**	**X**
0	0	1	0
1	1	0	X
X	0	X	X

For example:

10xxY^ 11x1Y = 01xxY

| This symbol (|) represents a bit-wise inclusive OR operation. The truth table resembles:

	0	**1**	**X**
0	0	1	0
1	1	1	1
X	0	1	X

For example:

10xxY| 11x1Y = 11x1Y

&& This symbol (&&) represents a bit-wise merge operation. The truth table resembles:

&&	**0**	**1**	**X**
0	0	*	0
1	*	1	1
X	0	1	X

An overlap, indicated by a ***** in the merge truth table, may occur if two patterns specify different values for a pattern bit. If an overlap occurs, the first pattern's value for that bit overrides the second pattern's value.

For example:

10xxY&&11x1Y = 10x1Y

Using Expressions in Addressing and Analyzer Expressions

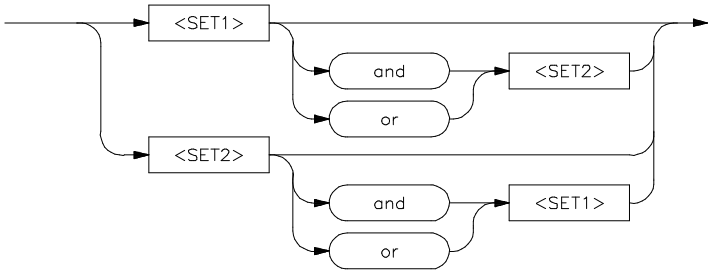
You can use the expression evaluation capability to form more powerful expressions for use in specifying addressing and analyzer expressions. For example, suppose you want to trigger the analyzer on the access to trap vector 13. Instead of calculating the address, since you know the base address is 080 hex and each vector is 4 address bytes, you can specify this as:

```
tg addr= (080h+ (13T*4))
```

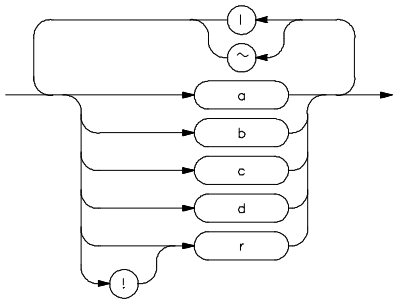
Analyzer Pattern Expressions

Syntax

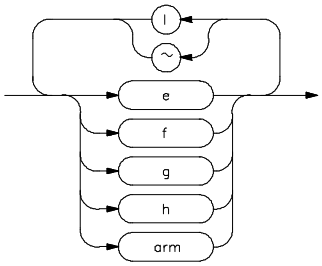
<COMPLEX_EXPR>



<SET1>
(restricted to one operator type in the set)



<SET2>
(restricted to one operator type in the set)



Description

In the PC Interface analyzer trace specification, you use pattern labels, which have been assigned to various simple expressions, to form complex expressions.

Pattern Labels and Ranges

You assign pattern labels to simple expressions using the analyzer trace specification form. For example:

```
Pattern a: addr=2000
Pattern b: data!=00
Pattern c: stat=dma
Pattern d: addr=2000 and data=23
Pattern e: addr!=2105 and data!=0fc
```

You can also assign the range value:

```
Range: data=42..44
```

Sets

The pattern labels, along with the range and arm specifications, are divided into two sets.

Set 1:

a,b,c,d,r,!r

Set 2:

e,f,g,h,arm

Intraset Operations

You use intraset operators to form relational expressions between members of the same set. The operators are:

~ (intraset logical NOR)

| (intraset logical OR)

The operators must remain the same throughout a given intraset expression. So, you could form the following types of intraset expressions:

```
a~b~r
```

(Pattern a NOR pattern b NOR range.)

```
b | !r
```

(Pattern b OR (NOT range).)

```
e | arm
```

(Pattern e OR arm.)

```
f ~ h
```

(Pattern f NOR pattern h.)

You **cannot** use the intraset operators to form expressions between set 1 and set 2. Also, remember that the intraset operator must remain the same throughout the set. Therefore, the following examples are **invalid**:

```
b~c|d
```

(This is incorrect because the operator must remain the same throughout the set.)

```
b~e
```

(You cannot use intraset operators for interset operations.)

Interset Operations

You use interset operators to form relational expressions between members of set 1 and set 2. The operators are:

and (interset logical AND)

or (interset logical OR)

You can then form the following types of expressions:

(set 1 expression) and (set 2 expression)

(set 1 expression) or (set 2 expression)

The order of sets does not matter:

(set 2 expression) and (set 1 expression)

Combination

You can use both the intraset and interset operators to form very powerful expressions.

```
a~b and e|arm
c or f~g~h
```

However, you cannot repeat different sets to extend the expression. The following is **invalid:**

```
a~b and e and c and g
```

DeMorgan's Theorem and Complex Expressions

It seems that you only have a few operators to form logical expressions. However, using the combination of the simple and complex expression operators, along with a knowledge of DeMorgan's Theorem, you can form virtually any expression you might need in setting up an analyzer specification.

DeMorgan's theorem in brief says that

A NOR B = **(NOT A) AND (NOT B)**

and

A NAND B = (NOT A) OR (NOT B)

The NOR function is provided as an intraset operator. However, the NAND function is not provided directly. Suppose you wanted to set up an analyzer trace of the condition

(addr= 2000) NAND (data= 23)

This can be done easily using the simple and complex expression capabilities. First, you would define the simple expressions as the inverse of the values you wanted to NAND:

```
Pattern a: addr!=2000
Pattern b: data!=23
```

Then you would OR these together using the intraset operators:

```
a|b
```

This is effectively the same as:

(NOT addr= 2000) OR (NOT data= 23) =
(addr= 2000) NAND (data= 23)

If you need an intraset AND operator, you can use the same theory. Suppose you actually wanted:

(addr= 2000) AND (data= 23)

First, define the simple expressions as the inverse values:

```
Pattern a: addr!=2000
Pattern b: data!=23
```

Then you would NOR these together using the intraset operators:

```
a~b
```

This is effectively the same as:

(NOT addr= 2000) NOR (NOT data= 23) =
(addr= 2000) AND (data= 23)

For further information on using and entering analyzer expresssions, refer to the *Analyzer PC Interface User's Guide*.

Notes

Index

A

^ a, **3-26, 4-9, 4-11**
absolute file, **6-15**
activate a window, **4-8**
active window, **3-17**
add software breakpoints, **6-20**
addition operator, **A-14**
ALERT messages, **3-8**
analyzer
  expressions, **A-16**
AND (bit-wise) operator, **A-14**
and, interset logical AND operator, **A-19**
appending to files, **3-22**
arm condition
  complex expressions, **A-18**
ASCII file loaded into a window, **4-9**
attributes of system windows, **4-4**
autoclear characteristic, **4-13**

B

^ b, **4-19**
background color, **4-12**
bases (number), **A-11**
basic emulation features, **3-26**
baud rate, **2-5**
binary number base specifier, **A-11**
bit-wise operators
  AND, **A-14**
  exclusive OR, **A-15**
  inclusive OR, **A-15**
  merge, **A-16**
break events, **6-18**
break the microprocessor, **6-5**
Breakpoint window, **6-18**
breakpoints syntax summary, **A-6**
buffer size characteristic, **4-13**
buffer size consumes memory, **4-7**

C
/c option, **3-6**
characteristics of the windows, **4-11**
clear software breakpoints, **6-23**
clear window before writing data, **4-13**
CMB EXECUTE signal, **6-8**
CMB measurement, **6-5**
CMB Trigger, **6-3, 6-8**
Code window, **3-17**
color characteristic, **4-12**
columns, **3-7**
command file, **3-14**
command files, **7-1**
command/message line, **3-8**
commands, **3-10**
commands in a log file, **3-20**
communication port, **3-13, 4-20**
complex expressions, **A-20**
config.sys file, **2-6**
configuration file, **3-2, 5-9**
configuration syntax summary, **A-8**
configurations, **1-3**
configure break events, **6-19**
control character summary, **4-11**
control software breakpoints, **6-19**
controlling emulators, **6-1**
conventions, **3-2, A-1**
Coordinated Measurement Bus (CMB), **6-3**
copy memory, **6-16**
Coresident Programs
problems with, **2-8**
create a command file, **3-14**
create a user-defined window, **4-5**
creating a user-defined window, **3-22**
creating command files, **7-1**
current directory, **5-2**
cursor location is retained, **4-11**

D
data entered into log file, **3-20**
data entry lines, **3-7**
data pattern in memory, **6-17**
dataword width, **2-5**
decimal number base specifier, **A-11**

default colors, **4-6**
define multiple environments, **2-4**
define the shell environment variable, **2-6**
define your own windows, **4-14**
delete a user-defined window, **4-7**
delete user-defined windows, **4-16**
DeMorgan's theorem, **A-20**
display I/O, **6-6**
display memory, **6-13**
display registers, **6-24**
display software breakpoints, **6-20**
display symbols, **5-4**
divide (integer) operator, **A-13**
double-line border, **4-4**

E

empty window, **4-10**
emulation features, **3-26**
emulation memory, **6-13**
emulation processor, **6-4**
Emulation window, **6-6**
emulator device table file, **2-2, 3-4**
emulator type, **2-5**
emulator-specific information, **5-11**
emulators, **6-1**
eram, **6-10**
erase a window, **4-8**
erasing your user-defined window, **3-24**
erom, **6-10**
error occurs, **6-2**
Error_Log window, **6-2**
example configuration files, **5-13**
example emulator configuration, **5-12**
example window configuration, **5-10**
exclusive OR (bit-wise) operator, **A-15**
executing a single system level command, **5-1**
executing example command file "tutorial", **3-27**
executing multiple system level commands, **5-3**
exit the PC Interface, **3-27**
exit the System Terminal window, **4-18**
expressions
  analyzer, complex configuration, **A-20**
  operators, **A-12**

F features of the PC Interface, **1-1**
features of the windows, **4-1, 4-5**
field in a form, **3-22**
file format, **6-16**
find a data pattern in memory, **6-17**
flexible disk, **5-3**
foreground color, **4-12**
fork feature, **3-11**
form, **3-2**
functions of the system windows, **4-5**

G general emulator configuration, **6-2**
getting started, **3-1**
global symbols, **5-4 - 5-5**
grd, **6-10**

H H,h, hexadecimal number base specifier, **A-11**
halted state, **6-6**
hexadecimal number base specifier, **A-11**
hide a window, **4-8**
highlighted border, **3-16**
host computer system, **3-13**
how many windows you can create, **4-16**
how to access and use the windows, **4-11**
how to create command files, **7-3**
how to delete user-defined windows, **4-7**
how to exit the PC Interface, **3-27, 5-15**
how to start the PC Interface, **3-3**
how to use command files, **7-7**
how to use the System Terminal window, **4-17**

I I/O port addresses, **6-6 - 6-7**
inclusive OR (bit-wise) operator, **A-15**
information, **1-3, 6-28**
installation and setup, **2-1**
interset operators, **A-19**
intraset operators, **A-18**
inverse values (complex analyzer expressions), **A-20**

K keys to perform various functions, **3-8**

L labels, **3-10**
learn about the PC Interface screen, **3-7**

load a window, **4-9**
load memory, **6-15**
load symbols, **5-4**
loading the PC Interface configuration file, **5-13**
local symbols, **5-4, 5-6**
locate a string in a window, **4-10**
location of window on screen, **4-12**
log both commands and results, **7-5**
log feature, **5-4**
log file, **3-14**
log only commands, **7-5**
log only results of commands, **7-5**
logging commands, **3-14**
logging commands and output to a file, **5-4**
logical emulator name, **2-5**
logical operators
See operators

M /m option, **3-5**
main level, **3-10, 3-12**
main PC Interface options, **3-10**
memory buffer, **4-6**
memory mapper, **6-9**
memory syntax summary, **A-7**
Memory window, **6-13**
merge (bit-wise) operator, **A-16**
microprocessor execution, **6-2**
modify I/O, **6-7**
modify memory, **6-14**
modify registers, **6-24**
modify the general emulator configuration, **6-2**
modify the memory mapper, **6-9**
modulo (integer) operator, **A-13**
monochrome monitor, **3-2**
monochrome monitor /m option, **3-3**
MS-DOS command, **3-16**
MS-DOS system level command execution, **5-2**
multiply (integer) operator, **A-13**

N name characteristic, **4-12**
NAND operator, **A-20**

nesting command files, **7-3**
NOR, intraset logical operator, **A-18**

O
O,o, octal number base specifier, **A-11**
octal number base specifier, **A-11**
one's complement (unary) operator, **A-13**
open a window, **4-5**
operators, **A-12**
  combining intraset and interset, **A-19**
  interset, **A-19**
  intraset, **A-18**
  precedence, **A-13**
options used when accessing the PC Interface, **3-6**
OR (bit-wise) operator, **A-15**
or, interset logical OR operator, **A-19**
OR, intraset logical operator, **A-18**
overlap
  bit-wise merge, **A-16**

P
parity, **2-5**
path for the log file, **3-15**
pattern
  expressions, **A-10**
  labels, **A-18**
PC Interface, **1-1**
physical port name, **2-5**
powerup reset address, **6-5**
precedence, operator, **A-13**
problems, **2-7**
  LAN software, **2-8**
  PC Interface and coresident programs, **2-8**
processor syntax summary, **A-5**

Q
/? option, **3-6**
Q,q, octal number base specifier, **A-11**

R
^ r, **4-19**
ranges, **A-18**
redirect the output of a command, **5-2**
register syntax summary, **A-4**
registers, **6-24**
registers automatically displayed, **6-4**
relational expressions, **A-18 - A-19**

reset button in the target system, **6-3**
reset the memory map, **6-12**
reset the microprocessor, **6-5**
results recorded in a log file, **3-20**
rotate left/right operator, **A-14**
rows, **3-7**
run the microprocessor, **6-2**
run various applications, **3-13**

S
$SYSWIN definition, **5-10**
search a window, **4-10**
set software breakpoints, **6-22**
sets (complex config. trace spec.), **A-18**
setup, **2-1**
shell environment variable, **2-6**
shift left/right operator, **A-14**
size characteristic, **4-12**
smallest window size, **4-12**
software breakpoints, **6-18**
solid highlighted border, **4-4**
start the microprocessor upon CMB events, **6-8**
start the PC Interface, **3-3**
status line, **3-7, 4-4**
step the microprocessor, **6-4**
stop bits, **2-5**
store a window to an ASCII file, **4-10**
store memory, **6-15**
store the PC Interface configuration, **5-9**
storing the memory map, **6-12**
subtraction operator, **A-14**
support services, **1-3**
symbols, **5-4 - 5-5**
syntax for invoking the PC Interface, **3-6**
syntax summary, **A-1**
system features, **5-1**
system fork feature, **3-11**
system level commands, **5-1**
system syntax summary, **A-3**
System Terminal window, **4-17**
System Terminal Window specifics, **4-20**
system window functions, **4-2**
system windows, **3-17, 4-1**

T T,t, decimal number base specifier, **A-11**
target system, **6-27**
target system microprocessor, **6-3**
temporarily remove a window from the screen, **4-8**
Terminal window, **4-17**
tram, **6-10**
transmit/receive pacing, **2-5**
transparency mode setting, **2-5**
trom, **6-10**
truth tables for logical operators, **A-12**
tutorial, **3-12**
two's complement (unary) operator, **A-13**

U $USERWIN definition, **5-10**
unary ones's complement operator, **A-13**
unary two's complement operator, **A-13**
user-defined window, **3-22**
user-defined windows, **4-1, 4-4**
using an editor to create a command file, **7-3**
using command files, **7-1**
using keys to perform various functions, **3-8**
using system features, **5-1**
using the emulator with a target system, **6-27**
using the log command to create command files, **7-4**
using windows, **4-1**

V value expressions, **A-10**
view a window, **4-8**

W /w option, **3-6**
wait, **3-15**
window attributes, **4-4**
window characteristics, **4-11**
window configuration information, **5-10**
window control character summary, **4-11**
window features, **4-5**
window name, **4-12**
window store feature, **4-9**
windows, **4-1**

X XOR (bit-wise) operator, **A-15**

Y Y,y, binary number base specifier, **A-11**

Z ^ z, **3-18, 4-10**
zoom a window, **3-16, 4-9**

Notes