
User's Guide

Graphical User Interface

Notice

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

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

© Copyright 1991, 1992, 1994, Hewlett-Packard Company.

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

HP is a trademark of Hewlett-Packard Company.

OSF/Motif and Motif are trademarks of the Open Software Foundation in the U.S. and other countries.

UNIX (R) is a registered trademark of UNIX System Laboratories Inc. in the U.S.A. and other countries.

Hewlett-Packard
P.O. Box 2197
1900 Garden of the Gods Road
Colorado Springs, CO 80901-2197, U.S.A.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1)(ii) of the Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013. Hewlett-Packard Company, 3000 Hanover Street, Palo Alto, CA 94304 U.S.A. Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

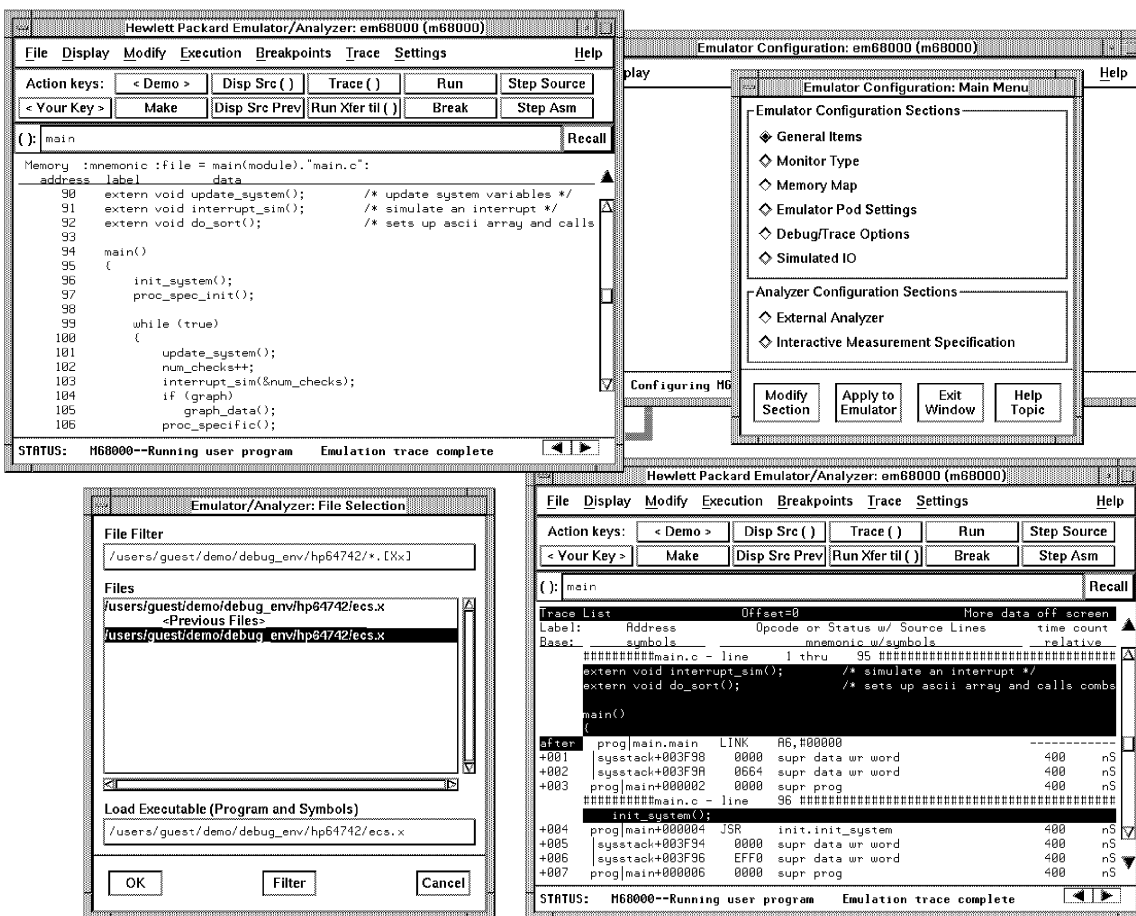
Printing History

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	B1475-97000, July, 1991
Edition 2	B1475-97001, August, 1991
Edition 3	B1475-97002, November, 1991
Edition 4	B1475-97003, March, 1992
Edition 5	B1475-97004, June, 1992
Edition 6	B1475-97005, March 1994

The Graphical User Interface



With the Graphical Emulator/Analyzer Interface, You Can ...

- Use the emulator and analyzer under an X Window System that supports OSF/Motif interfaces.
- Enter commands using pull-down or pop-up menus.
- Enter, recall, and edit commands using the command line pushbuttons.
- Enter file names, recalled commands, recalled values, etc., using dialog boxes.
- Set breakpoints by pointing the mouse cursor on a line in the mnemonic memory display and clicking.
- Create action keys for commonly used commands or command files.
- Configure the emulator and analyzer through a separate configuration interface.
- Open multiple emulator/analyzer interface windows in which you can view different types of information.
- Open windows for other HP 64700 interface products (for example, the C debugger or the software performance analyzer) which can run at the same time as the emulator/analyzer interface.

In This Book

This book describes the Graphical User Interface. It is organized into five parts whose chapters are described below.

Part 1. Quick Start Guide

Chapter 1 presents an overview of the Graphical User Interface and quickly shows you how to use the emulator and analyzer.

Part 2. User's Guide

Chapter 2 shows you how to start and exit the HP 64700 interfaces.

Chapter 3 shows you how to enter commands and use other features provided by the interface.

Chapter 4 shows how to start the configuration interface, use it to configure the emulator and analyzer, and save and re-load configurations.

Chapter 5 shows how to use the emulator and analyzer.

Chapter 6 shows how to customize the Graphical User Interface by setting X resources.

Part 3. Concept Guide

Chapter 7 contains conceptual information on X resources.

Part 4. Installation Guide

Chapter 8 outlines the installation of the Graphical User Interface.

Contents

Part 1 Quick Start Guide

1 Getting Started

The Emulator/Analyzer Interface — At a Glance 18

Graphical User Interface Conventions 20

The Getting Started Tutorial 24

Step 1: Start the demo 25

Step 2: Display the program in memory 26

Step 3: Run from the transfer address 27

Step 4: Step high-level source lines 28

Step 5: Display the previous mnemonic display 29

Step 6: Run until an address 30

Step 7: Display data values 31

Step 8: Display registers 32

Step 9: Step assembly-level instructions 33

Step 10: Trace the program 34

Step 11: Display memory at an address in a register 36

Step 12: Patch assembly language code 37

Step 13: Exit the emulator/analyzer interface 40

Part 2 User's Guide

2 Starting and Exiting HP 64700 Interfaces

Starting the Emulator/Analyzer Interface 45

- To start the emulator/analyzer interface 45
- To start the interface using the default configuration 46
- To run a command file on interface startup 47
- To display the status of emulators 47
- To unlock an interface that was left locked by another user 48

Opening Other HP 64700 Interface Windows 49

- To open additional emulator/analyzer windows 49
- To open the high-level debugger interface window 50
- To open the software performance analyzer (SPA) interface window 50

Exiting HP 64700 Interfaces 51

- To close an interface window 51
- To exit a debug/emulation session 52

3 Entering Commands

Using Menus, the Entry Buffer, and Action Keys 55

- To choose a pulldown menu item using the mouse (method 1) 56
- To choose a pulldown menu item using the mouse (method 2) 57
- To choose a pulldown menu item using the keyboard 57
- To choose popup menu items 59
- To place values into the entry buffer using the keyboard 60
- To copy-and-paste to the entry buffer 60
- To recall entry buffer values 63
- To use the entry buffer 63
- To copy-and-paste from the entry buffer to the command line entry area 64
- To use the action keys 65
- To use dialog boxes 65
- To access help information 69

Using the Command Line	70
To turn the command line on or off	70
To enter a command	71
To edit the command line using the command line pushbuttons	73
To edit the command line using the command line popup menu	74
To edit the command line using the keyboard	75
To recall commands	75
To get help about the command line	76
Using Command Files	77
To start logging commands to a command file	80
To stop logging commands to a command file	80
To playback (execute) a command file	81
Using Pod Commands	82
To display the pod commands screen	83
To use pod commands	83
To copy the pod commands screen to a file	83
Forwarding Commands to Other HP 64700 Interfaces	84
To forward commands to the high-level debugger	84
To forward commands to the software performance analyzer	85
4 Configuring the Emulator/Analyzer	
Configuring the Emulator	88
Using the Fully Graphical Configuration Interface	89
To start the fully graphical configuration interface	90
To modify a configuration section	92
To apply configuration changes to the emulator	94
To store configuration changes to a file	95
To change the configuration directory context	96
To display the configuration context	96
To access help topics	97
To access help for an individual configuration item	97
To exit the configuration interface	98
To load an existing configuration file	98
Mapping memory in the fully graphical configuration interface	99

Using the Softkey-Based Configuration Interface	101
To start the configuration interface	102
To modify a configuration section	104
To store a configuration	106
To change the configuration directory context	107
To display the configuration context	107
To access help information	108
To exit the configuration interface	108
To load a configuration	108
5 Using the Emulator/Analyzer	
Loading Programs and Symbols	111
To load programs	111
To load programs without symbols	112
To load only symbols	112
Using Symbols	113
To display global symbols	113
To display local symbols	114
To display a symbol's parent symbol	116
To copy-and-paste a full symbol name to the entry buffer	117
Using Context Commands	118
To display the current directory and symbol context	119
To change the directory context	120
To change the current working symbol context	120
Displaying and Modifying Memory	121
To display memory	121
To display memory in mnemonic format	122
To return to the previous mnemonic display	122
To display memory in hexadecimal format	123
To display memory in real number format	123
To display memory at an address	123
To display memory repetitively	124
To modify memory	124

Displaying Data Values	125
To display data values	125
To clear the data values display and add a new item	126
To add items to the data values display	126
Displaying and Modifying Registers	127
To display register contents	127
To modify register contents	128
Executing Programs	129
To run programs from the current PC	130
To run programs from an address	130
To run programs from the transfer address	130
To run programs from reset	131
To run programs until an address	131
To step high-level source lines	132
To step assembly-level instructions	133
To stop (break from) program execution	133
To reset the emulation processor	134
Using Breakpoints	135
To display the breakpoints list	136
To enable/disable breakpoints	137
To set a permanent breakpoint	139
To set a temporary breakpoint	141
To set all breakpoints	141
To deactivate a breakpoint	142
To re-activate a breakpoint	142
To clear a breakpoint	144
To clear all breakpoints	145
Tracing Programs	146
To trace after any state	147
To display the trace listing	147
To trace after an address	148
To trace before an address	149
To trace before an address and break on the trigger	150
To trace about an address	151
To qualify which states get stored	152
To prestore states before qualified store states	153

Contents

To trace until the trace is stopped	154
To stop a trace	154
To continually repeat the last trace	155
To recall trace specification commands	155
To store trace data to a file	156
To load a trace data file	156
To store a trace specification to a file	157
To load a trace specification file	157
Changing the Interface Settings	158
To set the source/symbol modes	158
To set the display modes	159
Displaying the Trace List	161
To disassemble the trace list	164
To specify trace dequeuing options	165
To display the trace without disassembly	166
To display symbols in the trace list	166
To display source lines in the trace list	167
To change the column width	167
To select the type of count information in the trace list	168
To offset addresses in the trace list	168
To reset the trace display defaults	169
To display the trace list around a specific line number	169
To change the number of states available for display	170
To display program memory associated with a trace list line	171
To open an edit window into the source file associated with a trace list line	171
Using System Commands	172
To edit files	173
To display the error log	175
To display the event log	176
To display status	177
To copy information to a file or printer	177
To open a terminal emulation window	178
Using Simulated I/O	179
To display the simulated I/O screen	179
To use simulated I/O keyboard input	180

Using Basis Branch Analysis 181
 To store BBA data to a file 181

6 Setting X Resources

To modify the Graphical User Interface resources 186
 To use customized scheme files 190
 To set up custom action keys 192
 To set initial recall buffer values 193
 To set up demos or tutorials 195

Part 3 Concept Guide

7 X Resources and the Graphical User Interface

X Resource Specifications 203
 Resource Names Follow Widget Hierarchy 203
 Class Names or Instance Names Can Be Used 204
 Wildcards Can Be Used 204
 Specific Names Override General Names 205

 How X Resource Specifications are Loaded 206
 Application Default Resource Specifications 206
 User-Defined Resource Specifications 206
 Load Order 207

 Scheme Files 208
 Resources for Graphical User Interface Schemes 208
 Scheme File Names 209
 Load Order for Scheme Files 209
 Custom Scheme Files 210

Part 4 Installation Guide

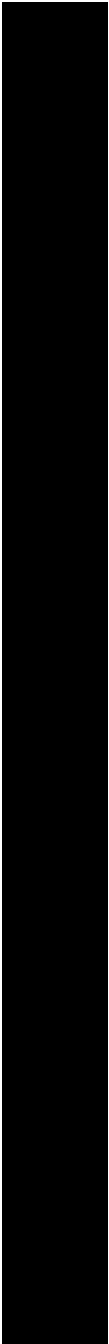
8 Installation

Installation at a Glance	214
Installation Overview for HP 9000 Hosted Systems	214
Installation Overview for Sun SPARCsystems	215
Installation for HP 9000 Hosted Systems	217
Step 1. Install the hardware in the HP 64700 Series Cardcage	217
Step 2. Configure the emulator for the communication channel	217
Step 3. Connect the emulator to your system	218
Step 4. Install the software	218
Step 5. Verify the software installation	220
Step 6a. Start the X server and the Motif Window Manager (mwm)	221
Step 6b. Start HP VUE	221
Step 7. Set the necessary environment variables	222
Step 8. Determine the logical name of your emulator	224
Step 9. Start the interface with the emul700 command	225
Step 10. Exit the Graphical User Interface	227
Installation for Sun SPARCsystems	228
Step 1. Install the hardware in the HP 64700 Series Cardcage	228
Step 2. Configure the emulator for the communication channel	228
Step 3. Connect the emulator to your system	229
Step 4. Install the software	229
Step 5. Start the X server and OpenWindows	230
Step 6. Set the necessary environment variables	230
Step 7. Verify the software installation	232
Step 8. Map your function keys	233
Step 9. Determine the logical name of your emulator	234
Step 10. Start the interface with the emul700 command	235
Step 11. Exit the Graphical User Interface	237

Part 1

Quick Start Guide

Part 1



1



Getting Started

The Emulator/Analyzer Interface — At a Glance

The screenshot shows the Hewlett Packard Emulator/Analyzer interface for the em68000 (m68000) processor. The interface is divided into several sections:

- Menu bar:** File, Display, Modify, Execution, Breakpoints, Trace, Settings, Help.
- Action keys:** A grid of buttons including < Demo >, Disp Src (), Trace (), Run, Step Source, < Your Key >, Make, Disp Src Prev, Run Xfer til (), Break, and Step Asm.
- Entry buffer:** A text field containing the current function name, currently showing "(): main". A Recall button is located to the right.
- Entry buffer recall button:** The Recall button.
- Display area:** A window showing memory contents with columns for address, label, and data. The data column contains C code snippets. A vertical scroll bar is on the right side.
- Scroll bar:** The vertical scroll bar on the right of the display area.
- Status line:** A line showing the current status: "STATUS: cws: main."main.c":".
- Command line:** A text area for entering commands, currently containing "display memory main mnemonic".
- Command line entry area:** The text area for entering commands.
- Softkey pushbuttons:** A row of buttons: run, trace, step, display, modify, break, end, ---ETC--.
- Command buttons, includes command recall button:** A row of buttons: Command: Return, Recall.
- Cursor buttons for command line area control:** A row of buttons: Cursor: Backup, Forward, Clear to end, Clear, Help.

When an X Window System that supports OSF/Motif interfaces is running on the host computer, the default emulator/analyzer interface is the Graphical User Interface which provides pull-down and pop-up menus, point and click setting of breakpoints, cut and paste, on-line help, customizable action keys and pop-up recall buffers, etc.



Menu Bar. Provides pulldown menus from which you select commands. When menu items are not applicable, they appear grayed-out and do not respond to mouse clicks.

Action Keys. User-defined pushbuttons. You can label these pushbuttons and define the action to be performed.

Entry Buffer. Wherever you see "(" in a pulldown menu, the contents of the entry buffer are used in that command. You can type values into the entry buffer, or you can cut and paste values into the entry buffer from the display area or from the command line entry area. You can also set up action keys to use the contents of the entry buffer.

Entry Buffer Recall Button. Allows you to recall entry buffer values that have been predefined or used in previous commands. When you click on the entry buffer **Recall** button, a dialog box appears that allows you to select values.

Display Area. Can show memory, data values, analyzer traces, registers, breakpoints, status, simulated I/O, global symbols, local symbols, pod commands (the emulator's underlying Terminal Interface), error log, or display log.

Whenever the mouse pointer changes from an arrow to a hand, you can press and hold the *select* mouse button to access popup menus. You can click the *select* mouse button to choose the first item in the popup menu

Scroll Bar. A "sticky slider" that allows navigation in the display area. Click on the upper and lower arrows to scroll to the top (home) and bottom (end) of the window. Click on the inner arrows to scroll one line. Drag the slider handle up or down to cause continuous scrolling. Click between the inner arrows and the slider handle to page up or page down.

Status Line. Displays the emulator and analyzer status. Also, when error and status messages occur, they are displayed on the status line in addition to being saved in the error log. You can press and hold the *select* mouse button to access the Status Line popup menu.



Command Line. The command line area is similar to the command line in the Softkey Interface; however, the graphical interface lets you use the mouse to enter and edit commands.

- **Command line entry area.** Allows you to enter commands from the command line.
- **Softkey pushbuttons.** Clicking on these pushbuttons, or pressing softkeys, places the command in the command line entry area. You can press and hold the *select* mouse button to access the Command Line popup menu.
- **Command buttons** (includes command recall button). The command **Return** button is the same as pressing the carriage return key—it sends the command in the command line entry area to the emulator/analyzer.

The command **Recall** button allows you to recall previous or predefined commands. When you click on the command **Recall** button, a dialog box appears that allows you to select a command.

- **Cursor buttons for command line area control.** Allow you to move the cursor in the command line entry area forward or backward, clear to the end of the command line, or clear the whole command line entry area.

You can choose not to display the command line area by turning it off. For the most common emulator/analyzer operations, the pulldown menus, popup menus, and action keys provide all the control you need. Choosing menu items that require use of the command line will automatically turn the command line back on.

Graphical User Interface Conventions

Choosing Menu Commands

This chapter uses a shorthand notation for indicating that you should choose a particular menu item. For example, the following instruction

Choose **File**→**Load**→**Configuration**

means to first display the **File** pulldown menu, then display the **Load** cascade menu, then select the **Configuration** item from the Load cascade menu.

Based on this explanation, the general rule for interpreting this notation can be stated as follows:

- The leftmost item in bold is the pulldown menu label.

- If there are more than two items, then cascade menus are involved and all items between the first and last item have cascade menus attached.
- The last item on the right is the actual menu choice to be made.



Mouse Button and Keyboard Bindings

Because the Graphical User Interface runs on different kinds of computers, which may have different conventions for mouse buttons and key names, the Graphical User Interface supports different bindings and the customization of bindings.

This manual refers to the mouse buttons using general (or "generic") terms. The following table describes the generic mouse button names and shows the default mouse button bindings.

Mouse Button Bindings and Description

Generic Button Name	Bindings:		Description
	HP 9000	Sun SPARCsystem	
<i>paste</i>	left	left	Paste from the display area to the entry buffer.
<i>command paste</i>	middle ¹	middle ¹	Paste from the entry buffer to the command line text entry area.
<i>select</i>	right	right	Click selects first item in popup menus. Press and hold displays menus.
<i>command select</i>	left	right	Displays pulldown menus.
<i>pushbutton select</i>	left	left	Actuates pushbuttons outside of the display area.

¹ Middle button on three-button mouse. Both buttons on two-button mouse.

The following tables show the default keyboard bindings.

Keyboard Key Bindings

Generic Key Name	HP 9000	Sun SPARCsystem
menu select	extend char	extend char
insert	insert char	insert char
delete	delete char	delete char
left-arrow	left arrow	left arrow
right-arrow	right arrow	right arrow
up-arrow	up arrow	up arrow
down-arrow	down arrow	down arrow
escape	escape	escape
TAB	TAB	TAB





The Getting Started Tutorial

This tutorial gives you step-by-step instructions on how to perform a few basic tasks using the emulator/analyzer interface.

The tutorial examples presented in this chapter make the following assumptions:

- The emulator and analyzer are installed into the HP 64700A Card Cage, the HP 64700A is connected to the host computer, and the emulator/analyzer interface software has been installed as outlined in the "Installation" chapter.
- The emulator has enough emulation memory for the demo program and is able to run out-of-circuit (that is, not plugged into the target system) or is plugged into a demo target system.

The Demonstration Program

The demonstration program used in this chapter is a simple environmental control system. The program controls the temperature and humidity of a room requiring accurate environmental control.

Step 1. Start the demo

A demo program and its associated files are provided with the Graphical User Interface.

- 1 Change to the demo directory.

For example, if your Graphical User Interface is for the HP 64742/3/4 family of 68000 emulators:

```
$ cd /usr/hp64000/demo/debug_env/hp64742 <RETURN>
```

Refer to the README file for more information on the demo program.

- 2 Check that "/usr/hp64000/bin" and "." are in your PATH environment variable. To see the value of PATH:

```
$ echo $PATH <RETURN>
```

- 3 If the Graphical User Interface software is installed on a different type of computer than the computer you are using, edit the "platformScheme" resource setting in the "Xdefaults.emul" file.

For example, if the Graphical User Interface will be run on a HP 9000 computer and displayed on a Sun SPARCsystem computer, change the platform scheme to "SunOS".

- 4 Start the emulator/analyzer demo.

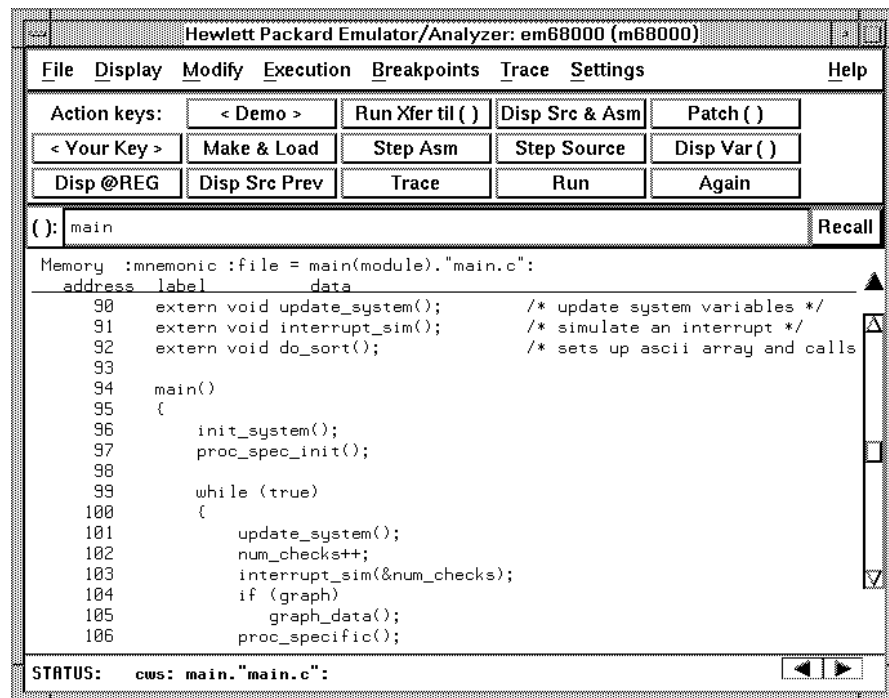
```
$ Startemul <logical_emul_name> <RETURN>
```

This script starts the emulator/analyzer interface (with a customized set of action keys), loads a configuration file for the demo program, and then loads the demo program.

The <logical_emul_name> in the command above is the logical emulator name given in the HP 64700 emulator device table file (/usr/hp64000/etc/64700tab.net).

Step 2: Display the program in memory

- 1 If the symbol "main" is not already in the entry buffer, move the mouse pointer to the entry buffer (notice the flashing I-beam cursor) and type in "main".
- 2 Choose **Display**→**Memory**→**Mnemonic ()**.



The screenshot shows the Hewlett Packard Emulator/Analyzer window for em68000 (m68000). The menu bar includes File, Display, Modify, Execution, Breakpoints, Trace, Settings, and Help. The Action keys section contains buttons for < Demo >, Run Xfer til (), Disp Src & Asm, Patch (), < Your Key >, Make & Load, Step Asm, Step Source, Disp Var (), Disp @REG, Disp Src Prev, Trace, Run, and Again. The entry buffer shows (): main with a Recall button. The memory display shows the following code:

```
Memory :mnemonic :file = main(module)."main.c":
address label data
90 extern void update_system(); /* update system variables */
91 extern void interrupt_sim(); /* simulate an interrupt */
92 extern void do_sort(); /* sets up ascii array and calls
93
94 main()
95 {
96     init_system();
97     proc_spec_init();
98
99     while (true)
100     {
101         update_system();
102         num_checks++;
103         interrupt_sim(&num_checks);
104         if (graph)
105             graph_data();
106         proc_specific();
```

The STATUS bar at the bottom indicates: STATUS: cws: main."main.c":

The default display mode settings cause source lines and symbols to appear in displays where appropriate.

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

Step 3: Run from the transfer address

The transfer address is the entry address defined by the software development tools and included with the program's symbol information.

- Click on the **Run Xfer til ()** action key.

```
Memory :@sp :mnemonic :file = main(module)."main.c":
address label      data
 90  extern void update_system();      /* update system variables */
 91  extern void interrupt_sim();      /* simulate an interrupt */
 92  extern void do_sort();            /* sets up ascii array and calls
 93
 94  main()
> 95  {
 96      init_system();
 97      proc_spec_init();
 98
 99      while (true)
100      {
101          update_system();
102          num_checks++;
103          interrupt_sim(&num_checks);
104          if (graph)
105              graph_data();
106          proc_specific();
STATUS:  M68000--Running in monitor      Software break: 0000fd2@sp
```

Notice the message "Software break: <address>" is displayed on the status line and that the emulator is "Running in monitor". When you run until an address, a breakpoint is set at the address before the program is run.

Notice the highlighted bar on the screen; it shows the current program counter.

Step 4: Step high-level source lines

You can step through the program by high-level source lines. The emulator executes as many instructions as are associated with the high-level program source lines.

- 1 To step a source line from the current program counter, click on the **Step Source** action key.

Notice that the highlighted bar (the current program counter) moves to the next high-level source line.

- 2 Step into the "init_system" function by continuing to step source lines, either by clicking on the **Step Source** action key or by clicking on the **Again** action key which repeats the previous command, or by choosing **Execution**→**Step Source**→**from PC**.

```
Memory :@sp :mnemonic :file = init_system(module).\"init_system.c\":
address  label      data
-----
26
27  void init_val_arr();
28
29  void
30  init_system()
> 31  { /* FUNCTION init_system() */
32      /* Initialize the target values for temperature and humidity */
33      target_temp = 73;
34      target_humid = 45;
35
36      /* Intialize the variables indicating the current environment */
37      /* conditions */
38      current_temp = 68;
39      current_humid = 41;
40
41      /* Set starting directions for temp and humid */
42      temp_dir = up;
STATUS:  cws: init_system.\"init_system.c\":
```

Step 5: Display the previous mnemonic display

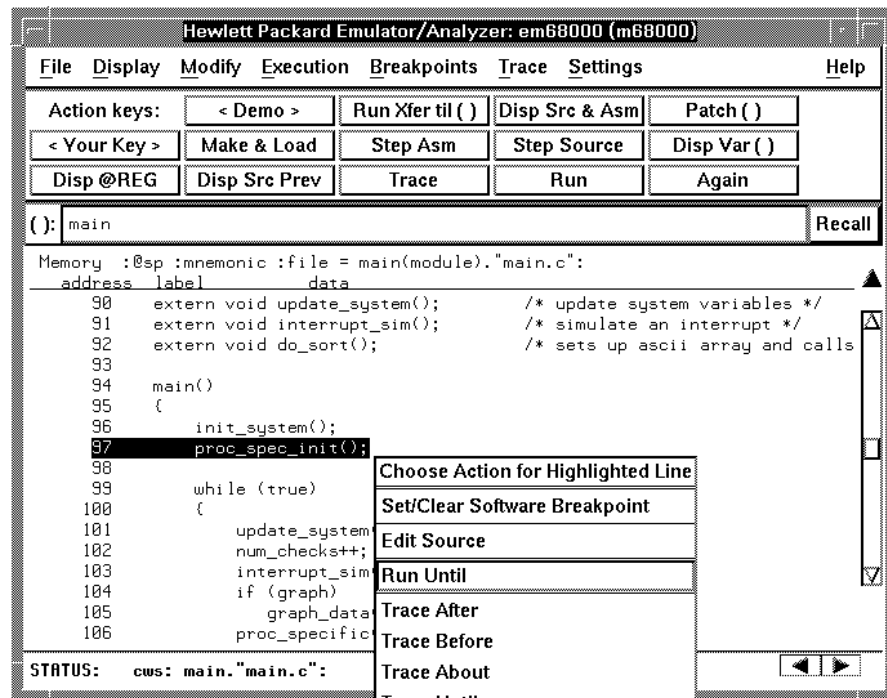
- Click on the **Disp Src Prev** action key, or choose **Display→Memory→Mnemonic Previous**.

The action key or command are useful, for example, when you have stepped into a function that you do not wish to look at—you can display the previous mnemonic display and run until the source line that follows the function call.

Step 6: Run until an address

When displaying memory in mnemonic format, a selection in the popup menu lets you run from the current program counter address until a specific source line.

- Position the mouse pointer over the line "proc_spec_init();", press and hold the *select* mouse button, and choose **Run Until** from the popup menu.



After the command has executed, notice the highlighted bar indicates the program counter has moved to the specified source line.

Step 7: Display data values

- 1 Position the mouse pointer over "num_checks" in the source line that reads "num_checks++;" and click the *paste* mouse button (notice "num_checks" is cut and pasted into the entry buffer).
- 2 Click on the **Disp Var ()** action key, or choose **Display→Data Values→Add()→int32**.

Data :update			
address	label	type	data
0076F4	_num_checks	int32	0

The "num_checks" variable is added to the data values display and its value is displayed as a 32-bit integer.

Step 8: Display registers

You can display the contents of the processor registers.

- Choose **Display**→**Registers** or **Display**→**Registers**→**BASIC** if there are several register classes to choose from.

```
Registers
-----
Next_PC 000FDC@sp
PC 0000FDC STATUS 2704 < s z > USP 00000000 SSP 0000EF94
D0-D7 00000174 0000001F 00008030 00000670 00000000 00000000 00000000 00000000
A0-A7 00008020 FFFFFFFF 00007730 00008028 000077F0 0000F158 0000EF94 0000EF94
```


Step 9: Step assembly-level instructions

You can step through the program one instruction at a time.

- To step one instruction from the current program counter, click on the **Step Asm** action key, or choose **Execution**→**Step Instruction**→**from PC**.

```
Registers
-----
Next_PC 000FDC@sp
PC 0000FDC STATUS 2704 < s z > USP 00000000 SSP 0000EF94
D0-D7 00000174 0000001F 00008030 00000670 00000000 00000000 00000000 00000000
A0-A7 00008020 FFFFFFFF 00007730 00008028 000077F0 0000F158 0000EF94 0000EF94

Step_PC 000FDC@sp JSR p.proc_spec_init
Next_PC 001BDC@sp
PC 00001BDC STATUS 2704 < s z > USP 00000000 SSP 0000EF90
D0-D7 00000174 0000001F 00008030 00000670 00000000 00000000 00000000 00000000
A0-A7 00008020 FFFFFFFF 00007730 00008028 000077F0 0000F158 0000EF94 0000EF90
```

Notice, when registers are displayed, stepping causes the assembly language instruction just executed to be displayed.

Step 10: Trace the program

When the analyzer traces program execution, it looks at the data on the emulation processor's bus and control signals at each clock cycle. The information seen at a particular clock cycle is called a state.

When one of these states matches the "trigger state" you specify, the analyzer stores states in trace memory. When trace memory is filled, the trace is said to be "complete."

- 1 Click on the **Recall** button to the right of the entry buffer.

A selection dialog box appears. You can select from entry buffer values that have been entered previously or that have been predefined.

- 2 Click on "main" in the selection dialog box, and click the "OK" pushbutton.

Notice that the value "main" has been returned to the entry buffer.

- 3 To trigger on the address "main" and store states that occur after the trigger, choose **Trace→After ()**.

Notice the message "Emulation trace started" appears on the status line. This shows that the analyzer has begun to look for the trigger state which is the address "main" on the processor's address bus.

- 4 Run the emulator demo program from its transfer address by choosing **Execution→Run→from Transfer Address**.

Notice that now the message on the status line is "Emulation trace complete". This shows the trigger state has been found and the analyzer trace memory has been filled.

- 5 To view the captured states, choose **Display→Trace**.

Chapter 1: Getting Started
Step 10: Trace the program

Trace List		Offset=0		More data off screen	
Label:	Address	Opcode or Status w/	Source Lines	time	count
Base:	_____	_____	_____	_____	_____
	symbols	mnemonic w/symbols	relative		
#####main.c - line 1 thru 95 #####					
		extern void interrupt_sim();	/* simulate an interrupt */		
		extern void do_sort();	/* sets up ascii array and calls combs		
		main()			
		{			
after	prog main.main	LINK	R6,#00000	-----	
+001	sysstack+003F98	0000	supr data wr word	400	nS
+002	sysstack+003F9A	0664	supr data wr word	400	nS
+003	prog main+000002	0000	supr prog	400	nS
#####main.c - line 96 #####					
		init_system();			
+004	prog main+000004	JSR	init.init_system	400	nS
+005	sysstack+003F94	0000	supr data wr word	400	nS
+006	sysstack+003F96	EFF0	supr data wr word	400	nS
+007	prog main+000006	0000	supr prog	400	nS

The default display mode settings cause source lines and symbols to appear in the trace list.

Captured states are numbered in the left-hand column of the trace list. Line 0 always contains the state that caused the analyzer to trigger.

Other columns contain address information, data values, opcode or status information, and time count information.

Step 11: Display memory at an address in a register

- 1 Click on the **Disp @REG** action key.

A command file dialog box appears.

- 2 Move the mouse pointer to the dialog box text entry area, type in the register name, and click on the **OK** button.

For example, when using the 68000 emulator, you can type in the register name "A7" to look at the contents of the stack.

Memory	:@sp	:bytes	:blocked	:update					:ascii
address	data		:hex						
00EF54-5B	00	00	2A	F8	00	00	03	E8	. . *
00EF5C-63	00	00	2A	F8	00	00	00	00	. . *
00EF64-6B	00	00	00	00	00	00	76	F0 v .
00EF6C-73	00	00	76	F2	00	00	10	4E	. . v N
00EF74-7B	00	00	10	64	00	00	00	30	. . . d . . . 0
00EF7C-83	00	00	06	70	00	00	00	00	. . . p
00EF84-8B	00	00	77	30	00	00	EF	94	. . w 0
00EF8C-93	00	00	0F	FC	00	00	76	F4 v .
00EF94-9B	00	00	EF	F0	00	00	06	64 d
00EF9C-A3	00	00	00	00	00	00	00	00
00EFA4-AB	00	00	00	00	00	00	77	F0 w .
00EFAC-B3	00	00	00	00	00	00	00	30 0
00EFB4-BB	00	00	00	02	00	00	00	15
00EFBC-C3	00	00	00	01	00	00	00	15
00EFC4-CB	00	00	00	00	00	00	00	40 0
00EFC4-D3	00	00	77	30	00	00	00	00	. . w 0
00EFD4-DB	00	00	00	C0	00	00	00	30 0

Step 12: Patch assembly language code

The **Patch ()** action key lets you patch code in your program.

- 1 With "main" still in the entry buffer, click on the **Run Xfer til ()** action key.
- 2 To display memory with assembly-level instructions intermixed with the high-level source lines, click on the **Disp Src & Asm** action key.

```
Memory :@sp :mnemonic :file = main(module)."main.c":
address  label      data
  91  extern void interrupt_sim();      /* simulate an interrupt */
  92  extern void do_sort();           /* sets up ascii array and calls
  93
  94  main()
  95  {
> 000FD2  pr|main.main 4E560000  LINK  A6,#00000
  96      init_system();
000FD6      4EB9000015  JSR   init.init_system
  97      proc_spec_init();
000FDC      4EB900001B  JSR   p.proc_spec_init
  98
  99      while (true)
000FE2      4E71        NOP
 100      {
 101          update_system();
000FE4      4EB9000016  JSR   up.update_system
 102          num_checks++;
```

- 3 Click on the **Patch ()** action key.

A window appears and the **vi** editor is started. Add the line:

```
LINK A6,#1234h
```

Exit out of the editor, saving your changes.

The file you just edited is assembled, and the patch main menu appears. Type "a <RETURN>" to apply the patch.

Chapter 1: Getting Started

Step 12: Patch assembly language code

```
Memory :@sp :mnemonic :file = main(module)."main.c":
address  label      data
  91     extern void interrupt_sim();      /* simulate an interrupt */
  92     extern void do_sort();             /* sets up ascii array and calls
  93
  94     main()
  95     {
> 000FD2  pr|main.main 4E561234  LINK  A6,#01234
  96         init_system();
000FD6         4EB9000015 JSR    init.init_system
  97         proc_spec_init();
000FDC         4EB900001B JSR    p.proc_spec_init
  98
  99         while (true)
000FE2         4E71      NOP
 100         {
 101             update_system();
000FE4         4EB9000016 JSR    up.update_system
 102             num_checks++;
```

Notice in the emulator/analyzer interface that the instruction at address "main" has changed.

- 4 Click on the **Patch ()** action key again.

A window running the **vi** editor again appears, allowing you to modify the patch code that was just created. Modify the line you added previously to:

```
LINK A6,#0
```

Exit out of the editor, saving your changes.

The file you just edited is assembled, and the patch main menu appears. Type "a <RETURN>" to apply the patch.

Notice in the emulator/analyzer interface that the instruction at address "main" has been changed back to what it was originally.

When patching a single address, make sure the new instruction takes up the same number of bytes as the old instruction; otherwise, you may inadvertently modify code that follows.

5 Type "main+4 thru main+15" in the entry buffer.

By entering an address range in the entry buffer (that is, <address> thru <address>) before clicking on the **Patch ()** action key, you can modify a patch template file which allows you to insert as much or as little code as you wish.

6 Click on the **Patch ()** action key again.

A window running the **vi** editor again appears. Suppose you want to patch the demo program so that the `proc_spec_init()` function is called before the `init_system()` function. Suppose also that there is memory available at address 8800H. Edit the patch template file as shown below.

```
; PCHS700 Assembly Patch File: PCHmain+4.s
;
; Date : Tue Jun 30 14:06:06 MDT 1992
; Dir  : /users/guest/demo/debug_env/hp64742
; Owner: guest
;
;   INCLUDE PCHSINC.s
;   ORG main+4
;   BRA patch1 ;You may want to change this name!
;   ORG 8800h ;You MUST set this address!
patch1 NOP
; !!!!!!!!! You may need to modify labels and operands of the !!!!!!!!
; !!!!!!!!! following code to match your assembler syntax !!!!!!!!
; !!!!!!!!! Patching Range: main+4 thru main+15
; !!!!!!!!! Insert new code here !!!!!!!!
;   JSR _proc_spec_init
;   JSR _init_system
;   BRA main+16 ;You MUST set this address also!
```

Notice that symbols can be used in the patch file. Exit out of the editor, saving your changes.

The file you just edited is assembled, and the patch main menu appears. Type "a <RETURN>" to apply the patch.

You can step through the program to view execution of the patch.





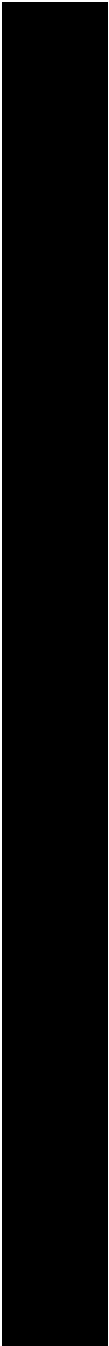
Step 13: Exit the emulator/analyzer interface

- To exit the emulator/analyzer interface and release the emulator, choose **File→Exit→Released**.

Part 2

User's Guide

Part 2





**Starting and Exiting HP 64700
Interfaces**

Starting and Exiting HP 64700 Interfaces

You can use several types of interfaces to the same emulator at the same time to give yourself different views into the target system.

The strength of the emulator/analyzer interface is that it lets you perform the real-time analysis measurements that are helpful when integrating hardware and software.

The C debugger interface (which is a separate product) lets you view the stack backtrace and high-level data structures, and it lets you use C language expressions and macros. These features are most useful when debugging software.

The Software Performance Analyzer interface (which is also a separate product) lets you make measurements that can help you improve the performance of your software.

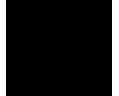
These interfaces can operate at the same time with the same emulator. When you perform an action in one of the interfaces, it is reflected in the other interfaces.

Up to 10 interface windows may be started for the same emulator. Only one C debugger interface window and one SPA window are allowed, but you can start multiple emulator/analyzer interface windows.

The tasks associated with starting and exiting HP 64700 interfaces are grouped into the following sections:

- Starting the emulator/analyzer interface.
- Opening other HP 64700 interface windows.
- Exiting HP 64700 interfaces.

Starting the Emulator/Analyzer Interface



Before starting the emulator/analyzer interface, the emulator and interface software must have already been installed as described in the "Installation" chapter.

This section describes how to:

- Start the interface.
- Start the interface using the default configuration.
- Run a command file on interface startup.
- Display the status of emulators defined in the 64700tab.net file.
- Unlock an interface that was left locked by another user.

To start the emulator/analyzer interface

- Use the **emul700 <emul_name>** command.

If **/usr/hp64000/bin** is specified in your PATH environment variable (as shown in the "Installation" chapter), you can start the interface with the **emul700 <emul_name>** command. The "emul_name" is the logical emulator name given in the HP 64700 emulator device table (**/usr/hp64000/etc/64700tab.net**).

If you are running a window system on your host computer (for example, the X Window System), you can run the interface in up to 10 windows. This capability provides you with several views into the emulation system. For example, you can display memory in one window, registers in another, an analyzer trace in a third, and data in the fourth.

Chapter 2: Starting and Exiting HP 64700 Interfaces

Starting the Emulator/Analyzer Interface

Examples

To start the emulator/analyzer interface for the 68000 emulator:

```
$ emul700 em68000 <RETURN>
```

The "em68000" in the command above is the logical emulator name given in the HP 64700 emulator device table file (/usr/hp64000/etc/64700tab.net).

```
# Blank lines and the rest of each line after a '#' character are ignored.
# The information in each line must be in the specified order, with one line
# for each HP series 64700 emulator. Use blanks or tabs to separate fields.
#
#-----+-----+-----+-----+
# Channel| Logical | Processor | Remainder of Information for the Channel
# Type | Name | Type | (IP address for LAN connections)
#-----+-----+-----+-----+
# lan:   em68000 m68000 21.17.9.143
# serial: em68000 m68000 myhost /dev/emcom23 OFF 9600 NONE XON 2 8
```

If you're currently running the X Window System, the Graphical User Interface starts; otherwise, the Softkey Interface starts.

The status message shows that the default configuration file has been loaded.

To start the interface using the default configuration

- Use the **emul700 -d <emul_name>** command.

In the **emul700 -d <emul_name>** command, the **-d** option says to use the default configuration. The **-d** option is ignored if the interface is already running in another window or on another terminal.

To run a command file on interface startup

- Use the **emul700 -c <cmd_file> <emul_name>** command.

You can cause command files to be run upon starting the interface by using the **-c <cmd_file>** option to the **emul700** command.

Refer to the "Using Command Files" section in the "Entering Commands" chapter for information on creating command files.

Examples

To start the emulator/analyzer interface and run the "startup" command file:

```
$ emul700 -c startup em68000 <RETURN>
```

To display the status of emulators

- Use the **emul700 -l** or **emul700 -lv** command.

The **-l** option of the **emul700** command lists the status of all emulators defined in the 64700tab and 64700tab.net files. If a logical emulator name is included in the command, just the status of that emulator is listed.

You can also use the **-v** option with the **-l** option for a verbose listing of the status information.

Examples

To list, verbosely, the status of the emulator whose logical name is "em68000":

```
$ emul700 -lv em68000 <RETURN>
```

The information may be similar to:

```
em68000 - m68000 running; user = guest  
description: M68000 emulation, w/64740EA, 60Kb emul mem  
user interfaces: xdebug, xemul, xperf, skemul, sktiming  
device channel: /dev/emcom23
```

Chapter 2: Starting and Exiting HP 64700 Interfaces

Starting the Emulator/Analyzer Interface

Or, the information may be similar to:

```
em68000 - m68000 running; user = guest@myhost
description:  M68000 emulation, w/64740EA, 60Kb emul mem
user interfaces:  xdebug, xemul, xperf, skemul, sktiming
internet address: 21.17.9.143
```

To unlock an interface that was left locked by another user

- Use the **emul700 -U <emul_name>** command.

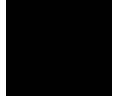
The **-U** option to the **emul700** command may be used to unlock the emulators whose logical names are specified. This command will fail if there currently is a session in progress.

Examples

To unlock the emulator whose logical name is "em68000":

```
$ emul700 -U em68000 <RETURN>
```


Opening Other HP 64700 Interface Windows



The **File**→**Emul700** menu lets you open additional emulator/analyzer interface windows or other HP 64700 interface windows if those products have been installed (for example, the software performance analyzer (SPA) interface and the high-level debugger interface).

This section shows you how to:

- Open additional emulator/analyzer interface windows.
- Open the high-level debugger interface window.
- Open the software performance analyzer (SPA) interface window.

To open additional emulator/analyzer windows

- To open additional Graphical User Interface windows, choose **File**→**Emul700**→**Emulator/Analyzer** under *Graphic Windows*
- To open additional conventional Softkey Interface windows, choose **File**→**Emul700**→**Emulator/Analyzer** under *Terminal Windows*.
- Enter the **emul700 <emul_name>** command in another terminal emulation window.

You can open additional Graphical User Interface windows, or terminal emulation windows containing the Softkey Interface.

When you open an additional window, the status line will show that this session is joining a session already in progress, and the event log is displayed.

You can enter commands in any window in which the interface is running. When you enter commands in different windows, the command entered in the first

window must complete before the command entered in the second window can start. The status lines and the event log displays are updated in all windows.

To open the high-level debugger interface window

- Choose **File**→**Emul700**→**High-Level Debugger ...** under "Graphic Windows".

For information on how to use the high-level debugger interface, refer to the debugger/emulator *User's Guide*.

To open the software performance analyzer (SPA) interface window

- Choose **File**→**Emul700**→**Performance Analyzer ...** under "Graphic Windows".

For information on how to use the software performance analyzer, refer to the *Software Performance Analyzer User's Guide*.

Exiting HP 64700 Interfaces

There are several options available when exiting the HP 764700 interfaces. You can simply close one of the open interface windows, or you can exit the debug session by closing all the open windows. When exiting the debug session, you can lock the emulator so that you can continue later, or you can release the emulation system so that others may use it. This section describes how to:

- Close an interface window.
- Exit a debug/emulation session.

To close an interface window

- In the interface window you wish to close, choose **File→Exit→Window**.

All other interface windows remain open, and the emulation session continues, unless the window closed is the only one open for the emulation session. In that case, closing the window ends the emulation session, but locks the emulator so that other users cannot access it.

To exit a debug/emulation session

- To exit the interface, save your configuration to a temporary file, and lock the emulator so that it cannot be accessed by other users, choose **File→Exit→Locked**.
- To exit the interface and release the emulator for access by other users, choose **File→Exit→Released**.

If you exit the interface locked, the interface saves the current configuration to a temporary file and locks the emulator to prevent other users from accessing it. When you again start the interface with the **emul700** command, the temporary file is reloaded, and therefore, you return to the configuration you were using when you quit the interface locked.

Also saved when you exit the interface locked are the contents of the entry buffer and command recall buffer. These recall buffer values will be present when you restart the interface.


In contrast, if you end released, you must have saved the current configuration to a configuration file (if the configuration has changed), or the changes will be lost.

3



Entering Commands

Entering Commands



This chapter shows you how to use the components of the Graphical User Interface to enter emulator/analyzer commands.

These tasks are grouped into the following sections:

- Using menus, the entry buffer, and action keys.
- Using the command line.
- Using command files.
- Using pod commands.
- Forwarding commands to other HP 64700 interfaces.

Using Menus, the Entry Buffer, and Action Keys

This section describes the tasks you perform when using the Graphical User Interface to enter commands. This section describes how to:

- Choose a pulldown menu item using the mouse.
- Choose a pulldown menu item using the keyboard.
- Use the popup menus.
- Use the entry buffer.
- Copy and paste to the entry buffer.
- Use action keys.
- Use dialog boxes.
- Access help information.



To choose a pulldown menu item using the mouse (method 1)

- 1 Position the mouse pointer over the name of the menu on the menu bar.
- 2 Press and hold the *command select* mouse button to display the menu.
- 3 While continuing to hold down the mouse button, move the mouse pointer to the desired menu item. If the menu item has a cascade menu (identified by an arrow on the right edge of the menu button), then continue to hold the mouse button down and move the mouse pointer toward the arrow on the right edge of the menu. The cascade menu will display. Repeat this step for the cascade menu until you find the desired menu item.
- 4 Release the mouse button to select the menu choice.

If you decide not to select a menu item, simply continue to hold the mouse button down, move the mouse pointer off of the menu, and release the mouse button.

Some menu items have an ellipsis ("...") as part of the menu label. An ellipsis indicates that the menu item will display a dialog or message box when the menu item is chosen.

To choose a pulldown menu item using the mouse (method 2)

- 1 Position the mouse pointer over the menu name on the menu bar.
- 2 Click the *command select* mouse button to display the menu.
- 3 Move the mouse pointer to the desired menu item. If the menu item has a cascade menu (identified by an arrow on the right edge of the menu button), then repeat the previous step and then this step until you find the desired item.
- 4 Click the mouse button to select the item.

If you decide not to select a menu item, simply move the mouse pointer off of the menu and click the mouse button.

Some menu items have an ellipsis ("...") as part of the menu label. An ellipsis indicates that the menu item will display a dialog or other box when the menu item is chosen.

To choose a pulldown menu item using the keyboard

- To initially display a pulldown menu, press and hold the **menu select** key (for example, the "Extend char" key on a HP 9000 keyboard) and then type the underlined character in the menu label on the menu bar. (For example, "f" for "File". Type the character in lower case only.)
- To move right to another pulldown menu after having initially displayed a menu, press the **right-arrow** key.

Chapter 3: Entering Commands

Using Menus, the Entry Buffer, and Action Keys

- To move left to another pulldown menu after having initially displayed a menu, press the **left-arrow** key.
- To move down one menu item within a menu, press the **down-arrow** key.
- To move up one menu item within a menu, press the **up-arrow** key.
- To choose a menu item, type the character in the menu item label that is underlined. Or, move to the menu item using the arrow keys and then press the **<RETURN>** key on the keyboard.
- To cancel a displayed menu, press the **Escape** key.

The interface supports keyboard mnemonics and the use of the arrow keys to move within or between menus. For each menu or menu item, the underlined character in the menu or menu item label is the keyboard mnemonic character. Notice the keyboard mnemonic is not always the first character of the label. If a menu item has a cascade menu attached to it, then typing the keyboard mnemonic displays the cascade menu.

Some menu items have an ellipsis ("...") as part of the menu label. An ellipsis indicates that the menu item will display a dialog or other box when the menu item is chosen.

Dialog boxes support the use of the keyboard as well. To direct keyboard input to a dialog box, you must position the mouse pointer somewhere inside the boundaries of the dialog box. That is because the interface *keyboard focus policy* is set to *pointer*. That just means that the window containing the mouse pointer receives the keyboard input.

In addition to keyboard mnemonics, you can also specify keyboard accelerators which are keyboard shortcuts for selected menu items. Refer to the "Setting X Resources" chapter and the "Softkey.Input" scheme file for more information about setting the X resources that control defining keyboard accelerators.

To choose popup menu items

- 1 Move the mouse pointer to the area whose popup menu you wish to access. (If a popup menu is available, the mouse pointer changes from an arrow to a hand.)
- 2 Press and hold the *select* mouse button.
- 3 After the popup menu appears (while continuing to hold down the mouse button), move the mouse pointer to the desired menu item.
- 4 Release the mouse button to select the menu choice.

If you decide not to select a menu item, simply continue to hold the mouse button down, move the mouse pointer off of the menu, and release the mouse button.

The following popup menus are available in the Graphical User Interface:

- Mnemonic Memory Display.
- Breakpoints Display.
- Global Symbols Display.
- Local Symbols Display.
- Status Line.
- Command Line.



To place values into the entry buffer using the keyboard

- 1 Position the mouse pointer within the text entry area. (An "I-beam" cursor will appear.)
- 2 Enter the text using the keyboard.

To clear the entry buffer text area from beginning until end, press the <CTRL>u key combination.

To copy-and-paste to the entry buffer

- To copy and paste a discrete text string as determined by the interface, position the mouse pointer over the text to copy and click the *paste* mouse button.
- To specify the exact text to copy to the entry buffer: press and hold the *paste* mouse button; drag the mouse pointer to highlight the text to copy-and-paste; release the *paste* mouse button.

You can copy-and-paste from the display area, the status line, and from the command line entry area.

When you position the pointer and click the mouse button, the interface expands the highlight to include the most complete text string it considers to be discrete. Discrete here means that the interface will stop expanding the highlight in a given direction when it discovers a delimiting character not determined to be part of the string. A common delimiter would, of course, be a space.

When you press and hold the mouse button and drag the pointer to highlight text, the interface copies all highlighted text to the entry buffer when you release the mouse button.

Because the interface displays absolute addresses as hex values, any copied and pasted string that can be interpreted as a hexadecimal value (that is, the string

Chapter 3: Entering Commands Using Menus, the Entry Buffer, and Action Keys

contains only numbers 0 through 9 and characters "a" through "f") automatically has an "h" appended.

Note

If you have multiple Graphical User Interface windows open, a copy-and-paste action in any window causes the text to appear in all entry buffers in all windows. That is because although there are a number of entry buffers being displayed, there is actually only one entry buffer and it is common to all windows. That means you can copy a symbol or an address from one window and then use it in another window.

On a memory display or trace display, a symbol may not be completely displayed because there are too many characters to fit into the width limit for a particular column of the display. To make a symbol usable for copy-and-paste, you can scroll the screen left or right to display all, or at least more, of the characters from the symbol. The interface displays absolute addresses as hex values.

Text pasted into the entry buffer replaces that which is currently there. You cannot use paste to append text to existing text already in the entry buffer.

See "To copy-and-paste from the entry buffer to the command line entry area" for information about pasting the contents of the entry buffer into the command line entry area.

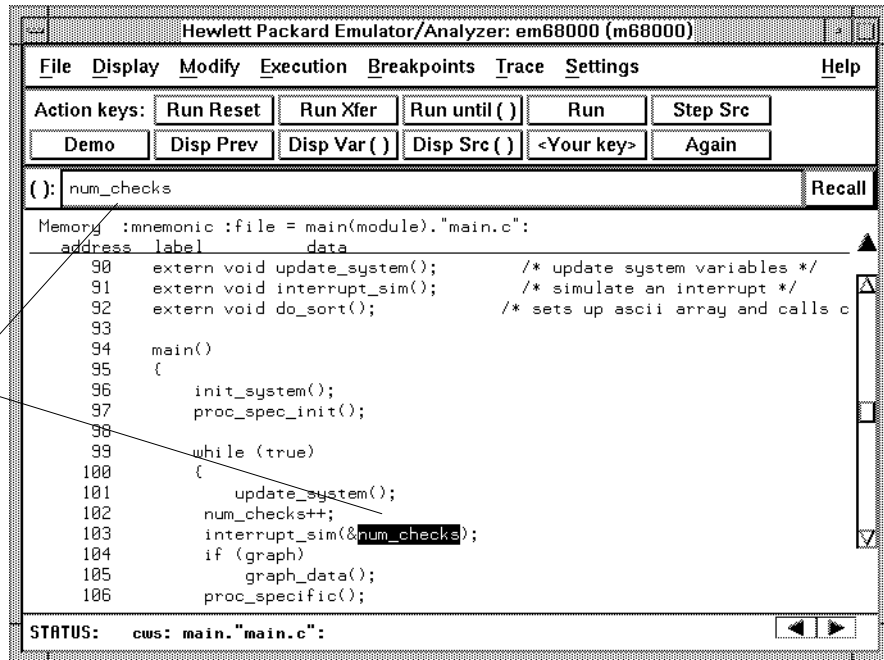


Chapter 3: Entering Commands Using Menus, the Entry Buffer, and Action Keys

Example

To paste the symbol "num_checks" into the entry buffer from the interface display area, position the mouse pointer over the symbol and then click the paste mouse button.

A mouse click causes the interface to expand the highlight to include the symbol "num_checks" and paste the symbol into the entry buffer.



To recall entry buffer values

- Position the mouse pointer over the **Recall** button just to the right of the entry buffer text area, click the mouse button to bring up the Entry Buffer Recall dialog box, and then choose a string from that dialog box.

The Entry Buffer Recall dialog box contains a list of entries gained during the emulation session as well as any predefined entries present at interface startup.

If you exit the emulation/analysis session with the interface "locked", recall buffer values are saved and will be present when you restart the interface.

You can predefine entries for the Entry Buffer Recall dialog box and define the maximum number of entries by setting X resources (refer to the "Setting X Resources" chapter).

See the following "To use dialog boxes" section for information about using dialog boxes.

To use the entry buffer

- 1 Place information into the entry buffer (see the previous "To place values into the entry buffer using the keyboard", "To copy-and-paste to the entry buffer", or "To recall entry buffer values" task descriptions).
- 2 Choose the menu item, or click the action key, that uses the contents of the entry buffer (that is, the menu item or action key that contains "()").

To copy-and-paste from the entry buffer to the command line entry area

- 1 Place text to be pasted into the command line in the entry buffer text area.

You may do that by:

- Copying the text from the display area using the copy-and-paste feature.
 - Enter the text directly by typing it into the entry buffer text area.
 - Choose the text from the entry buffer recall dialog box.
- 2 Position the mouse pointer within the command line text entry area.
 - 3 If necessary, reposition the cursor to the location where you want to paste the text.
 - 4 If necessary, choose the insert or replace mode for the command entry area by pressing the <Insert> key.
 - 5 Click the *command paste* mouse button to paste the text in the command line entry area at the current cursor position.

The entire contents of the entry buffer are pasted into the command line at the current cursor position.

Although a paste from the display area to the entry buffer affects all displayed entry buffers in all open windows, a paste from the entry buffer to the command line only affects the command line of the window in which you are currently working.

See "To copy-and-paste to the entry buffer" for information about pasting information from the display into the entry buffer.

To use the action keys

- 1 If the action key uses the contents of the entry buffer, place the desired information in the entry buffer.
- 2 Position the mouse pointer over the action key and click the action key.

Action keys are user-definable pushbuttons that perform interface or system functions. Action keys can use information from the entry buffer — this makes it possible to create action keys that are more general and flexible.

Several action keys are predefined when you first start the Graphical User Interface. You can use the predefined action keys to make, load, run, and step through the demo program. You'll really appreciate action keys when you define and use your own.

Action keys are defined by setting an X resource. Refer to the chapter "Setting X Resources" for more information about creating action keys.

To use dialog boxes

- 1 Click on an item in the dialog box list to copy the item to the text entry area.
- 2 Edit the item in the text entry area (if desired).
- 3 Click on the "OK" pushbutton to make the selection and close the dialog box, click on the "Apply" pushbutton to make the selection and leave the dialog box open, or click on the "Cancel" pushbutton to cancel the selection and close the dialog box.

The graphical interface uses a number of dialog boxes for selection and recall:

Directory Selection	Selects the working directory. You can change to a previously accessed directory, a predefined directory, or specify a new directory.
---------------------	---

Chapter 3: Entering Commands

Using Menus, the Entry Buffer, and Action Keys

File Selection	From the working directory, you can select an existing file name or specify a new file name.
Entry Buffer Recall	You can recall a previously used entry buffer text string, a predefined entry buffer text string, or a newly entered entry buffer string, to the entry buffer text area.
Command Recall	You can recall a previously executed command, a predefined command, or a newly entered command, to the command line.
Settings Display Modes	You can set the display mode and customize the display presentation for memory and trace list displays.
Modify Register	You can view and modify values of any selected register, as well as recalling previous values of the registers.
Symbol Selection	Selects the current working symbol (cws). You can change to a previously accessed cws, a predefined cws, or specify a new cws.

The dialog boxes share some common properties:

- Most dialog boxes can be left on the screen between uses.
- Dialog boxes can be moved around the screen and do not have to be positioned over the graphical interface window.
- If you iconify the interface window, all dialog boxes are iconified along with the main window.

Except for the File Selection dialog box, predefined entries for each dialog box (and the maximum number of entries) are set via X resources (refer to the "Setting X Resources" chapter).

Examples

To use the File Selection dialog box:

The file filter selects specific files.

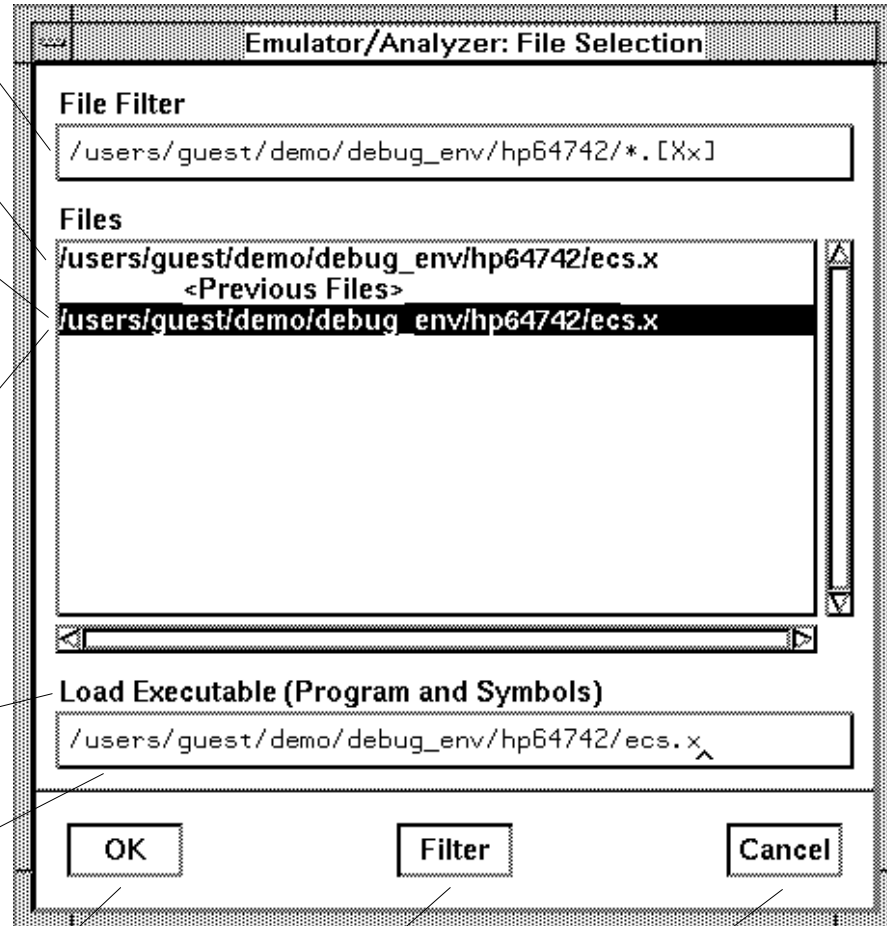
A list of filter-matching files from the current directory.

A list of files previously accessed during the emulation session.

A single click on a file name from either list highlights the file name and copies it to the text area. A double click chooses the file and closes the dialog box.

Label informs you what kind of file selection you are performing.

Text entry area. Text is either copied here from the recall list, or entered directly.



Clicking this button chooses the file name displayed in the text entry area and closes the dialog box.

Entering a new file filter and clicking this button causes a list of files matching the new filter to be read from the directory.

Clicking this button cancels the file selection operation and closes the dialog box.

Chapter 3: Entering Commands
Using Menus, the Entry Buffer, and Action Keys

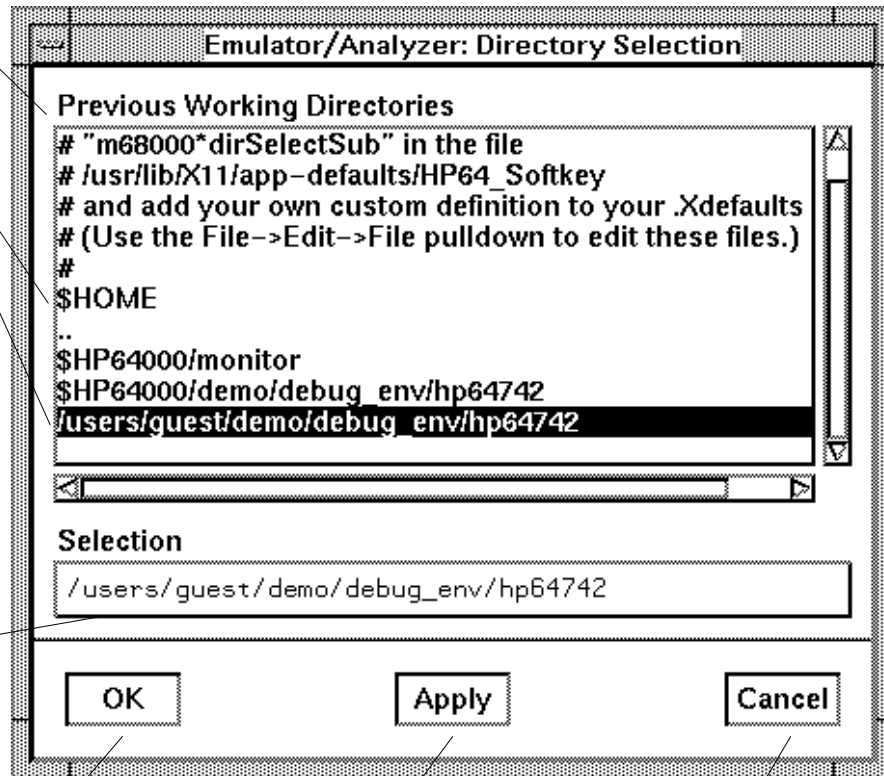
To use the Directory Selection dialog box:

Label informs you of the type of list displayed.

A list of predefined or previously accessed directories.

A single click on a directory name from the list highlights the name and copies it to the text area. A double click chooses the directory and closes the dialog box.

Text entry area. Directory name is either copied here from the recall list, or entered directly.



Clicking this button chooses the directory displayed in the text entry area and closes the dialog box.

Clicking this button chooses the directory displayed in the text entry area, but keeps the dialog box on the screen instead of closing it.

Clicking this button cancels the directory selection operation and closes the dialog box.

To access help information

- 1 Display the Help Index by choosing **Help**→**General Topic...** or **Help**→**Command Line...**
- 2 Choose a topic of interest from the Help Index.

The Help Index lists topics covering operation of the interface as well other information about the interface. When you choose a topic from the Help Index, the interface displays a window containing the help information. You may leave the window on the screen while you continue using the interface.



Using the Command Line

When using the Graphical User Interface, the *command line* portion of the interface gives you the option of entering commands in the same manner as they are entered in the Softkey Interface. Additionally, the graphical interface makes the softkey labels pushbuttons so commands may be entered using the mouse.

This section describes how to:

- Turn the command line off/on.
- Enter commands.
- Edit commands.
- Recall commands.
- Display the help window.

To turn the command line on or off

- To turn the command line on or off using the pulldown menu, choose **Settings**→**Command Line**.
- To turn the command line on or off using the status line popup menu: position the mouse pointer within the status line area, press and hold the *select* mouse button, and choose **Command Line On/Off** from the menu.
- To turn the command line off using the command line entry area popup menu: position the mouse pointer within the entry area, press and hold the *select* mouse button, and choose **Command Line Off** from the menu.

Turns display of the command line area "on" or "off." On means that the command line is displayed and you can use the softkey label pushbuttons, the command return and recall pushbuttons, and the cursor pushbuttons for command line editing. Off means the command line is not displayed and you use only the pulldown menus and the action keys to control the interface.

The command line area begins just below the status line and continues to the bottom of the emulator/analyzer window. The status line is not part of the command line and continues to be displayed whether the command line is on or off.

Choosing certain pulldown menu items while the command line is off causes the command line to be turned on. That is because the menu item chosen requires some input at the command line that cannot be supplied another way.



To enter a command

- 1 To build a command using the softkey pushbuttons, successively position the mouse pointer on a pushbutton and click the *pushbutton select* mouse button until a complete command is formed.

To build a command using the keyboard function keys, successively press function keys corresponding to softkey buttons until a complete command is formed.

To build a command using direct keyboard entry, type the command directly into the command line entry area until a complete command is formed.

Or:

Type the first few characters of a command token and then press the <Tab> key to cause the interface to complete the command token. Continue to use *token completion* until a complete command is formed.

Chapter 3: Entering Commands

Using the Command Line

- 2 To execute a completed command using the Command Line pushbutton, click the pushbutton labeled **Return** (found near the bottom of the command line in the "Command" group).

To execute a completed command using the keyboard, press the <RETURN> key on the keyboard.

To execute a completed command using the Command Line entry area popup menu: position the mouse pointer in the command line entry area; press and hold the *select* mouse button until the Command Line popup menu appears; then, choose the **Execute Command** menu item.

You can combine pushbutton, function key, and keyboard entry to form a complete command.

A complete command is a string of softkey labels and text entered with the keyboard. You know a command is complete when **Return** pushbutton is not grayed-out. The interface does not check or act on a command, however, until the command is executed. (In contrast, commands resulting from pulldown menu choices and action keys are supplied with the needed carriage return as part of the command.)

To edit the command line using the command line pushbuttons

- To position the cursor at a specific character, place the mouse pointer on the character and click the *select* mouse button.
- To clear the command line, click the **Clear** pushbutton.
- To clear the command line from the cursor position to the end of the line, click the **Clear to end** pushbutton.
- To move to the right one command word or token, click the **Forward** pushbutton.
- To move to the left one command word or token, click the **Backup** pushbutton.
- To insert characters at the cursor position, press the **insert key** to change to insertion mode, and then type the characters to be inserted.
- To replace characters at the cursor position, press the **insert key** to change to replacement mode, and then type the replacement characters.
- To delete characters to the left of the cursor position, press the <BACKSPACE> key.

When the cursor arrives at the beginning of a command word or token, the softkey labels change to display the possible choices at that level of the command.

When moving by words left or right, the **Forward** pushbutton becomes grayed-out and unresponsive when the cursor reaches the end of the command string. Similarly, the **Backup** pushbutton becomes grayed-out and unresponsive when the cursor reaches the beginning of the command.



To edit the command line using the command line popup menu

- To position the cursor at a specific character, place the mouse pointer on the character and click the *select* mouse button.
- To clear the command line: position the mouse pointer within the Command Line entry area; press and hold the *select* mouse button until the Command Line popup menu appears; choose **Clear Entire Line** from the menu.
- To clear the command line from the cursor position to the end of the line: position the mouse pointer at the place where you want the clear-to-end to start; press and hold the *select* mouse button until the Command Line popup menu appears; choose **Clear to End of Line** from the menu.
- To insert characters: position the mouse pointer where you wish to locate the text cursor (or over a non-text area to use the current text cursor location); press and hold the *select* mouse button to display the Command Line popup menu; choose **Position Cursor, Insert Mode** from the menu; type the characters to be inserted.
- To replace characters: position the mouse pointer where you wish to locate the text cursor (or over a non-text area to use the current text cursor location); press and hold the *select* mouse button to display the Command Line popup menu; choose **Position Cursor, Replace Mode** from the menu; type the characters to be inserted.

When the cursor arrives at the beginning of a command word or token, the softkey labels change to display the possible choices at that level of the command.

To edit the command line using the keyboard

- With the mouse pointer in the display area, on the status line, or in the command line area, use the <Left arrow>, <Right arrow>, <Tab>, <Shift><Tab>, <Insert char>, <Back space>, <Delete char>, <Clear line>, and <CTRL>u keys.

The <Left arrow> and <Right arrow> keys move the cursor single spaces to the left or right.

The <Tab> and <Shift><Tab> keys move the cursor to the next or previous word on the command line.

The <Insert char> key enters the insert editing mode and allows characters or command options to be inserted at the cursor location.

The <Back space> key deletes the character to the left of the cursor.

The <Delete char> key deletes the character to the right of the cursor.

The <Clear line> key deletes the characters from the cursor to the end of the line.

The <CTRL>u key erases the command line.



To recall commands

- 1 Click the pushbutton labeled **Recall** in the Command Line to display the dialog box.
- 2 Choose a command from the buffer list. (You can also enter a command directly into the text entry area of the dialog box.)

Because all command entry methods in the interface — pulldown menus, action keys, and command line entries — are echoed to the command line entry area, the contents of the Command Recall dialog box is not restricted to just commands entered directly into the command line entry area.

The Command Recall dialog box contains a list of interface commands executed during the session as well as any predefined commands present at interface startup.

Chapter 3: Entering Commands

Using the Command Line

If you exit the emulation/analysis session with the interface "locked", commands in the recall buffer are saved and will be present when you restart the interface.

You can predefine entries for the Command Recall dialog box and define the maximum number of entries by setting X resources (refer to the "Setting X Resources" chapter).

See "To use dialog boxes" for information about using dialog boxes.

You can also recall the most recent 20 commands by pressing <CTRL>r while the mouse pointer is in the display area, on the status line, or in the command line area. Pressing <CTRL>b cycles forward through the recall buffer.

To get help about the command line

- To display the help topic explaining the operation of the command line, press the **Help** pushbutton located near the bottom-right corner of the Command Line area.

Using Command Files

You can execute a series of commands that have been stored in a command file. You can create command files by logging commands while using the interface or by using an editor on your host computer.

Once you create a command file, you can execute the file in the emulation environment by typing the name of the file on the command line and pressing <RETURN>.

Command files execute until an end-of-file is found or until a syntax error occurs. You can stop a command file by pressing <CTRL>c or the <Break> key.

This section shows you how to:

- Start logging commands to a command file.
- Stop logging commands to a command file.
- Playback (execute) a command file.

Nesting Command Files

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

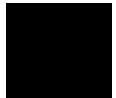
Comments in Command Files

Text that follows a pound sign (#), up to the end of the line, is interpreted as a comment.

Using the wait Command

When editing command files, you can insert **wait** commands to pause execution of the command file at certain points.

If you press <CTRL>c to stop execution of a command file while the "wait" command is being executed from the command file, the <CTRL>c will terminate the "wait" command, but will not terminate command file execution. To do this, press <CTRL>c again.



Chapter 3: Entering Commands

Using Command Files

Use the **wait measurement_complete** command after changing the trace depth. By doing this, when you copy or display the trace after changing the trace depth, the new trace states will be available. Otherwise the new states won't be available.

Passing Parameters

Command files provide a convenient method for passing parameters by using a parameter declaration line preceding the commands in the command file. When the command file is called, the system will prompt you for current values of the formal parameters listed.

Parameters are defined as:

Passed Parameters - These are ASCII strings passed to a command file. Any continuous set of ASCII characters can be passed. Spaces separate the parameters.

Formal Parameters - These are symbols preceded by an ampersand (&), which are the variables of the command file.

The ASCII string passed (passed parameter) will be substituted for the formal parameter when the command file is executed.

The only way to pass a parameter containing a space is to enclose the parameter in double quotes (") or single quotes ('). Thus, to pass the parameter HP 9000 to a command file, you can use either "HP 9000" or 'HP 9000'.

The special parameter **&ArG_IeFT** gets set to all the remaining parameters specified when the command file was invoked. This lets you use variable size parameter lists. If no parameters are left, **&ArG_IeFT** gets set to NULL.

Consider the command file example (named CMDFILE) shown below:

```
PARMS &ADDR &VALUE1
#
# modify a location or list of locations in memory
# and display the result
#
modify memory &ADDR words to &VALUE1 &ArG_IeFT
display memory &ADDR blocked words
```

When you execute CMDFILE, you will be prompted with:

Define command file parameter [&ADDR]

To pass the parameter, enter the address of the first memory location to be modified. You will then be prompted for **&VALUE1**. If you enter, for example, "0,-1,20, 0ffff, 4+5*4", the first parameter "0,-1,20," is passed to **&VALUE1** and the remaining parameters "0ffff," and "4+5*4" are passed to **&ArG_IeFT**.

You can also pass the parameters when you invoke the command file (for example, CMDFILE 1000h 0,-1,20, 0ffff, 4+5*4).

Other Things to Know About Command Files

You should know the following about using command files:

- 1 Command files may contain shell variables. Only those shell variables beginning with "\$" followed by an identifier will be supported. An identifier is a sequence of letters, digits or underscores beginning with a letter or underscore. The identifier may be enclosed by braces "{ }" or entered directly following the "\$" symbol. Braces are required when the identifier is followed by a letter, a digit or an underscore that is not interpreted as part of its name.

For example, assume a directory named /users/softkeys and the shell variable "S". The value of "S" is "soft". By specifying the directory as /users/\${S}keys the correct result is obtained. However, if you attempt to specify the directory as /users/\$Skeys, the Softkey Interface looks for the value of the variable "Skeys". This is not the operators intended result. You may not get the intended result unless Skeys is already defined to be "softkeys".

You can examine the current values of all shell variables defined in your environment with the command "env".

- 2 Positional shell variables, such as \$1, \$2, and so on, are not supported. Neither are special shell variables, such as \$@, \$*, and so on, supported.
- 3 You can continue command file lines. This is done by avoiding the line feed with a backslash (\). A line terminated by "\" is concatenated with any following lines until a line that does not contain a backslash is found. A line constructed in this manner is recognized and executed as one single command line. If the last line in a command file is terminated by "\", it appears on the command line but is not executed. Normally, the line feed is recognized as the command terminator. The UNIX environment recognizes three quoting

Chapter 3: Entering Commands

Using Command Files

characters for shell commands which are double quotes ("), single quotes ('), and the backslash symbol (\).

For example, the following three lines are treated as a single shell command. The two hidden line feeds are ignored because they are inside the two single quotes ('):

```
!awk '/$/ { blanks++ }  
END { print blanks }  
' an_unix_file
```

To start logging commands to a command file

- Choose **File**→**Log**→**Record** and use the dialog box to select a command file name.
- Using the command line, enter the **log_commands to <file>** command.

To stop logging commands to a command file

- Choose **File**→**Log**→**Stop**.
- Using the command line, enter the **log_commands off** command.

To playback (execute) a command file

- Choose **File**→**Log**→**Playback** and use the dialog box to select the name of the command file you wish to execute.
- Using the command line, enter the name of the command file and press <RETURN>.

If you enter the name of the command file in the command line and the interface cannot find the command file in the current directory, it searches the directories specified in the HP64KPATH environment variable.

To interrupt playback of a command file, press the <CTRL>c key combination. (The mouse pointer must be within the interface window.)

If you press <CTRL>c to stop execution of a command file while the "wait" command is being executed from the command file, the <CTRL>c will terminate the "wait" command, but will not terminate command file execution. To do this, press <CTRL>c again.



Using Pod Commands

Pod commands are Terminal Interface commands. The Terminal Interface is the low-level interface that resides in the firmware of the emulator.

A pod command used in the Graphical User Interface bypasses the interface and goes directly to the emulator. Because some pod commands can cause the Graphical User Interface to become out-of-sync with the emulator, or even cause the interface to terminate abnormally, they must be used with care.

For example, if you change configuration items, the actual state of the emulator will no longer match the internal record the interface keeps about the state of the emulator.

Issuing certain communications-related commands can prevent the interface from communicating with the emulator and cause abnormal termination of the interface.

However, it is sometimes necessary to use pod commands. For example, you must use a pod command to execute the emulator's *performance verification (pv)* routine. Performance verification is an internal self-test procedure for the emulator.

Remember that pod commands can cause trouble for the high-level interface if they are used indiscriminately.

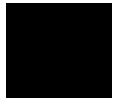
This section shows you how to:

- Display the pod commands screen.
- Use pod commands.
- Copy the pod commands screen to a file.

To display the pod commands screen

- Choose **Display**→**Pod Commands**.

The pod commands screen displays the results of pod (Terminal Interface) commands. To set the interface to use pod commands, choose **Settings**→**Pod Command Keyboard**.



To use pod commands

- To begin using pod commands, choose **Settings**→**Pod Command Keyboard**.
- To end using pod commands, click the **suspend** pushbutton softkey.

The **Settings**→**Pod Command Keyboard** command displays the pod commands screen and activates the keyboard for entering pod command on the command line.

To copy the pod commands screen to a file

- To append the contents of the Terminal Interface screen to the contents of a file, select **File**→**Copy**→**Pod Commands**

Forwarding Commands to Other HP 64700 Interfaces

To allow the emulator/analyzer interface to run concurrently with other HP 64700 interfaces like the high-level debugger and software performance analyzer, a background "daemon" process is necessary to coordinate actions in the interfaces.

This background process also allows commands to be forwarded from one interface to another. Commands are forwarded using the **forward** command available in the command line. The general syntax is:

```
forward <interface_name> "<command_string>" <RETURN>
```

This section shows you how to:

- Forward commands to the high-level debugger.
- Forward commands to the software performance analyzer.

To forward commands to the high-level debugger

- Enter the **forward debug "<command string>"** command using the command line.

Examples

To send the "Program Run" command to the debugger:

```
forward debug "Program Run" <RETURN>
```

Or, since only the capitalized key is required:

```
forward debug "P R" <RETURN>
```

To forward commands to the software performance analyzer

- Enter the **forward perf "<command string>"** command using the command line.

Examples

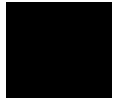
To send the "profile" command to the software performance analyzer:

forward perf "profile" <RETURN>





4



Configuring the Emulator/Analyzer

Configuring the Emulator

This chapter describes how to configure an emulator. When you plug an emulator into a target system, you must configure the emulator so that it operates correctly in the target system.

Two types of configuration interfaces appear in Graphical User Interfaces. The type of configuration interface in your emulator will depend on the emulator you are using.

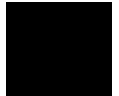
- Fully graphical. It consists of a top-level dialog box that you can use to access individual dialog boxes and displays for each of the configuration sections.
- Softkey based. It consists of a top-level dialog box that you can use to access a series of questions that set up the associated emulation configuration section.

This chapter describes the two types of emulator configurations in general terms. For information about your emulator's specific configuration options, refer to your emulator *User's Guide*.

Using the Fully Graphical Configuration Interface

The configuration tasks of the fully graphical interface are described in this section of the manual. It shows you how to modify, store, and load configurations. It also shows you how to use the memory mapping display. Individual descriptions show you how to:

- Start the configuration interface.
- Modify a configuration section.
- Apply configuration changes to the emulator.
- Change the configuration directory context.
- Display the configuration context.
- Access help topics.
- Access help for an individual configuration item.
- Store configuration changes to a file.
- Load a saved configuration file.
- Exit the configuration interface.



To start the fully graphical configuration interface

- Choose **Modify**→**Emulator Config...** from the emulator/analyzer interface pulldown menu.

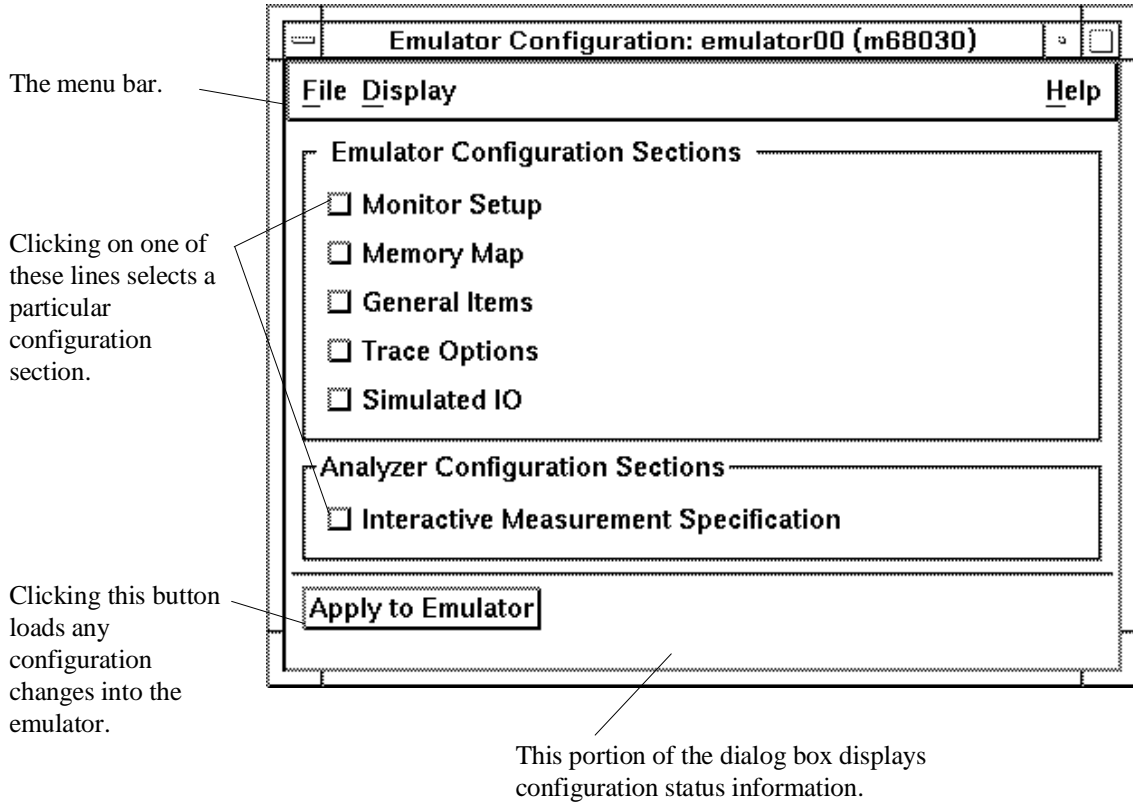
The configuration interface top-level dialog box is displayed.

The top-level dialog box on the following page is used in the HP 64747 emulator. It may appear slightly different from the configuration top-level dialog box of your emulator, but the way the dialog box accesses configuration sections will be the same.

The configuration interface may be left running while you are using the emulator/analyzer interface.

Examples

An emulator configuration interface top-level dialog box is shown below.



To modify a configuration section

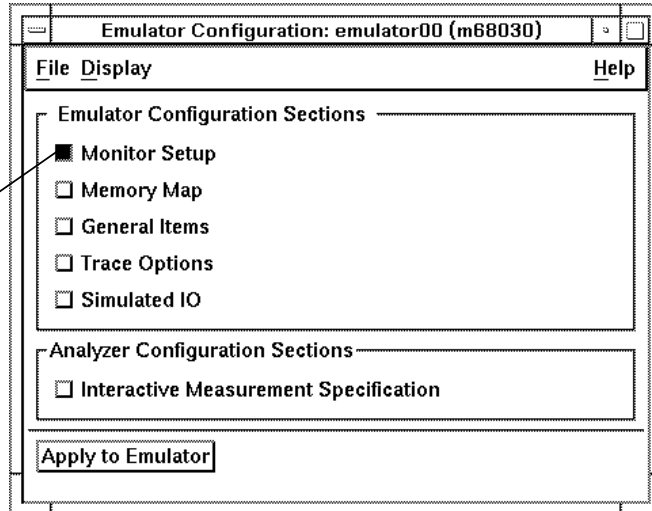
- 1 Start the emulator configuration interface.
- 2 Click on a section name in the configuration interface top-level dialog box.
- 3 Use the section dialog box to make changes to the configuration.

As soon as you change a configuration option, the change is recorded, but not loaded (as seen by a "Changes Not Loaded" message that appears at the bottom of the top-level dialog box).

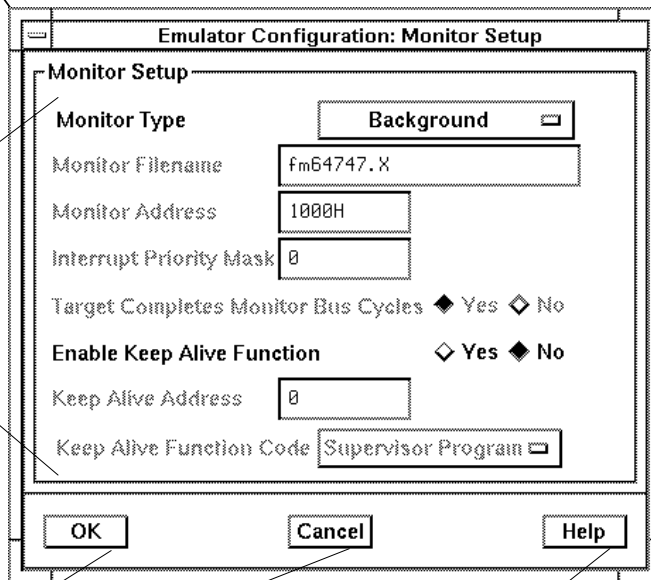
Examples

Most configuration sections provide dialog boxes similar to the following.

The dialog for this section has been opened



Configuration options in this section



Closes the dialog box

Cancels all changes since last "OK", "Apply to Emulator", or store to file.

Presents emulator configuration help topic browser.

To apply configuration changes to the emulator

- Click the "Apply to Emulator" button in the top-level dialog box.

This loads the configuration changes into the emulator. Status text to the right shows whether the load was successful.

You can apply configuration changes to the emulator at any time (even while section dialog boxes are open). This lets you verify changes without closing section dialog boxes.

The "Apply to Emulator" button does not store configuration changes to a file.

To store configuration changes to a file

- Choose **File**→**Store...** from the pulldown menu in the top-level configuration interface window, and use the file selection dialog box to name the configuration file.

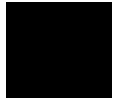
When modifying a configuration, you can store your answers at any time.

Configuration information is saved in a file with the extension ".EA".

CAUTION

Do not modify configurations by editing ".EA" files. Use the configuration interface to modify and save configurations.

For more information on how to use dialog boxes, refer to Chapter 3, "Entering Commands".



To change the configuration directory context

- Choose **File**→**Directory...** from the pulldown menu in the top-level configuration interface window, and use the directory selection dialog box to specify the new directory.

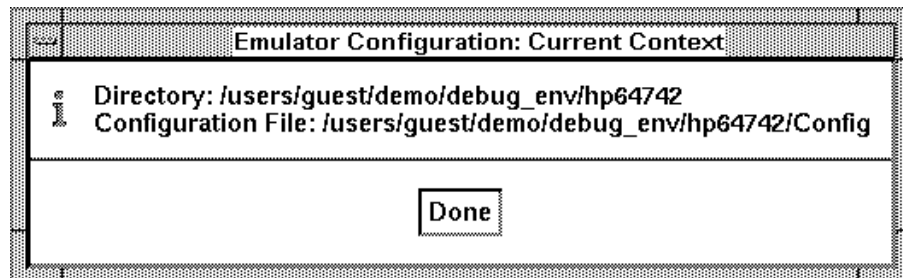
The directory context specifies the directory to which configuration files are stored and from which they are loaded.

For more information on how to use dialog boxes, refer to Chapter 3, "Entering Commands".

To display the configuration context

- Choose **Display**→**Context...** from the pulldown menu in the top-level configuration interface window.

The current directory context and the current configuration files are displayed in a window. Click the "Done" pushbutton when you wish to close the window.



To access help topics

- Choose **Help**→**General Topic...** from the pulldown menu in the configuration interface top-level dialog box, click on a topic in the selection dialog box, and click the "OK" button.

To access help for an individual configuration item

- Place the mouse pointer on the line of interest and press the f1 keyboard key.
- Choose **Help**→**On Item...** in the top-level dialog box or the memory map window. The mouse pointer changes from an arrow to a question mark. Place the question mark over a selection button or in the entry field on the line of interest, and click the *pushbutton select* mouse button.

If you are using the **Help**→**On Item...** pulldown menu and its associated question mark, you may find that in some dialog boxes, the question mark may not obtain a help screen when you place it on a command name, but the help screen may be obtained when you place the question mark over an input field or button associated with the command name. This is due to a known defect in Motif 1.1.

The configuration interface provides individual help for each item in the top level dialog box and throughout the configuration section dialog boxes, except the Memory Map display; there is no **Help**→**On Item...** in the Memory Map display.

To exit the configuration interface

- Choose **File**→**Exit...** from the pulldown menu in the top-level dialog box (or type <CTRL>x).

If configuration changes have not been stored to a file, a confirmation dialog box appears, giving you the options of storing, exiting without storing, or canceling the exit.

To load an existing configuration file

- In the emulator/analyzer interface, choose **File**→**Load**→**Emulator Config...** from the pulldown menu, and use the file selection dialog box to specify the configuration file to be loaded.

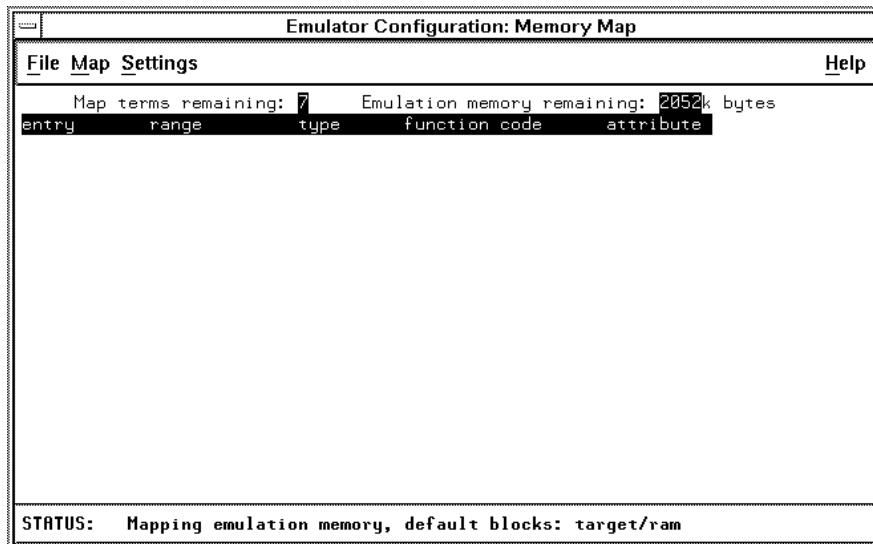
This command loads previously created and stored configuration files. You cannot load a configuration while the configuration interface is running.

Mapping memory in the fully graphical configuration interface

This section shows how to use the memory map popup menus and dialog box. It discusses the details of how to map memory for an emulator, its emulation monitor, and the target system. The Memory Map shown on this page is for an emulator using its background monitor. If you decide to use a foreground monitor, the first entry you will see in the Memory Map will be the address range reserved for the monitor.

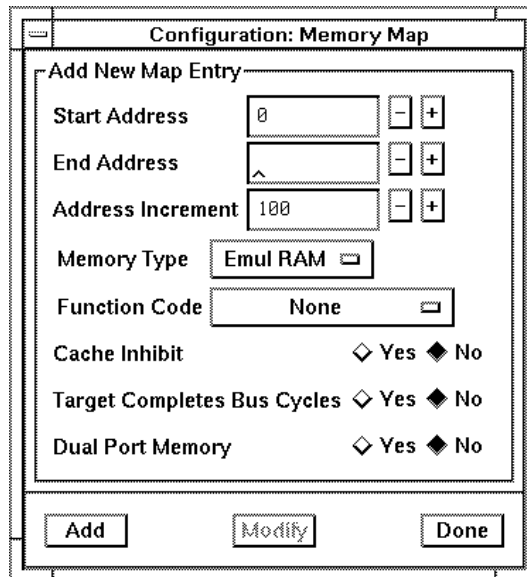
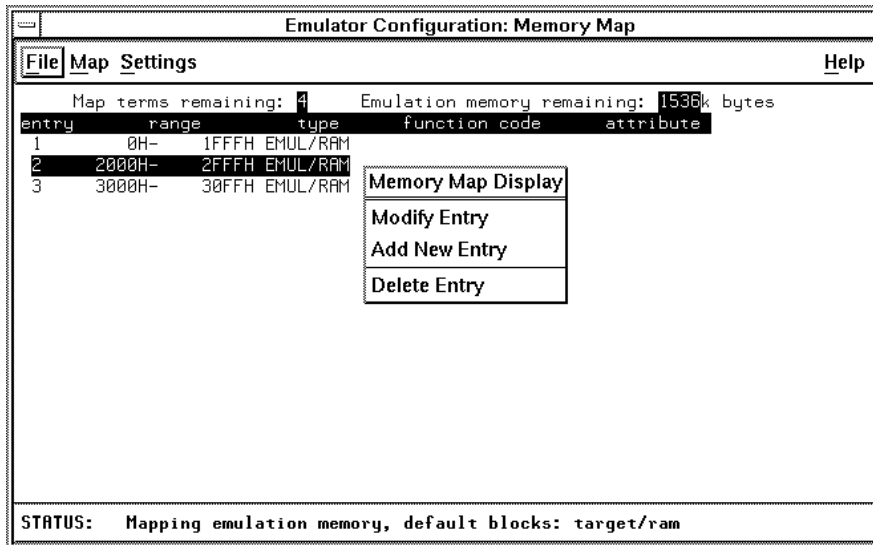
By default, the emulator assumes all memory addresses are in RAM space in your target system. If you wish to load some of your target program in emulation memory, or identify some of your memory addresses as ROM or Guarded, enter those specifications in the memory map.

- To create a new map entry or modify an existing entry, choose **Map**→**Add New Entry**, or **Map**→**Modify Entry**→<number>. The appropriate Memory Map dialog box will open (see next page).



Chapter 4: Configuring the Emulator/Analyzer
Using the Fully Graphical Configuration Interface

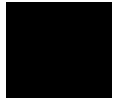
- To create a new map entry or modify an existing entry from the Memory Map popup menu, press and hold the *select* mouse button and choose **Add New Entry**, or **Modify Entry**. The appropriate Memory Map dialog box (below) will open.



Using the Softkey-Based Configuration Interface

This section shows you how to use the softkey-based configuration interface. It shows you how to modify, store, and load configurations using the top-level dialog box and each associated series of questions. Individual descriptions show you how to:

- Start the configuration interface.
- Modify a configuration section.
- Change the configuration directory context.
- Display the configuration context.
- Access help information.
- Store a configuration.
- Load a configuration.
- Exit the configuration interface.



To start the configuration interface

- Choose **Modify**→**Emulator Config...** from the emulator/analyzer interface pulldown menu.

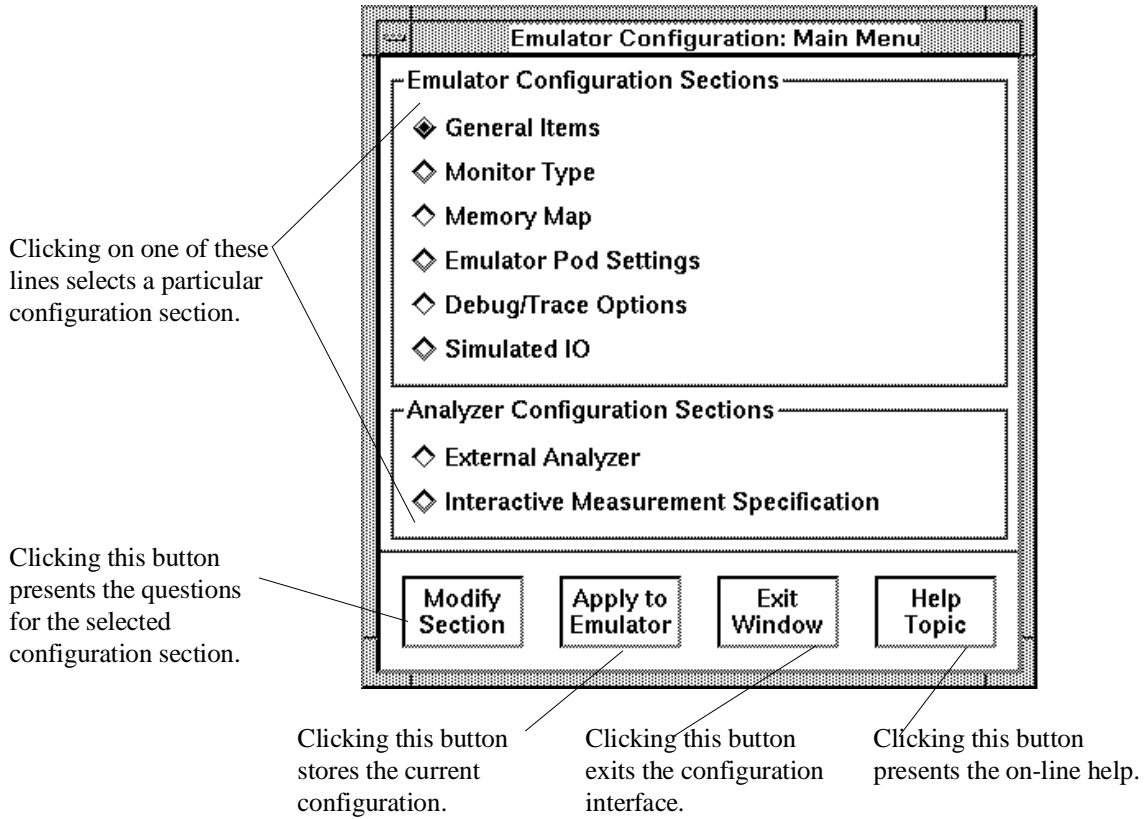
The configuration interface main menu (see example on next page) is displayed.

The configuration sections that are presented depend on the hardware and features of your particular emulator.

The configuration interface may be left running while you are using the emulator/analyzer interface.

Examples

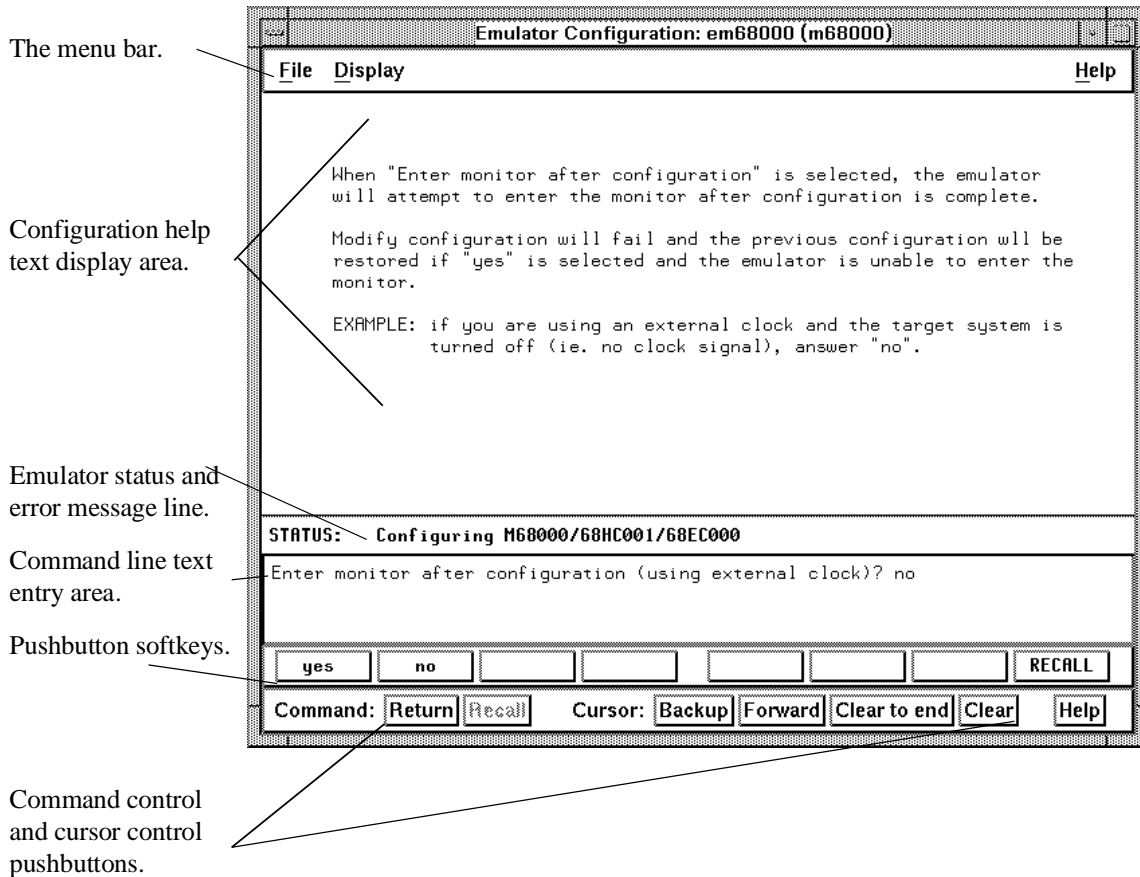
The 68000 emulator configuration interface main menu is shown below.



To modify a configuration section

- 1 Start the emulator configuration interface.
- 2 Click on a section name in the configuration interface main menu, and click the "Modify Section" pushbutton.
- 3 Use the command line to answer the configuration questions.

Each configuration section presents a window similar to the following.



Chapter 4: Configuring the Emulator/Analyzer Using the Softkey-Based Configuration Interface

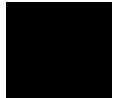
To answer a configuration question, click the softkey pushbutton that has your answer. Or, click on the "Return" command pushbutton to accept the answer that is shown.

When you answer a configuration question, you are normally presented with the next question in the section; however, there are some cases when a carriage return is required, and you can supply it by clicking the "Return" command pushbutton or by pressing the RETURN key on your keyboard.

At the last question of a configuration section, you are asked if you wish to return to the main menu. You can click the "next_sec" softkey pushbutton to access the questions in the next configuration section.

To recall a configuration question, click the "RECALL" softkey pushbutton. If you do this at the starting question of a configuration section, you are asked if you want to return to the main menu.

In order for the emulator to recognize any configuration changes, the configuration must be applied to the emulator.



To store a configuration

- When answering the configuration questions, choose **File**→**Store...** from the pulldown menu, and use the File Selection dialog box to name the configuration file.
- From the configuration interface main menu, click on the "Apply to Emulator" button, and use the File Selection dialog box to name the configuration file.

The file to which the configuration is stored becomes the current configuration file. The emulator only recognizes configuration changes when they are stored or loaded.

When modifying a configuration, you can choose to store your answers at any time. This is useful for quickly verifying the effect a configuration change has on the emulator.

For more information on how to use dialog boxes, refer to Chapter 3, "Entering Commands."

To change the configuration directory context

- When answering the configuration questions, choose **File**→**Directory...** from the pulldown menu, and use the Directory Selection dialog box to specify the new directory.

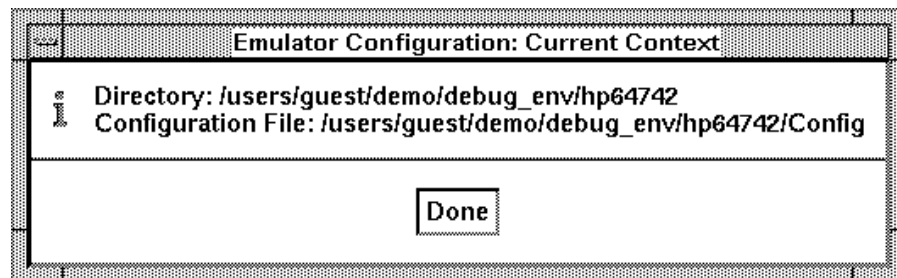
The directory context specifies the directory to which configuration files are stored and from which they are loaded.

For more information on how to use dialog boxes, refer to Chapter 3, "Entering Commands."

To display the configuration context

- When answering the configuration questions, choose **Display**→**Context...** from the pulldown menu.

The current directory context and the current configuration files are displayed in a window. Click the "Done" pushbutton when you wish to close the window.



To access help information

- When answering the configuration questions, choose **Help**→**General Topic...** from the pulldown menu.
- From the configuration interface main menu, click on the "Help Topic" button.

To exit the configuration interface

- When answering the configuration questions, choose **File**→**Exit...** from the pulldown menu (or type <CTRL>x), and click "Yes" in the confirmation dialog box.
- From the configuration interface main menu, click the "Exit Window" button, and click "Yes" in the confirmation dialog box.

The confirmation dialog box only appears if changes have been made to the current configuration.

When you choose "Yes" from the confirmation dialog box, any modifications made to the configuration which haven't been stored are lost. Choosing "No" from the confirmation dialog box cancels the exit and keeps the emulator configuration interface running.

To load a configuration

- In the emulator/analyzer interface, choose **File**→**Load**→**Emulator Config...** from the pulldown menu, and use the File Selection dialog box to specify the configuration file to be loaded.

You can load previously created and stored configuration files.

5



Using the Emulator/Analyzer

Using the Emulator/Analyzer

These tasks are grouped into the following sections:

- Loading programs and symbols.
- Using symbols.
- Using context commands.
- Displaying and modifying memory.
- Displaying data values.
- Displaying and modifying registers.
- Executing programs.
- Using breakpoints.
- Tracing program execution.
- Changing the interface settings.
- Using system commands.
- Using simulated I/O.
- Using basis branch analysis.

Loading Programs and Symbols

When you compile (or assemble) and link programs, an absolute file is created. You can load absolute files into emulation or target system memory.

Two types of absolute files can be loaded: IEEE-695 format files, and HP format absolute files. Both types of absolute files can contain symbolic information.

You can choose to load programs without loading the associated symbolic information.

Also, you can load only the symbolic information from a program's absolute file. This is for situations where the program already exists in target system ROM.

This section shows you how to:

- Load programs.
- Load programs without symbols.
- Load only symbols.

To load programs

- Choose **File**→**Load**→**Executable ...** and use the dialog box to select the absolute file.

This command loads the executable part of your absolute file into emulation memory or target RAM. Any symbolic information found in the absolute file is also loaded.

When symbolic information is loaded, you can display global and local symbols. Also, symbolic information can appear in the memory, breakpoint, data, register, and trace displays.

To load programs without symbols

- Choose **File**→**Load**→**Program Only ...** and use the dialog box to select the absolute file.

This command loads the executable part of your absolute file into emulation memory and target RAM but does not load symbolic information found in the absolute file.

Symbolic information currently loaded into the interface remains and is unaffected by this command.

To load only symbols

- Choose **File**→**Load**→**Symbols Only ...** and use the dialog box to select the absolute file.

This command causes only the symbolic information found in the absolute file to be loaded. The executable part of the absolute file is not loaded.

This option is particularly useful for loading symbols for files located in target ROM so that you can view symbols with that code.

Using Symbols

The Graphical User Interface gives you an interactive ability to follow the symbol scope from the highest to the lowest level and back up again.

If symbol information is present in the absolute file, it is loaded along with the absolute file unless you use load the program without symbols. Both global symbols and symbols that are local to a program module can be displayed.

Long symbol names can be truncated in the symbols display; however, you can increase the width of the symbols display by starting the interface with more columns (refer to the "Setting X Resources" chapter).

This section shows you how to:

- Display global symbols.
- Display local symbols.
- Display a symbol's parent symbol.
- Copy-and-paste a full symbol name to the entry buffer.

To display global symbols

- Choose **Display**→**Global Symbols**.

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

If there is more than a screen full of information, you can use the up arrow, down arrow, <NEXT>, or <PREV> keys to scroll the information up or down on the display.

To display local symbols

- When displaying symbols, position the mouse pointer over a symbol on the symbol display screen and click the *select* mouse button.
- When displaying symbols, position the mouse pointer over the symbol, press and hold the *select* mouse button, and choose **Display Local Symbols** from the popup menu.
- Position the mouse cursor in the entry buffer and enter the module whose local symbols are to be displayed; then, choose **Display→Local Symbols ()**.

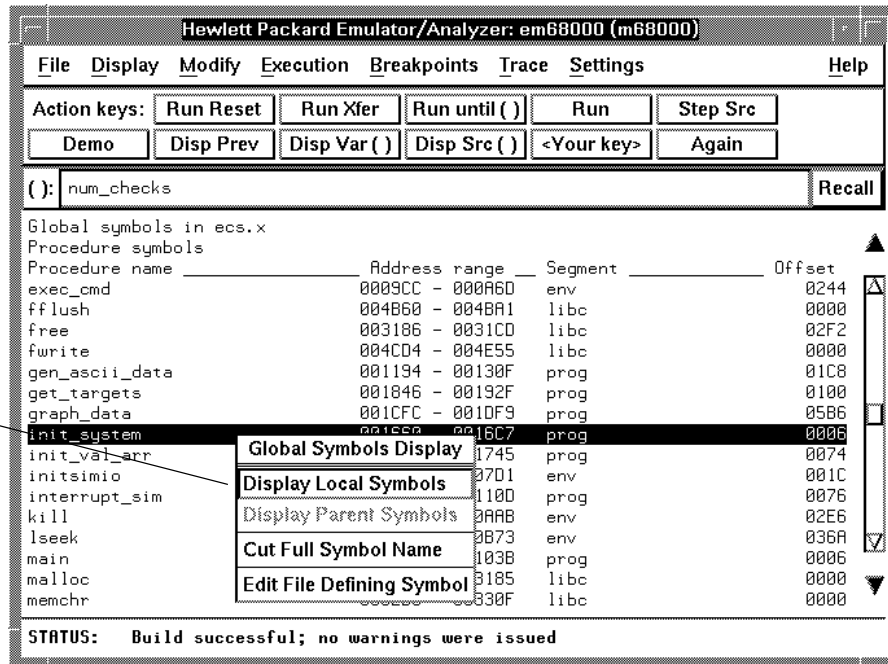
The **Display→Local Symbols ()** command displays the local symbols associated with the symbol representing the context. If no local symbols are associated, the interface displays the parent symbol.

To display the address ranges associated with the high-level program's source file line numbers, you must display the local symbols in the file.

Examples

To display local symbols using the symbols display popup menu:

View the local symbols associated with the highlighted symbol by choosing this menu item.



If local symbols exist within the scope of the symbol chosen, then the display changes to show those symbols. Otherwise, the interface issues an error.

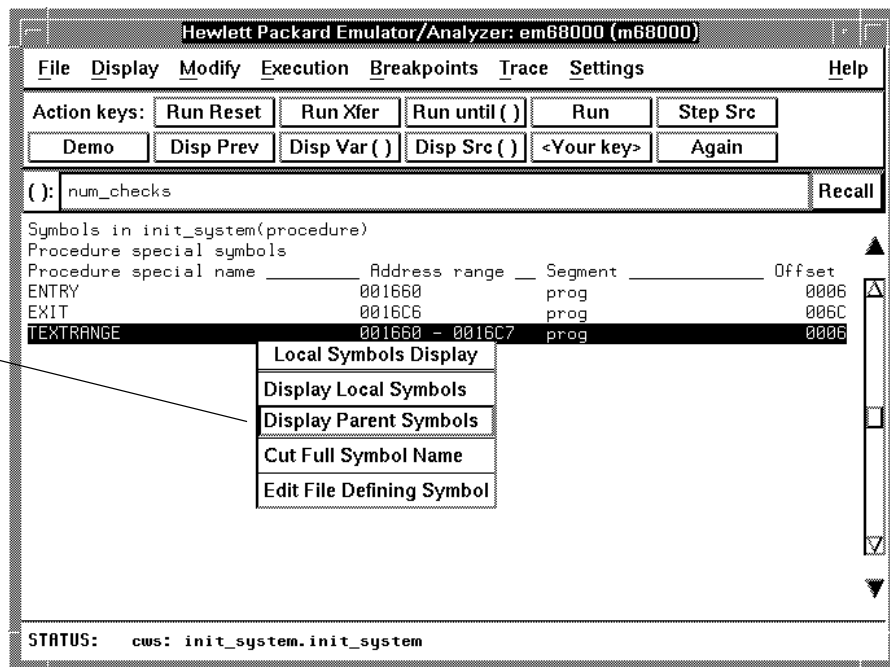
To display a symbol's parent symbol

- When displaying symbols, position the mouse pointer over the symbol, press and hold the *select* mouse button, and choose **Display Parent Symbols** from the popup menu.

If a parent symbol does not exist for the highlighted symbol, this menu item will be grayed-out and unresponsive to mouse clicks.

Examples

View the parent symbol associated with the highlighted symbol by choosing this menu item.



To copy-and-paste a full symbol name to the entry buffer

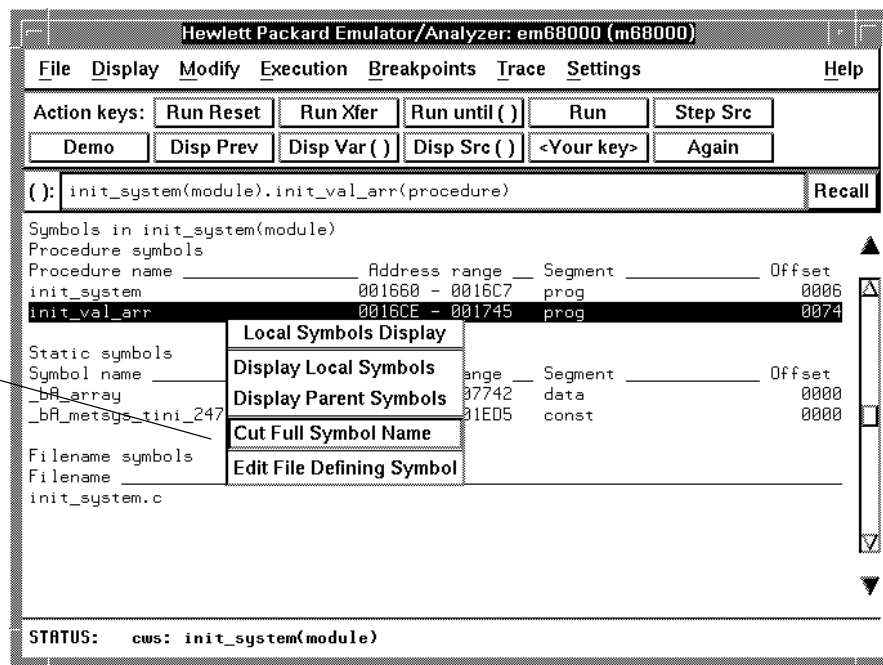
- When displaying symbols, position the mouse pointer over the symbol, press and hold the *select* mouse button, and choose **Cut Full Symbol Name** from the popup menu.

Once the full symbol name is in the entry buffer, you can use it with pulldown menu items or paste it to the command line area.

By cutting the full symbol name, you get the complete names of symbols that have been truncated. Also, you are guaranteed of specifying the proper scope of the symbol.

Examples

Copy the full name of the highlighted symbol to the entry buffer by choosing this menu item.



Using Context Commands

The commands in this section display and control the directory and symbol contexts for the interface.

Directory context. The current directory context is the directory accessed by all system references for files—primarily load, store, and copy commands—if no explicit directory is mentioned. Unless you have changed directories since beginning the emulation session, the current directory context is that of the directory from which you started the interface.

Symbol context. The emulator/analyzer interface and the Symbol Retrieval Utilities (SRU) together support a current working symbol context. The current working symbol represents an enclosing scope for local symbols. If symbols have not been loaded into the interface, you cannot display or change the symbol context.

This section shows you how to:

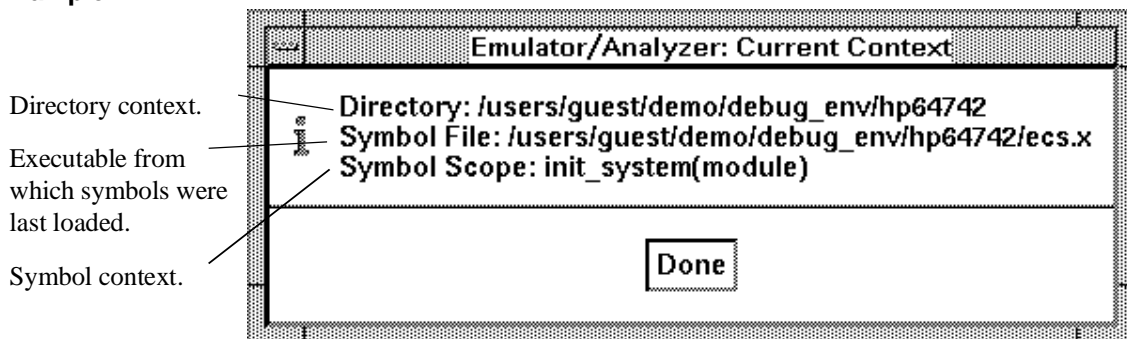
- Display the current directory and symbol context.
- Change the directory context.
- Change the symbol context.

To display the current directory and symbol context

- Choose **Display**→**Context**.

The current directory and working symbol contexts are displayed, and also the name of the last executable file from which symbols were loaded.

Example



To change the directory context

- Choose **File**→**Context**→**Directory** and use the dialog box to select a new directory.

The Directory Selection dialog box contains a list of directories accessed during the emulation session as well as any predefined directories present at interface startup.

You can predefine directories and set the maximum number of entries for the Directory Selection dialog box by setting X resources (see the "Setting X Resources" chapter).

You can also change the current directory context by executing a "cd" command from the command line.

To change the current working symbol context

- Choose **File**→**Context**→**Symbols** and use the dialog box to select the new working symbol context.

You can predefine symbol contexts and set the maximum number of entries for the Symbol Scope Selection dialog box by setting X resources (see the "Setting X Resources" chapter).

Displaying local symbols or displaying memory in mnemonic format causes the working symbol context to change as well. The new context will be that of the local symbols or memory locations displayed.

You can also change the current working symbol context by typing the **cws** <**symbol_context**> command on the command line. (Because **cws** is a hidden command and doesn't appear on a softkey label, you have to type it in.)

Displaying and Modifying Memory

You can display and modify the contents of memory in hexadecimal formats and in real number formats. You can also display the contents of memory in assembly language mnemonic format.

This section shows you how to:

- Display memory.
- Mixing memory displays with associated source lines.
- Display memory in mnemonic format.
- Display memory in mnemonic format at the current PC.
- Return to the previous mnemonic display.
- Display memory in hexadecimal format.
- Display memory in real number format.
- Display memory at an address.
- Display memory repetitively.
- Modify memory.
- Modify memory at an address.



To display memory

- Choose **Display**→**Memory**.

This command either re-displays memory in the format specified by the last memory display command, or, if no previous command has been executed, displays memory as hexadecimal bytes beginning at address zero.

To display memory in mnemonic format

- To display memory at a particular address, place an absolute or symbolic address in the entry buffer; then, choose **Display→Memory→Mnemonic ()**.
- To display memory at the current program counter address, choose **Display→Memory→Mnemonic at PC**.

A highlighted bar shows the location of the current program counter address. This allows you to view the program counter while stepping through user program execution.

Whether source lines, assembly language instructions, or symbols are included in the display depends on the modes you choose with the **Settings→Source/Symbols Modes** or **Settings→Display Modes** pulldown menu items. See the "Changing the Interface Settings" section.

If symbols are loaded into the interface, the default is to display source only.

To return to the previous mnemonic display

- Choose **Display→Memory→Mnemonic Previous**.

This command is useful for quickly returning to the previous mnemonic memory display.

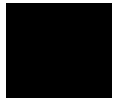
For example, suppose you are stepping source lines and you step into a function that you would like to step over. You can return to the previous mnemonic memory display, set a breakpoint at the line following the function call, and run the program from the current program counter.

To display memory in hexadecimal format

- Place an absolute or symbolic address in the entry buffer; then, choose **Display→Memory→Hex ()** and select the size from the cascade menu.

This command displays memory as hexadecimal values beginning at the address in the entry buffer.

All sizes may not be available for all processors, or special sizes may exist for certain processors. The number of bytes actually in a word or long is also dependent on the processor. Refer to your processor-specific documentation for more information about types and sizes.



To display memory in real number format

- Place an absolute or symbolic address in the entry buffer; then, choose **Display→Memory→Real ()** and select the size from the cascade menu.

Displays memory as a list of real number values beginning at the address in the entry buffer. Short generally means four byte real numbers and long means eight byte real numbers. Some processors may define additional real number types or different byte counts. Refer to your processor-specific documentation for more information about real types.

To display memory at an address

- Place an absolute or symbolic address in the entry buffer; then, choose **Display→Memory→At ()**.

This command displays memory in the same format as that of the last memory display command. If no previous command has been issued, memory is displayed as hexadecimal bytes.

To display memory repetitively

- Choose **Display**→**Memory**→**Repetitively**.

The memory display is constantly updated. The format is specified by the last memory display command.

This command is ignored if the last memory display command was a mnemonic display.

To modify memory

- Choose **Modify**→**Memory** and complete the command using the command line.
- To modify memory at a particular address, place an absolute or symbolic address in the entry buffer; then, choose **Modify**→**Memory at ()** and complete the command using the command line.
- Using the command line, enter the **modify memory** command.

You can modify the contents of one memory location or a range of memory locations. Options allow you to modify memory in byte, short, word, and real number formats.

Displaying Data Values

The data values display lets you view the contents of memory as data types. You can display data values in the following formats:

- bytes
- 8-bit integers
- unsigned 8-bit integers
- chars
- words
- 16-bit integers
- unsigned 16-bit integers
- long words
- 32-bit integers
- unsigned 32-bit integers

This section shows you how to:

- Display data values.
- Clear the data values display and add a new item.
- Add item to the data values display.

To display data values

- Choose **Display**→**Data Values**.

Items must be added to the data values display before you can use this command.

The data display shows the values of simple data types in the user program. When the display mode setting turns ON symbols, a label column that shows symbol values is added to the data display.

Step commands and commands that cause the emulator to enter the monitor (for example, encountering a breakpoint) cause the data values screen to be updated.

To clear the data values display and add a new item

- Place an absolute or symbolic address in the entry buffer; then, choose **Display→Data Values→New ()** and select the data type from the cascade menu.

To add items to the data values display

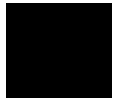
- Place an absolute or symbolic address in the entry buffer; then, choose **Display→Data Values→Add ()** and select the data type from the cascade menu.

Displaying and Modifying Registers

You can display and modify the contents of emulation processor registers. Most emulators have at least a BASIC class of registers. Some emulators have additional register classes whose register contents can be displayed and modified. Consult your emulator-specific Softkey Interface documentation for a definition of the register classes.

This section shows you how to:

- Display register contents.
- Modify register contents.



To display register contents

- Choose **Display**→**Registers**.

Displays at least the BASIC class of registers. Some processors will have additional register classes that can be displayed. If that is so, this menu item may have a cascade menu attached to it.

To modify register contents

- Choose **Modify**→**Registers...** and use the dialog box to name the register and specify its value.

Clicking the "Recall" pushbutton lets you select register names and values from predefined or previously specified entries.

Placing the mouse pointer in the text entry area lets you type in the register name and value.

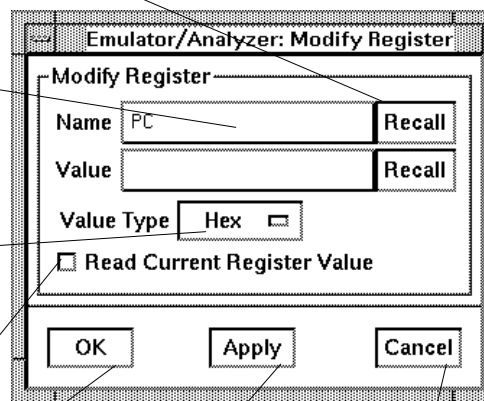
To define the type of value, press and hold the *command select* mouse button and drag the mouse to select the value type.

Clicking this checkbox causes the current value of the named register to be placed in the "Value" text entry area.

Clicking this button modifies the register to the value specified and closes the dialog box.

Clicking this button modifies the register to the value specified and leaves the dialog box open.

Clicking this button cancels modification and closes the dialog box.



Executing Programs

You can use the emulator to run programs, break program execution into the monitor, step through the program by high-level source lines or by assembly language instructions, and reset the emulation processor.

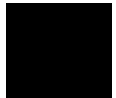
When displaying memory in mnemonic format, a highlighted bar shows the current program counter address. When you step, the mnemonic memory display is updated to highlight the new program counter address.

When displaying registers, the register display is updated to show you the contents of the registers after each step.

You can open multiple interface windows to display memory in mnemonic format and registers at the same time. Both windows are updated after stepping.

This section shows you how to:

- Run programs from the current PC.
- Run programs from an address.
- Run programs from the transfer address.
- Run programs from reset.
- Run programs until an address.
- Step high-level source lines.
- Step assembly-level instructions.
- Stop (break from) program execution.
- Reset the emulation processor.



To run programs from the current PC

- Choose **Execution**→**Run**→**from PC**.

When the emulator is executing the user program, the message "Running user program" is displayed on the status line.

To run programs from an address

- Position the mouse pointer in the entry buffer and enter the address you want to run from; then, choose **Execution**→**Run**→**from ()**.

To run programs from the transfer address

- Choose **Execution**→**Run**→**from Transfer Address**.

Most software development tools allow you to specify a starting or entry address for program execution. That address is included with the absolute file's symbolic information and is known by the interface as the "transfer address".

Before you can run from the transfer address, it must exist in the absolute file, and you must load symbols along with the program code from the absolute file. If the interface does not detect a transfer address, this menu item is grayed-out and unresponsive to mouse clicks.

To run programs from reset

- Choose **Execution**→**Run**→**from Reset**.

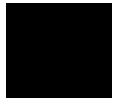
This command resets the emulation processor and begins executing the user program at either the start address for the processor, or at the address fetched from the reset vector for the processor. It may be necessary to supply a reset signal from your target system as well. See your processor-specific documentation for information about the exact mechanism involved.

To run programs until an address

- When displaying memory in mnemonic format, position the mouse pointer over the line that you want to run until; then press and hold the *select* mouse button and choose **Run Until** from the popup menu.
- Position the mouse pointer in the entry buffer and enter the address you want to run from; then, choose **Execution**→**Run**→**until** ().

When you run until an address, a breakpoint is set at the address and the program is run from the current program counter.

When using the command line, you can combine the various types of run commands; for example, you can run from the transfer address until another address.



To step high-level source lines

- To step a program one high-level source line from the current program counter, choose **Execution→Step Source→from Current PC**.
- To step a program one high-level source line from a specific address, place an absolute or symbolic address in the entry buffer and then choose **Execution→Step Source→from ()**.
- To step a program one high-level source line from the transfer address, choose **Execution→Step Source→from Transfer Address**.

Because one high-level source line can equal many lines of assembly code, stepping one high-level line may execute a number of assembly language instructions. This is particularly true in the case of "stepping from transfer address". The interface may have to execute several hundred assembly instructions of startup code before stepping through the first line of your source code. It might be better to set a breakpoint, and then run until the breakpoint is reached, instead of using a step command. See "To run until an address" for an easy way to do this.

If you are displaying memory in mnemonic format, a step command causes a highlight to appear in the memory display to indicate the location of the current program counter.

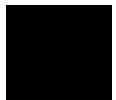
You should not use "step source" to step through large sections of assembly language instructions or step through code generated from files containing only assembly language instructions. The interface is looking for high-level language source lines and will infinitely search the source file in an attempt to find high-level source lines.

When stepping through instructions associated with source lines, execution can remain in a loop and the message "Stepping source line 1; Next PC: <address>" is displayed on the status line. In this situation you can abort the step command by pressing <CTRL>c.

To step assembly-level instructions

- To step a program one assembly language instruction from the current program counter, choose **Execution→Step Instruction→from Current PC**.
- To step a program one assembly language instruction from a specific address, place an absolute or symbolic address in the entry buffer; then, choose **Execution→Step Instruction→from ()**.
- To step a program one assembly language instruction from the transfer address, choose **Execution→Step Instruction→from Transfer Address**.

This command executes a single assembly language instruction. If you are displaying memory in mnemonic format, a step command causes a highlight to appear in the memory display to indicate the location of the current program counter.



To stop (break from) program execution

- Choose **Execution→Break**.

This command stops program execution and breaks to the monitor.

It may be necessary to break to the emulation monitor in order to perform certain operations—especially if the emulator is restricted to real-time runs. Refer to your processor-specific documentation for more information.

Breakpoints and the "running until an address" command allow you to stop execution at particular points in the user program.

To reset the emulation processor

- Choose **Execution**→**Reset**.

The "reset" command causes the processor to be held in a reset state until a "break", "run", or "step" command is entered. A CMB execute signal will also cause the emulator to run if reset. Also, a request to access memory or registers while reset may cause a break into the monitor.



Using Breakpoints

Breakpoints allow you to stop target program execution at a particular address and transfer control to the emulation monitor. Set breakpoints at the first word of program instructions. Otherwise, your breakpoint instruction may be interpreted as data and no breakpoint cycle will occur.

Note

Version A.04.00 or greater of the HP 64700 system firmware provides support for permanent as well as temporary breakpoints. If your version of HP 64700 system firmware is less than A.04.00, only temporary breakpoints are supported.

When the emulator executes a breakpoint instruction in your program, execution stops immediately. If the breakpoint is temporary, it is inactivated. If the breakpoint is permanent, it remains active.

The breakpoints feature can be enabled or disabled.

When the breakpoints feature is enabled, you can set breakpoints using the pulldown menu or by clicking the *select* mouse button when the pointer is over a line in the mnemonic memory display.

Active breakpoints are indicated by asterisks in the mnemonic memory display, and currently defined breakpoints are listed in the breakpoints display.

You can deactivate permanent breakpoints. You can also deactivate temporary breakpoints that are currently pending (temporary breakpoints that have been set but not encountered during program execution). You can clear breakpoints when they are no longer needed.

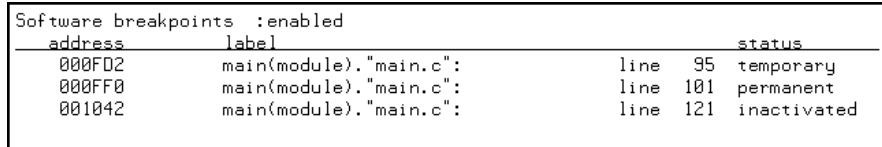
This section shows you how to:

- Display the breakpoints list.
- Enable/disable breakpoints.
- Set a permanent breakpoint.
- Set a temporary breakpoint.
- Set all breakpoints.
- Deactivate a breakpoint.

- Re-activate a breakpoint.
- Clear a breakpoint.
- Clear all breakpoints.

To display the breakpoints list

- Choose **Display**→**Breakpoints** or **Breakpoints**→**Display**.



```
Software breakpoints :enabled
address      label
-----
000FD2      main(module). "main.c":      line 95  temporary
000FF0      main(module). "main.c":      line 101 permanent
001042      main(module). "main.c":      line 121 inactivated
```

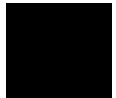
The breakpoints display shows the address and status of each breakpoint currently defined. If symbolic addresses are turned on (when setting the display modes), the symbolic label associated with a breakpoint is also displayed.

The status of a breakpoint can be:

- | | |
|-------------|---|
| temporary | Which means the temporary breakpoint has been set but not encountered during program execution. These breakpoints are removed when the breakpoint is encountered. |
| permanent | Which means the permanent breakpoint is active. Permanent breakpoints remain active after they are encountered during execution. |
| inactivated | Which means the breakpoint has been inactivated. Pending breakpoints are inactivated when they are encountered during program execution. Both temporary and permanent breakpoints can be inactivated (and restored) using the breakpoints display popup menu. |

pending Which means the temporary breakpoint has been set but not encountered during program execution. When encountered, these breakpoints are inactivated, but retained in the breakpoints list. Pending breakpoints can only be set using the softkey command line with commands like **modify software_breakpoints set 1000** and not selecting the additional options **<temporary>** or **<permanent>**. The "pending" breakpoints status is retained for compatibility with older product software versions.

In the breakpoints display, a popup menu is available, obtained by pressing the *select* mouse button. You can inactivate or restore the status of any breakpoint in the breakpoints list, as well as enable or disable the breakpoints feature, using the popup menu.



To enable/disable breakpoints

- Choose the **Breakpoints→Enable** toggle.
- When displaying the breakpoint list, press and hold the *select* mouse button and then choose **Enable/Disable Software Breakpoints** from the popup menu.

The breakpoints feature must be enabled before you can set, inactivate, or clear breakpoints.

If breakpoints were set when the feature was disabled, they are "inactivated" when the feature is re-enabled, and you must set them again.

The **Breakpoints→Enable** toggle is a switch. Use it to enable and disable breakpoints.

Chapter 5: Using the Emulator/Analyzer Using Breakpoints

Examples

To enable software breakpoints using the breakpoints display popup menu:

The screenshot shows the Hewlett Packard Emulator/Analyzer interface for the em68000 (m68000) processor. The main window displays a table of software breakpoints. A context menu is open over the table, highlighting the 'Enable/Disable Software Breakpoints' option. An annotation points to this menu item with the text: 'Bring up menu and choose this item to change states.'

address	label	line	status
000FD2	main(module).\"main.c\":	95	inactivated
000FF0	main(module).\"main.c\":	101	inactivated
001042	main(module).\"main.c\":	121	inactivated

STATUS: H68000--Running in monitor Software break: 000fd2@sp

To set a permanent breakpoint

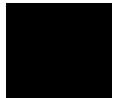
- When displaying memory in mnemonic format, position the mouse pointer over the program line at which you wish to set the breakpoint and click the *select* mouse button. Or, press and hold the *select* mouse button and choose **Set/Clear Software Breakpoint** from the popup menu.
- Place an absolute or symbolic address in the entry buffer; then, choose **Breakpoints→Permanent ()**

Permanent breakpoints are available if your version of HP 64700 system firmware is A.04.00 or greater.

The breakpoints feature must be enabled before individual breakpoints can be set.

Note that you must only set software breakpoints at memory locations which contain instruction opcodes (not operands or data).

When displaying memory in mnemonic format, asterisks (*) appear next to breakpoint addresses. An asterisk shows the breakpoint is active. Also, if assembly level code is being displayed, the disassembled instruction mnemonic at the breakpoint address will show the breakpoint instruction.



Chapter 5: Using the Emulator/Analyzer Using Breakpoints

Examples

To set permanent breakpoints using the mnemonic memory display popup menu:

Click this line to set a breakpoint.

Click this line to clear a breakpoint. (Asterisks indicate breakpoints that are set.)

Bring up menu and choose this item to set (or clear) a breakpoint on the highlighted line.

The screenshot shows the Hewlett Packard Emulator/Analyzer interface for the em68000 (m68000) processor. The main window displays the source code for the file "main.c". The code is as follows:

```
Memory :mnemonic :file = main(module)."main.c":
address label      data
90  extern void update_system();      /* update system variables */
91  extern void interrupt_sim();      /* simulate an interrupt */
92  extern void do_sort();            /* sets up ascii array and calls c
93
94  main()
95  {
96      init_system();
97      proc_spec_init();
98
99      while (true)
100     {
* 101         update_system();
102         num_checks++;
103         interrupt_sim(&num_checks);
104         if (graph)
105             graph_data();
106         proc_specific();
```

The line 103, `interrupt_sim(&num_checks);`, is highlighted. A context menu is open over this line, showing the following options:

- Choose Action for Highlighted Line
- Set/Clear Software Breakpoint
- Edit Source
- Run Until

The status bar at the bottom of the window displays "STATUS: M68000--Running in monitor".

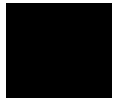
To set a temporary breakpoint

- Place an absolute or symbolic address in the entry buffer; then, choose **Breakpoints**→**Temporary** () (or **Breakpoints**→**Set** () if your version of HP 64700 system firmware is less than A.04.00).

The breakpoints feature must be enabled before individual breakpoints can be set.

Note that you must only set software breakpoints at memory locations which contain instruction opcodes (not operands or data).

When displaying memory in mnemonic format, asterisks (*) appear next to breakpoint addresses. An asterisk shows the breakpoint is active. Also, if assembly level code is being displayed, the disassembled instruction mnemonic at the breakpoint address will show the breakpoint instruction.



To set all breakpoints

- When displaying the breakpoint list, position the mouse pointer within the breakpoints display screen, press and hold the *select* mouse button, and choose **Set All Breakpoints** from the popup menu.
- Choose **Breakpoints**→**Set All**.

Breakpoints must be enabled before being set.

To deactivate a breakpoint

- When displaying breakpoints, position the mouse pointer over the line displaying the active breakpoint and click the *select* mouse button. Or, press and hold the *select* mouse button and choose **Set/Inactivate Breakpoint** from the popup menu.

A deactivated breakpoint remains in the breakpoint list and can be re-activated later. Deactivating a breakpoint is different than clearing a breakpoint because a cleared breakpoint is removed from the breakpoints list.

To re-activate a breakpoint

- When displaying breakpoints, position the mouse pointer over the line displaying the inactivated breakpoint and click the *select* mouse button. Or, press and hold the *select* mouse button and choose **Set/Inactivate Breakpoint** from the popup menu.

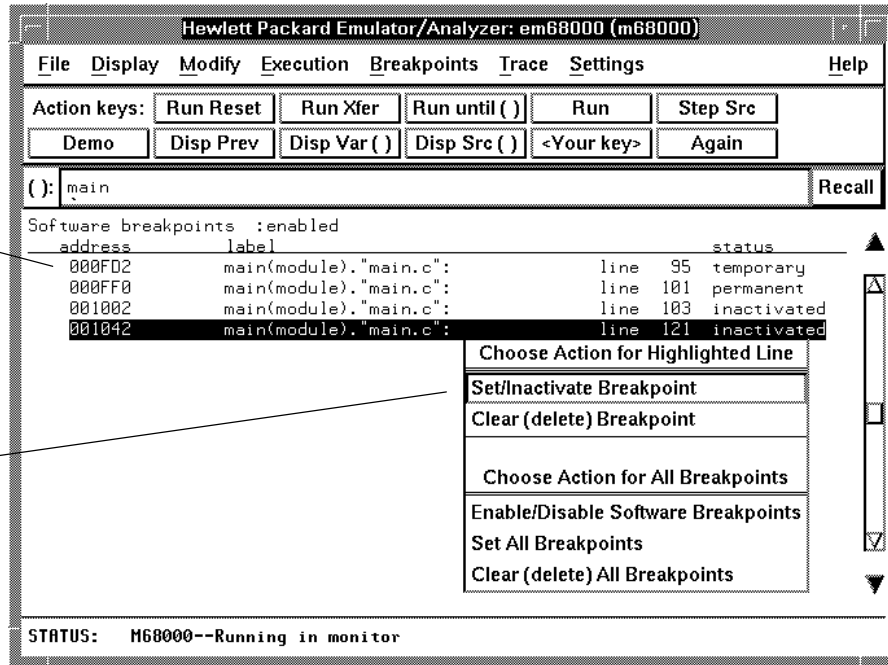
The "inactivated" breakpoint either becomes "temporary" (or "pending") if it was set as a temporary breakpoint or "permanent" if it was set as a permanent breakpoint.

Examples

To re-activate breakpoints using the breakpoints display popup menu:

Change status with a mouse click on this line (menu and highlight do not appear).

Choose this menu item to change the state of the highlighted breakpoint.



To clear a breakpoint

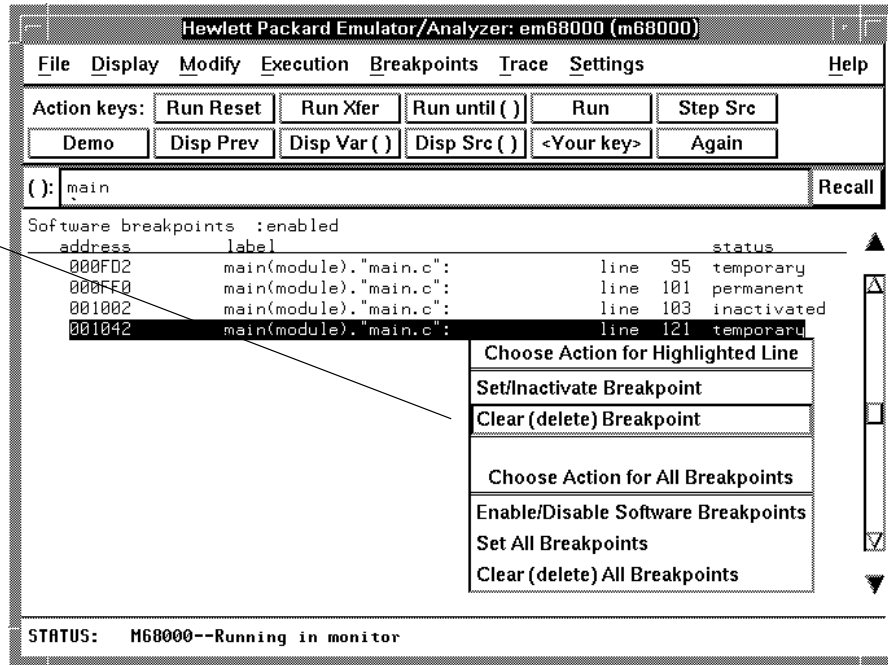
- When displaying memory in mnemonic format, position the mouse pointer over the program line at which you wish to clear a currently set breakpoint (notice the asterisk at the left of the line) and click the *select* mouse button. Or, press and hold the *select* mouse button and choose **Set/Clear Software Breakpoint** from the popup menu.
- When displaying breakpoints, position the mouse pointer over the line displaying the breakpoint you wish to clear, press and hold the *select* mouse button, and choose **Clear (delete) Breakpoint** from the popup menu.
- Place an absolute or symbolic address in the entry buffer; then choose **Breakpoints→Clear ()**.

When you clear a breakpoint, it is removed from the breakpoints list.

Examples

To clear a software breakpoint using the breakpoints display popup menu:

Bring up the menu and choose this item to clear the highlighted breakpoint.



To clear all breakpoints

- When displaying breakpoints, position the mouse pointer within the Breakpoints Display screen, press and hold the *select* mouse button, and choose **Clear (delete) All Breakpoints** from the popup menu.
- Choose **Breakpoints**→**Clear All**.

Tracing Programs

The Graphical User Interface has menus that let you specify some simple analyzer measurements like tracing after, about, or before an address. You can also qualify which states get stored and prestore two states before qualified store states.

However, the analyzer has a great deal more capability than is available in the menus. You can access this capability by using the command line to make trace specifications. Refer to your emulator *User's Guide* for information on using the command line to make more complex trace specifications.

Once a trace specification command is entered, either with the menus or the command line, it can be recalled using the graphical interface. Also, trace specifications and trace data can be stored to, and loaded from, files.

This section shows you how to:

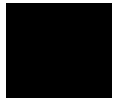
- Trace after any state.
- Display the trace listing.
- Trace after an address.
- Trace before an address.
- Trace before an address and break on the trigger.
- Trace about an address.
- Qualify which states get stored.
- Prestore states before qualified store states.
- Trace until the trace is stopped.
- Stop a trace.
- Repeat the last trace.
- Repeat the last trace repetitively.
- Recall trace specification commands.
- Store trace data to a file.

- Load a trace data file.
- Store a trace specification to a file.
- Load a trace specification file.

To trace after any state

- Choose **Trace**→**Everything**.

Any captured state triggers the analyzer, and all states captured after the trigger state are stored until trace memory is full.



To display the trace listing

- Choose **Trace**→**Display**
- Choose **Display**→**Trace**

Whether source lines, disassembled trace states, or symbols are included in the display depends on the modes you choose with the **Settings**→**Source/Symbol Modes** or **Settings**→**Display Modes** pulldown menu items. See the "Changing the Interface Settings" section.

To trace after an address

- Place an absolute or symbolic address in the entry buffer; then, choose **Trace→After ()**.
- When displaying memory in mnemonic format, position the mouse pointer over the program line which you wish to trace after, press and hold the *select* mouse button and choose **Trace After** from the popup menu.

The analyzer triggers when the specified address is captured. Bus states that occur after the trigger are stored in trace memory. Consequently, the trigger state appears at the top of the trace listing.

Examples

Notice that states after the trigger address, "update_system", are stored.

Trace List		Offset=0	More data off screen	
Label:	Address	Opcode or Status w/ Source Lines	time count	
Base:	symbols	mnemonic w/symbols	relative	
#####update_sys.c - line 1 thru 46 #####				
		#include <stdio.h>		
		void		
		update_system()		
		{		
after	up.update_system	LINK A6, #00000		-----
+001	sysstack+003F8C	0000 supr data wr word	400	nS
+002	sysstack+003F8E	0FFC supr data wr word	400	nS
+003	update_sy+000002	0000 supr prog	400	nS
+004	update_sy+000004	MOVEM.L D2-D4/A2-A4, -(A7)	400	nS
+005	sysstack+003F88	0000 supr data wr word	400	nS
+006	sysstack+003F8A	EF94 supr data wr word	400	nS
+007	update_sy+000006	3838 supr prog	400	nS
+008	update_sy+000008	MOVEA.L #00000744, A2	400	nS
+009	sysstack+003F86	783C supr data wr word	400	nS

To trace before an address

- Place an absolute or symbolic address in the entry buffer; then, choose **Trace**→**Before** ().
- When displaying memory in mnemonic format, position the mouse pointer over the program line which you wish to trace after, press and hold the *select* mouse button and choose **Trace Before** from the popup menu.

The analyzer triggers when the specified address is captured. Bus states that occur before the trigger are stored in trace memory. Consequently, the trigger state appears at the bottom of the trace listing.

Examples

Notice that states before the trigger address, "update_system", are stored.

Trace List		Offset=0	More data off screen	
Label:	Address	Opcode or Status w/	Source Lines	time count
Base:	symbols	mnemonic w/symbols	relative	
	{			
	update_system();			
-007	prog main+00001E	MOVE.B #001,00001(R2)		600 nS
-006	prog main+000020	0001 supr prog		400 nS
-005	prog main+000022	0001 supr prog		400 nS
-004	prog main+000024	JSR up.update_system		400 nS
-003	_bA_array+000001	01 supr data wr byte		400 nS
-002	prog main+000026	0000 supr prog		400 nS
-001	prog main+000028	174C supr prog		400 nS
	#####update_sys.c - line 1 thru 46 #####			
	#include <stdio.h>			
	void			
	update_system()			
	{			
before	up.update_system	LINK R6,####		400 nS

To trace before an address and break on the trigger

- Place an absolute or symbolic address in the entry buffer; then, choose **Trace→Until ()**.
- When displaying memory in mnemonic format, position the mouse pointer over the program line which you wish to trace before, press and hold the *select* mouse button and choose **Trace Until** from the popup menu.

The analyzer triggers when the specified address is captured, and the program execution breaks when the trigger occurs. Bus states that occur before the trigger are stored in trace memory. Consequently, the trigger state appears at the bottom of the trace listing.

Note that the actual break may be several cycles after the analyzer trigger.

To trace about an address

- Place an absolute or symbolic address in the entry buffer; then, choose **Trace**→**About ()**.
- When displaying memory in mnemonic format, position the mouse pointer over the program line which you wish to trace after, press and hold the *select* mouse button and choose **Trace About** from the popup menu.

The analyzer triggers when the specified address is captured. Bus states that occur before and after the trigger are stored in trace memory. Consequently, the trigger state appears in the middle of the trace listing.

Examples

Notice that states before and after the trigger address, "update_system", are stored.

Trace List		Offset=0		More data off screen	
Label:	Address	Opcode or Status w/	Source Lines	time count	
Base:	symbols	mnemonic w/symbols		relative	
-007	prog main+00001E	MOVE.B	#001,00001(A2)	600	nS
-006	prog main+000020	0001	supr prog	400	nS
-005	prog main+000022	0001	supr prog	400	nS
-004	prog main+000024	JSR	up.update_system	400	nS
-003	_bA_array+000001	01	supr data wr byte	400	nS
-002	prog main+000026	0000	supr prog	400	nS
-001	prog main+000028	174C	supr prog	400	nS
#####update_sys.c - line 1 thru 46 #####					
#include <stdio.h>					
void					
update_system()					
{					
about	up.update_system	LINK	A6,#00000	400	nS
+001	sysstack+003F8C	0000	supr data wr word	400	nS
+002	sysstack+003F8E	0FFC	supr data wr word	400	nS

To qualify which states get stored

- Place an absolute or symbolic address in the entry buffer, and choose **Trace→Only ()**.

Only bus states that match the address placed in the entry buffer are stored in the trace.

To use data and status qualifiers as well as address qualifiers, or to store only states in a range or outside a range, use the command line to make the trace specification.

Examples

Storing only accesses to the address "target_temp" results in the following display:

Trace List		Offset=0		More data off screen	
Label:	Address	Opcode or Status w/ Source Lines		time count	
Base:	symbols	mnemonic w/symbols		relative	
after	set_output+00003E	ORI.B	###, (R3)	-----	
+001	dat _target_temp	0058	supr data rd word	37.4	mS
+002	dat _target_temp	0058	supr data rd word	5.80	uS
+003	dat _target_temp	0058	supr data rd word	2.4	uS
+004	dat _target_temp	0058	supr data rd word	2.8	uS
+005	dat _target_temp	0058	supr data rd word	50.4	mS
+006	dat _target_temp	0057	supr data wr word	800	nS
+007	dat _target_temp	0057	supr data rd word	800	nS
+008	dat _target_temp	0057	supr data rd word	5.38	mS
+009	dat _target_temp	0057	supr data rd word	57.7	mS
+010	dat _target_temp	0057	supr data rd word	5.80	uS
+011	dat _target_temp	0057	supr data rd word	2.4	uS
+012	dat _target_temp	0057	supr data rd word	2.8	uS
+013	dat _target_temp	0057	supr data rd word	29.9	mS
+014	dat _target_temp	0056	supr data wr word	800	nS
+015	dat _target_temp	0056	supr data rd word	800	nS

To prestore states before qualified store states

- Place an absolute or symbolic address in the entry buffer, and choose **Trace→Only () Prestore**.

Only bus states that match the address placed in the entry buffer and the state or states that precede it are stored in the trace. If you are using the emulation-bus analyzer with deep trace memory (HP 64794), one preceding (prestore) state will be stored in the trace. If you are using a 1K emulation-bus analyzer (HP 64703, HP 64704, or HP 64706), two preceding (prestore) states will be stored in the trace.

Prestore is useful, for example, to identify code that is corrupting a variable.

To qualify the states that are prestored, use the command line to make the trace specification.

Examples

Storing only accesses to the address "target_temp" and prestoring the two previous states results in the following display:

Trace List	Offset=0	More data off screen	
Label:	Address	Opcode or Status w/ Source Lines	time count
Base:	symbols	mnemonic w/symbols	relative
after	set_output+00003A	MOVE.B #**,****(A2)	
pstore	set_output+00004E	CMPR.W (A4),A0	
pstore	set_output+000050	BGT.W*****	
+003	dat _target_temp	0055 supr data rd word	15.2 mS
pstore	dat _func_needed	09 supr data wr byte	
pstore	set_output+000062	CMPR.W 03,A0	
+006	dat _target_temp	0055 supr data rd word	5.80 uS
#####update_sys.c - line 180 #####			
if (temperature >= target_temp)			
pstore	set_output+000074	MOVER.W (A4),A0	
pstore	set_output+000076	CMPR.W 03,A0	
+009	dat _target_temp	0055 supr data rd word	2.4 uS
pstore	set_output+00008A	CMPR.W (A4),A0	
pstore	set_output+00008C	BGE.W*****	
+012	dat _target_temp	0055 supr data rd word	2.8 uS
pstore	update_sy+000009	01 supr data wr byte	

To trace until the trace is stopped

- Choose **Trace→Until Stop**.

This command causes the analyzer to continuously fill the trace buffer until you issue a **Trace→Stop**.

The "trace until stop" command is used to capture the states that lead up to some condition that causes the microprocessor to stop executing instructions, for example, breaks on guarded memory accesses or breaks on writes to ROM.

Once the break or halt occurs, there are no more states to be captured and stored in trace memory. Consequently, when you stop the trace, the buffer contains the states that led up to the break or halt.

Note that the "trace until stop" command is not useful when using a foreground monitor (unless the code that causes the break also causes the processor to halt) because the analyzer continues to fill the trace buffer with foreground monitor states after a break. In this case, you can use the command line to enter a trace command that stores only states outside the range of the foreground monitor program (for example, **trace only not range <mon_start_addr> thru <mon_end_addr> on_halt**).

To stop a trace

- Choose **Trace→Stop**.

Stops the current trace. You must use this command to stop a trace started with a **Trace→Until Stop** command. Otherwise, stopping the trace is usually not necessary. It may be that you have specified a trigger condition that will never occur. In that case, you can stop the trace before re-specifying it.

You do not have to stop a trace in order to begin viewing a partial trace because the interface supports incremental trace uploading. After the trigger condition occurs, the interface begins uploading and displaying trace states as they are captured.

To continually repeat the last trace

- Choose **Trace**→**Repetitively**.

Continually repeats the last trace command. Successive traces begin as soon as the results from the just-completed trace are displayed.

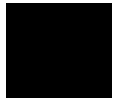
To recall trace specification commands

- Choose **Trace**→**Trace Spec ...** and use the selection dialog box to recall, edit, and enter the command.

The Trace Specification Selection dialog box lets you recall, edit, and enter trace commands that have been executed during the emulation session or trace commands that have been predefined.

You can predefine entries for the Trace Specification Selection dialog box and define the maximum number of entries by setting X resources (refer to the "Setting X Resources" chapter).

See the "To use dialog boxes" description in the "Entering Commands" chapter for information about using selection dialog boxes.



To store trace data to a file

- 1 Choose **File**→**Store**→**Trace Data ...**
- 2 Choose an existing trace data file or specify a new file using the displayed File Selection dialog box.

Stores a trace listing in a file in a binary format that can later be reloaded into the interface and displayed like any other trace listing. The interface will always append the suffix ".TR" to a trace data file. For that reason, you should only specify the file base name when creating a new file.

To load a trace data file

- 1 Choose **File**→**Load**→**Trace Data ...**
- 2 Choose the trace data file using the displayed File Selection dialog box.

Loads a previously saved trace from a binary trace data file (with a ".TR" suffix).

The interface will try to display the trace listing in the display format active when the trace data was stored. If the interface needs symbols to replace absolute addresses or to find high-level source lines, and symbols are not loaded, an error occurs.

For example, suppose "source-mixed" was the display mode when the trace was captured and the executable file "test1" was the file being executed in the emulator/target system. To reload and display a trace listing saved from that emulation session requires reloading the symbols for "test1".

To store a trace specification to a file

- 1 Choose **File**→**Store**→**Trace Spec...**
- 2 Choose an existing trace specification file or specify a new file using the displayed File Selection dialog box.

Stores the most recently executed trace specification in a special format file. The specification can be reloaded and reused at some later time. The interface will always append the suffix ".TS" to a trace specification file. For that reason, you should only specify the file base name when creating a new file.

To load a trace specification file

- 1 Choose **File**→**Load**→**Trace Spec...**
- 2 Choose the trace specification file using the displayed File Selection dialog box.

Loads a previously saved trace specification so the trace command can be repeated. The trace specification is also placed in the Trace Specification Selection dialog box.

Loading a trace specification does not start the trace; to do this, you must enter the trace command either by selecting it from the Trace Specification Selection dialog box or by using the **Trace**→**Again** pulldown menu item.

Changing the Interface Settings

This section shows you how to:

- Set the source/symbol modes.
- Set the display modes.

To set the source/symbol modes

- To display assembly language mnemonics with absolute addresses, choose **Settings→Source/Symbol Modes→Absolute**.
- To display assembly language mnemonics with absolute addresses replaced by global and local symbols where possible, choose **Settings→Source/Symbol Modes→Symbols**.
- To display assembly language mnemonics intermixed with high-level source lines, choose **Settings→Source/Symbol Modes→Source Mixed**.
- To display only high-level source lines, choose **Settings→Source/Symbol Modes→Source Only**.

The source/symbol modes affect mnemonic memory displays and trace displays.

Each display mode cascade menu choice is a toggle. Choosing one of these items causes it to be the only one active and toggles all others off. Provided that symbols were loaded, the interface defaults to:

- Source only for mnemonic memory displays.
- Source mixed for trace listing displays.

To set the display modes

- Choose **Settings**→**Display Modes...** to open the display modes dialog box.

Press and hold the *select* mouse button and drag the mouse to select "Source Only", "Source Mixed", or "Off".

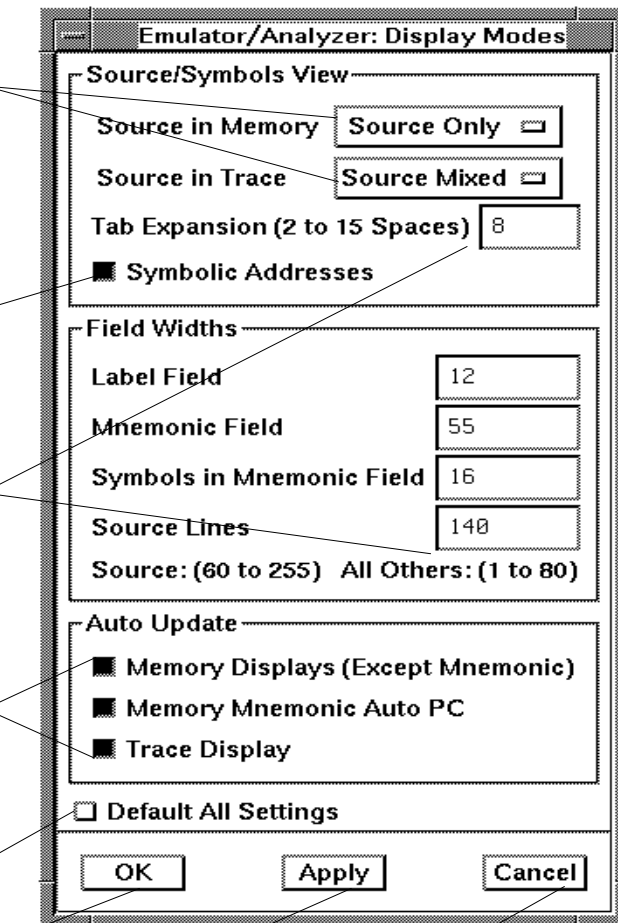
Clicking toggles whether symbolic information is displayed.

Move the mouse pointer to the text entry area and type in the value. Descriptions of the modes follow.

Clicking toggles auto update settings.

Clicking this checkbox changes all display mode settings to their defaults.

Clicking this button saves your changes and closes the dialog box.



Clicking this button saves your changes and leaves the dialog box open.

Clicking this button cancels your changes and closes the dialog box.

Source/Symbols View

Source in Memory specifies whether source lines are included, mixed with assembly code, or excluded from mnemonic memory displays.

Source in Trace specifies whether source lines are included, mixed with stored states, or excluded from trace displays.

Symbolic Addresses specifies whether symbols are included in displays.

Tab Expansion sets the number of spaces displayed for tabs in source lines.

Source/Symbols View

Label Field sets the width (in characters) of the address field in the trace list or label (symbols) field in any of the other displays.

Mnemonic Field sets the width (in characters) of the mnemonic field in memory mnemonic, trace list, and register step mnemonic displays. It also changes the width of the status field in the trace list.

Symbols in Mnemonic Field sets the maximum width of symbols in the mnemonic field of the trace list, memory mnemonic, and register step mnemonic displays.

Source Lines sets the width (in characters) of the source lines in the memory mnemonic display.

Auto Update

Memory Displays (Except Mnemonic) toggles whether absolute memory displays are automatically updated after commands that change memory contents or whether you must enter memory display commands to update the display. You may wish to turn off memory display updates when displaying memory mapped I/O.

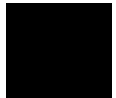
Memory Mnemonic Auto PC toggles whether memory mnemonic displays automatically jump to the new PC location when the PC changes (such as during stepping or break). You may wish to turn off the automatic update of memory mnemonic displays when you want to examine a specific area of memory regardless of the location of the current PC (such as during stepping).

Trace Displays toggles whether trace displays are automatically updated when trace measurements complete or whether you must enter trace display commands to update the display. You may wish to turn off trace display updates in one emulator/analyzer window in order to compare the display with a new trace display in another emulator/analyzer window.

Displaying the Trace List

The *trace list* is your view of the analyzer's record of processor bus activity. You can specify what is shown in the trace list to make it easier to find the information of interest. For example, you can display symbol information where available, or source lines from the high-level languages used to write the target program. You can also change the column widths and set options for disassembly of the trace list.

This section covers many of the options available for controlling the trace display. Display control is available through the **Trace→Display Options...** dialog box, and the trace list popup menu.



Chapter 5: Using the Emulator/Analyzer
Displaying the Trace List

Examples To use the Trace Options dialog box:

Click to select the desired format of trace disassembly.

Click to select the way that absolute status information is shown in the trace list.

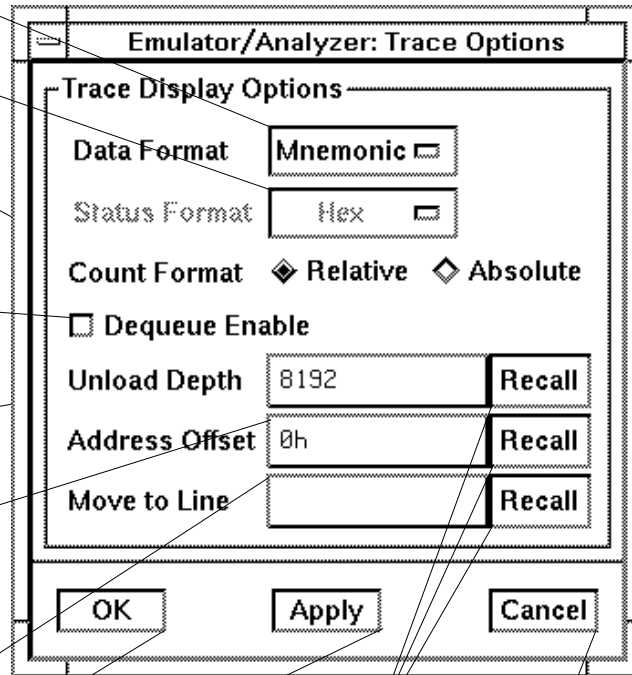
Click to select count reference: Relative (to preceding state), or Absolute (to trigger).

Click to select trace list dequeuing, if available for your emulator.

Enter the desired depth of the trace memory to be unloaded for display or storage in a file.

Enter a value to be subtracted from addresses and symbol/source-line references shown in the trace list.

Enter the desired trace list line number to be placed on screen.



Click OK to specify the trace options and close the dialog box.

Click Apply to specify the trace options and leave the dialog box open.

Click these pushbuttons to select predefined or previously specified entries.

Click this pushbutton to cancel the entries and close the dialog box.

Examples

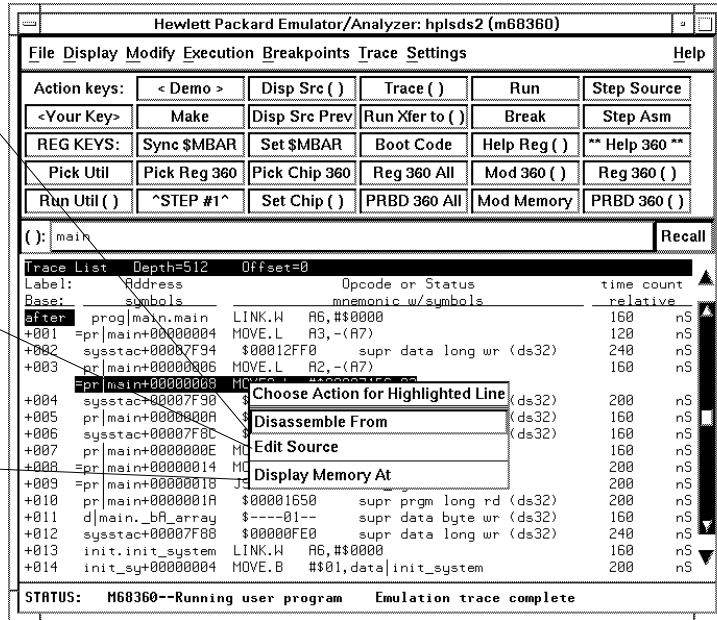
To use the trace list popup menu:

Click to begin trace disassembly from the selected line, moving that line to the top of the display. Note: "Disassemble From" is not available in all emulators.

Click to open an edit window into the source file that contains the address of the selected line.

Click to open a display window into memory containing the address of the selected line.

Note that the format of the memory display will be mnemonic for addresses in the code segment and absolute otherwise.



To disassemble the trace list

- Choose **Trace**→**Display Options...** and in the dialog box, select Data Format Mnemonic. Then click the OK or Apply pushbutton.
- Use the mouse to place the cursor on a line in the trace list where you want disassembly to begin. Then press the *select* mouse button, and click on **Disassemble From** in the trace list popup menu, if available for your emulator.

Disassembly of instruction data means that you will see instructions as they would appear in an assembly language program listing. That is, instruction mnemonics and operands are shown instead of hexadecimal instruction data.

The analyzer interface normally disassembles instruction data in the trace list. However, if you specify **Absolute** data format, that mode remains in effect until you select the **Mnemonic** option.

When you select a particular trace list line where disassembly is to begin, be sure to select a line that corresponds to an analyzer state with an opcode fetch. The analyzer interface disassembles and displays the trace starting with the state you specify.

To specify trace dequeuing options

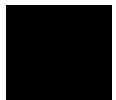
- Choose **Trace→Display Options...** and in the dialog box, select Dequeue Enable. Then click the OK or Apply pushbutton.

Trace dequeuing is not available in all emulators.

A dequeued trace list is available through the disassembly options. In a dequeued trace list, unused instruction prefetch cycles are discarded, and operand cycles are placed immediately following the corresponding instruction fetch. If you choose a non-dequeued trace list, instruction and operand fetches are shown exactly as captured by the analyzer.

Once the dequeuer has been started on the correct opcode, it will continue to disassemble correctly unless an unusual condition causes it to misinterpret the data.

Depending on the emulator you are using, you may see TAKEN, NOT TAKEN, or ?TAKEN? beside a branch in your dequeued trace list. TAKEN is shown beside a branch if the dequeuer determines that the branch was taken. NOT TAKEN is shown if the dequeuer determines that the branch was definitely not taken. ?TAKEN? means the dequeuer was not able to determine whether or not the branch was taken.



To display the trace without disassembly

- Choose **Trace→Display Options...** and in the dialog box, select Data Format Absolute. You can select Hex, Binary, or Mnemonic format for display of status information. Then click the OK or Apply pushbutton.

For some measurements, it may be more convenient for you to view the trace data without instruction disassembly. The Data Format Absolute selection in the **Trace→Display Options...** dialog box allows you to do this. Notice that once you enter this format selection, subsequent trace lists will be displayed in this format until you select the mnemonic format with the dialog box again.

You can select the display format for the status information when you choose Data Format Absolute in the dialog box. The status information can be displayed in binary, hex, or as mnemonics that indicate the nature of the current bus cycle (such as a read or write).

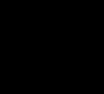
To display symbols in the trace list

- Choose **Settings→Source/Symbol Modes→Symbols**, or choose **Settings→Display Modes ...**, and in the dialog box, click on **Symbolic Addresses**. In the Field Widths area of the dialog box, you can select the widths of the Label Field and Symbols in Mnemonic Field to control the display space allocated to the symbols.

To display source lines in the trace list

- Choose **Settings**→**Source/Symbol Modes**→**Source Mixed** or **Settings**→**Source/Symbol Modes**→**Source Only** .
- Choose **Settings**→**Display Modes...**, and in the dialog box, click on **Source in Trace** and select either **Source Mixed** or **Source Only** from the submenu.

If you developed your target programs in a high-level language such as “C,” you can display the source code in the trace list with the corresponding assembly language statements. Or, you can choose to display only the source listing without the assembly language information.



To change the column width

- Choose **Settings**→**Display Modes...**, and select desired widths for information in the trace list by using the dialog box. Refer to "To display symbols in the trace list" earlier in this chapter for details of how to use the dialog box.

You can display more information by widening a column or ignore the information by narrowing the column. For example, you might want to widen the label column so that you can see the complete names of the symbols in that column.

To select the type of count information in the trace list

- Choose **Trace→Display Options...** and in the dialog box, select Count Format Relative or Absolute, as desired. Then click the OK or Apply pushbutton.

The count information in the trace list is always displayed if it is turned on. To turn on the trace counting function, enter a command beginning with **trace counting** on the command line.

When using the 1K analyzer, the trace memory is 512 states deep if counting states or time is turned on and 1024 states deep if counting is turned off. To disable counting in the 1K analyzer, enter **trace counting off** on the command line. When using the deep analyzer, full memory depth is always available; the depth of the deep analyzer is not affected by the counting selected.

To offset addresses in the trace list

- Choose **Trace→Display Options...** and in the dialog box, enter the desired offset value in the field beside Address Offset. Then click the OK or Apply pushbutton.

The Address Offset option allows you to cause the address information in the trace display to be offset by the amount specified. The offset value is subtracted from the instruction's physical address to yield the address that is displayed.

If code gets relocated and therefore makes symbolic information invalid, you can use the Address Offset option to change the address information so that it again agrees with the symbolic information.

You can also specify an offset to cause the listed addresses to match the addresses in compiler or assembler listings.

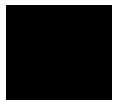
To reset the trace display defaults

- Choose **Settings**→**Display Modes...** Then in the dialog box, click on Default All Settings, and click the OK pushbutton. This leaves the trace display in the "source intermixed and symbols on" mode.

To display the trace list around a specific line number

- Choose **Trace**→**Display Options...** and in the dialog box, enter the desired trace list line number in the field beside Move to Line. Then click the OK or Apply pushbutton.

If you need to move to a particular state quickly, you can use this method. This places the specified state in the center of the current trace display.



To change the number of states available for display

- Choose **Trace→Display Options...** and in the dialog box, enter the desired number of states to be made available for display in the field beside Unload Depth. Then click the OK or Apply pushbutton.

This selects the number of states to be available in the trace list for displaying, copying, or storing to a file. If you are using the deep analyzer, the depth of the trace list buffer depends on whether or not you installed memory modules on the analyzer card, and the capacity of the memory modules installed. If you are using the 1K analyzer, the trace list buffer is 512 or 1024 states deep (depending on whether or not you turn on the state/time count).

When you display the trace list, the interface requests the number of states specified by the trace depth from the emulator. If you want faster trace performance, you can decrease the trace depth. To display more states, you can increase the trace depth. The trace depth setting only regulates the number of states sent from the emulation-bus analyzer to the interface. You still need to use the **Pg Up** and **Pg Dn** keys to page through the trace list.

You can change the Unload Depth specification after a trace is complete to view more information. For example, you may have captured 256K states in the deep analyzer, but specified an Unload Depth of only 8K states to improve system performance. After viewing the 8K states, you might specify unload of 16K states. When you increase the unload depth, the additional depth will be unloaded as long as a new trace has not been started. For normal measurements, specify an Unload Depth that obtains best performance. You can increase Unload Depth afterward, if needed. If you decrease Unload Depth to less than the present unloaded trace depth, the unload process will stop.

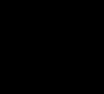
Trace data is always unloaded around the trigger position. For example, if you specified unload of 8K memory depth in the deep analyzer:

- If trigger is the first state in memory, the first 8K states in memory will be unloaded.
- If trigger is the center state in memory, the 4K states before trigger and the 4K states after trigger will be unloaded.
- If trigger is the last state in memory, the last 8K states will be unloaded.

To display program memory associated with a trace list line

- Using the mouse, place the cursor on the line in the trace list where you want to see the associated content of program memory. Then press the *select* mouse button, and click on **Display Memory At** in the trace list popup menu.

You will see a display of memory at the location of the program that emitted the selected trace list line. This is the same as placing the program address of the selected trace list line in the entry buffer and choosing **Display→Memory→At()** in the pulldown menus.



To open an edit window into the source file associated with a trace list line

- Using the mouse, place the cursor on the line in the trace list whose source file you wish to edit. Then press the *select* mouse button, and click on **Edit Source** in the trace list popup menu.

A new window will open. It will show the source file that emitted the line you selected in the trace list. An edit session will be in progress on the source file in the new window. When you complete the desired edit, save the file and close the window.

Using System Commands

This section shows you how to:

- Edit files.
- Display the error log.
- Display the event log.
- Display status.
- Copy information to a file or printer.
- Open a terminal emulation window.

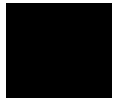
To edit files

- Choose **File**→**Edit**→**File** and use the dialog box to specify the file name.
- To edit a file based on an address in the entry buffer, place an address reference (either absolute or symbolic) in the entry buffer; then, choose **File**→**Edit**→**At () Location**.
- To edit a file based on the current program counter, choose **File**→**Edit**→**At PC Location**.
- To edit a file associated with a symbol when you are displaying symbols, position the mouse pointer over the symbol, press and hold the *select* mouse button, and choose **Edit File At Symbol** from the popup menu.
- To edit a file when displaying memory in mnemonic format, position the mouse pointer over the line of source where you want to begin the edit, press and hold the *select* mouse button, and choose **Edit Source** from the popup menu.

When editing files at addresses, the interface determines which source file contains the code generated for the address and opens an edit session on the file. The interface will issue an error if it cannot find a source file for the address.

The interface will choose the "vi" editor as its default editor, unless you specify another editor by setting an X resource. Refer to the chapter "Setting X Resources" for more information about setting this resource.

You must load symbols before most of these commands will work because symbol information is needed to locate the files.

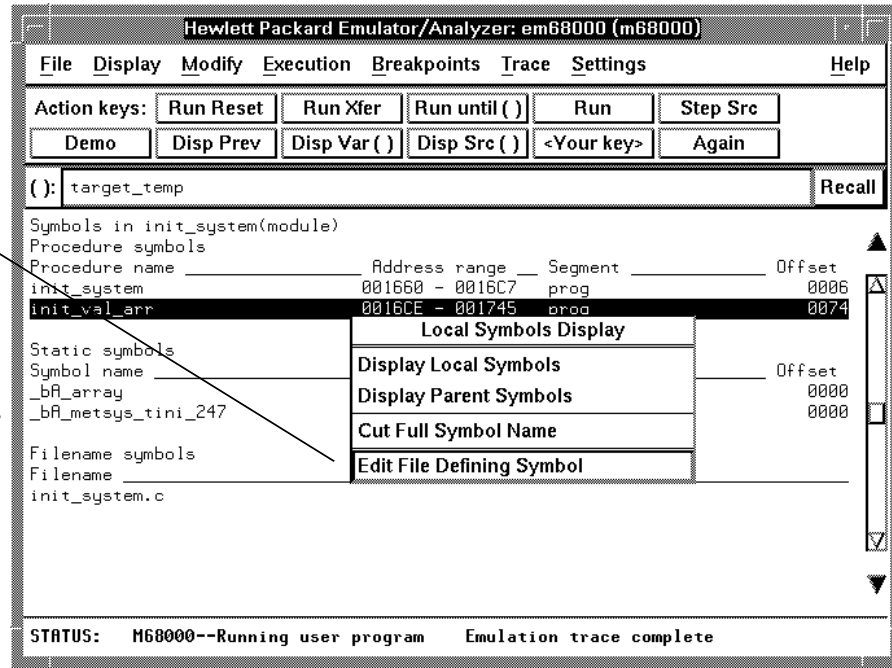


Chapter 5: Using the Emulator/Analyzer Using System Commands

Examples

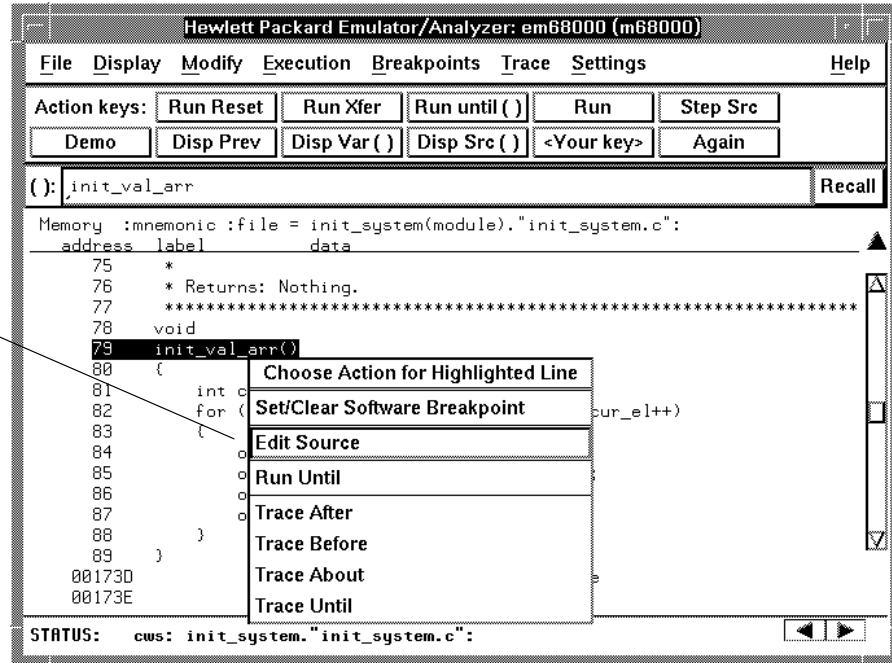
To edit a file that defines a symbol:

Choosing this menu item brings up a terminal window with an edit session open on the file where the highlighted symbol is defined.



To edit a file at a source line:

Choosing this menu item brings up a terminal window with an edit session open on the file where the highlighted source line exists.



To display the error log

- Choose **Display**→**Error Log**.
- Position the mouse pointer on the status line, press and hold the *select* mouse button, and then choose **Display Error Log** from the popup menu.

The last 100 error messages that have occurred during the emulation session are displayed.

To display the event log

- Choose **Display**→**Event Log**.
- Position the mouse pointer on the status line, press and hold the *select* mouse button, and then choose **Display Event Log** from the popup menu.

The last 100 events that have occurred during the emulation session are displayed.

The status of the emulator and analyzer are recorded in the event log, as well as the conditions that cause the status to change (for example, breakpoints and trace commands).

Examples

An event log display might look something like:

Event Log		
Time	Type	Message
11:00:11	OTHER	Loaded configuration file: /users/guest/demo/debug_env/hp64742/Config
11:00:27	SYMBOLS	Loaded symbol data base: /users/guest/demo/debug_env/hp64742/ecs.x
11:00:29	EVENTS	Loaded absolute file: /users/guest/demo/debug_env/hp64742/ecs.x
11:00:30	OTHER	XEnv_68k_except symbol seen; enter "help software_breakpoints"
11:35:02	PROC	M68000--Running user program
11:35:03	EVENTS	Software breakpoint: 0000fd20sp
11:35:04	PROC	M68000--Running in monitor
11:39:39	PROC	M68000--Emulation reset
11:39:43	PROC	M68000--Running in monitor
11:48:25	PROC	M68000--Running user program
11:50:03	TRACE	trace after update_system
11:50:05	TRACE	Emulation trace started
11:50:05	TRACE	Emulation trace complete
11:51:37	TRACE	trace before update_system

Notice that trace commands are listed in the event log display.

To display status

- Choose **Display**→**Status**

The emulator status screen tells you the present execution state of the emulator ("running user program" for example) as well as detailed trace status.

To copy information to a file or printer

- Choose **File**→**Copy**, select the type of information from the cascade menu, and use the dialog box to select the file or printer.

ASCII characters are copied to the file or printer.

If you copy information to an existing file, it will be appended to the file.

Refer to the following paragraphs for details about the different copy options.

Display ... Copies information currently in the display area. This option is useful for restricting the number of lines that are copied. Also, this option is useful for copying the contents of register classes other than BASIC.

Memory ... Copies the contents of a range of memory. The format is the same as specified in the last display memory command. For example, if you copy memory after displaying a range of memory in mnemonic format, the file would contain the mnemonic memory information. If there is no previous display memory command, the format used is a blocked hex byte format beginning at address zero.

Data Values ... Copies the contents of the defined data values last displayed. An error occurs if you try to copy data values to a file if you have not yet displayed data values.

Trace ... The most recently captured trace is copied to the file. The copied trace listing is formatted according to the current display mode.



Chapter 5: Using the Emulator/Analyzer Using System Commands

You can set the display mode with the **Settings**→**Source/Symbol Modes** or **Settings**→**Display Modes** pulldown menu items. See the "Changing the Interface Settings" section.

Registers ... Copies the current values of the BASIC register class to a file. To copy the contents of the other register classes, first display the registers in that class, and then use the **File**→**Copy**→**Display ...** command.

Breakpoints ... Copies the breakpoints list. If no breakpoints are present in the list, only the enable/disable status is copied.

Status ... Copies the emulator/analyzer status display.

Global Symbols ... Copies the global symbols. If symbols have not been loaded, this menu item is grayed-out and unresponsive.

Local Symbols () ... Copies the local symbols from the symbol scope named (by an enclosing symbol) in the entry buffer. If symbols have not been loaded, this menu item is grayed-out and unresponsive.

Pod Commands ... Copies the last 100 lines from the pod commands display.

Error Log ... Copies the last 100 lines from the error log display.

Event Log ... Copies the last 100 lines from event log display.

To open a terminal emulation window

- Choose **File**→**Term...**

This command opens a terminal window into the current working directory context.

Using Simulated I/O

Simulated I/O is a feature of the emulator/analyzer interface that lets you use the same keyboard and display that you use with the interface to provide input to programs and display program output.

To use simulated I/O, your programs must communicate with the simulated I/O control address and the buffer locations that follow it. (The Hewlett-Packard AxLS compilers, if your program uses I/O, automatically link with environment dependent routines that communicate with the simulated I/O control address and buffer.)

Also, before simulated I/O can work, the emulator must be configured to enable polling of the simulated I/O control address and to define the control address location.

This section shows you how to:

- Display the simulated I/O screen.
- Use simulated I/O keyboard input.

Refer to the *Simulated I/O User's Guide* for complete details on how simulated I/O works.

To display the simulated I/O screen

- Choose **Display**→**Simulated IO**.

Before you can display simulated I/O, polling for simulated I/O must be enabled in the emulator configuration.

Examples

```
Simulated I/O display                               Status messages disabled
display is open
51 56 78 87          h  H          t  T
52 57 79 89          h  H          t  T
52 58 80 91          h  H          t  T
53 59 79 90          h  H          t  T
53 60 79 89          h  H          t  T
54 61 78 88          h  H          t  T
54 62 78 87          h  H          t  T
55 63 77 86          h  H          t  T
55 64 77 85          h  H          t  T
55 63 76 84          h  H          t  T
54 62 76 83          h  H          t  T
54 61 75 82          h  H          t  T
53 53 75 75          H          t  T
49 52 71 74          h  H          t  T
49 51 71 73          h  H          t  T
48 50 70 72          h  H          t  T
```

A message tells you whether the display is open or closed. You can modify the configuration to enable status messages.

To use simulated I/O keyboard input

- To begin using simulated I/O input, choose **Settings**→**Simulated IO Keyboard**.
- To end simulated I/O and return to using the interface, use the **suspend** softkey.

The command line entry area is used for simulated input with the keyboard. Therefore, if the command line is turned off, choosing this menu item will turn command line display back on.

If you are planning to use even a modest amount of simulated I/O input during an emulation session, it might be a good idea to open another Emulator/Analyzer window to be used exclusively for simulated I/O input and output.

Using Basis Branch Analysis

Basis branch analysis (BBA) is provided by the HP Branch Validator product. This product is used to analyze the testing of your programs, create more complete test suites, and quantify your level of testing.

The HP Branch Validator records branches executed in a program and generates reports that provide information about program execution during testing. It uses a special C preprocessor to add statements that write to a data array when program branches are taken. After running the program in the emulator (using test input), you can store the BBA information to a file. Then, you can generate reports based on the stored information.

This section shows you how to:

- Store BBA data to a file.

Refer to the *HP Branch Validator (BBA) User's Guide* for complete details on the BBA product and how it works.

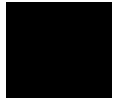
To store BBA data to a file

- Choose **File**→**Store**→**BBA Data** and use the selection dialog box to specify the file name.

The default file name "bbadump.data" can be selected from the dialog box.



6



Setting X Resources

Setting X Resources

The Graphical User Interface is an X Window System application which means it is a *client* in the X Window System client-server model.

The X server is a program that controls all access to input devices (typically a mouse and a keyboard) and all output devices (typically a display screen). It is an interface between application programs you run on your system and the system input and output devices.

An X *resource* controls an element of appearance or behavior in an X application. For example, in the graphical interface, one resource controls the text in action key pushbuttons as well as the action performed when the pushbutton is clicked.

By modifying resource settings, you can change the appearance or behavior of certain elements in the graphical interface.

When the graphical interface starts up, it reads resource specifications from a set of configuration files. Resources specifications in later files override those in earlier files. Files are read in the following order:

- 1 The application defaults file. For example, `/usr/lib/X11/app-defaults/HP64_Softkey` when the operating system is HP-UX or `/usr/openwin/lib/X11/app-defaults/HP64_Softkey` when the operating system is SunOS.
- 2 The `$XAPPLRESDIR/HP64_Softkey` file. (The `XAPPLRESDIR` environment variable defines a directory containing system-wide custom application defaults.)
- 3 The server's `RESOURCE_MANAGER` property. (The `xrdb` command loads user-defined resource specifications into the `RESOURCE_MANAGER` property.)

If no `RESOURCE_MANAGER` property exists, user defined resource settings are read from the `$HOME/.Xdefaults` file.

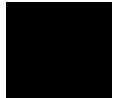
- 4 The file named by the XENVIRONMENT environment variable.
If the XENVIRONMENT variable is not set, the \$HOME/.Xdefaults-*host* file is read (typically contains resource specifications for a specific remote host).
- 5 Resource specifications included in the command line with the **-xrm** option.
- 6 System scheme files in directory /usr/hp64000/lib/X11/HP64_schemes.
- 7 System-wide custom scheme files located in directory \$XAPPLRESDIR/HP64_schemes.
- 8 User-defined scheme files located in directory \$HOME/.HP64_schemes (note the dot in the directory name).

Scheme files group resource specifications for different displays, computing environments, and languages.

This chapter shows you how to:

- Modify the Graphical User Interface resources.
- Use customized scheme files.
- Set up custom action keys.
- Set initial recall buffer values.
- Set up demos or tutorials.

Refer to the "X Resources and the Graphical Interface" chapter for more detailed information.



To modify the Graphical User Interface resources

You can customize the appearance of an X Windows application by modifying its X resources. The following tables describe some of the commonly modified application resources.

Application Resources for Schemes		
Resource	Values	Description
HP64_Softkey.platformScheme	HP-UX SunOS (custom)	Names the subdirectory for platform specific schemes. This resource should be set to the platform on which the X server is running (and displaying the Graphical User Interface) if it is different than the platform where the application is running.
HP64_Softkey.colorScheme	BW Color (custom)	Names the color scheme file.
HP64_Softkey.sizeScheme	Small Large (custom)	Names the size scheme file which defines the fonts and the spacing used.
HP64_Softkey.labelScheme	Label \$LANG (custom)	Names to use for labels and button text. The default uses the \$LANG environment variable if it is set and if a scheme file named Softkey.\$LANG exists in one of the directories searched for scheme files; otherwise, the default is Label.
HP64_Softkey.inputScheme	Input (custom)	Specifies mouse and keyboard operation.

Chapter 6: Setting X Resources
To modify the Graphical User Interface resources

Commonly Modified Application Resources		
Resource	Values	Description
HP64_Softkey.lines	24 (min. 18)	Specifies the number of lines in the main display area.
HP64_Softkey.columns	100 (min. 80)	Specifies the number of columns, in characters, in the main display area.
HP64_Softkey.enableCmdline	True False	Specifies whether the command line area is displayed when you initially enter the Graphical User Interface.
*editFile	(example) vi %s	Specifies the command used to edit files.
*editFileLine	(example) vi +%d %s	Specifies the command used to edit a file at a certain line number.
*<proc>*actionKeysSub.keyDefs	(paired list of strings)	Specifies the text that should appear on the action key push buttons and the commands that should be executed in the command line area when the action key is pushed. Refer to the "To set up custom action keys" section for more information.
*<proc>*dirSelectSub.entries	(list of strings)	Specifies the initial values that are placed in the File → Context → Directory popup recall buffer. Refer to the "To set initial recall buffer values" section for more information.
*<proc>*recallSub.entries	(list of strings)	Specifies the initial values that are placed in the entry buffer (labeled "(:)"). Refer to the "To set initial recall buffer values" section for more information.



Chapter 6: Setting X Resources

To modify the Graphical User Interface resources

The following steps show you how to modify the Graphical User Interface's X resources.

- 1 Copy part or all of the HP64_Softkey application defaults file to a temporary file.

The HP64_Softkey file contains the default definitions for the graphical interface application's X resources.

For example, on an HP 9000 computer you can use the following command to copy the complete HP64_Softkey file to HP64_Softkey.tmp (note that the HP64_Softkey file is several hundred lines long):

```
cp /usr/lib/X11/app-defaults/HP64_Softkey HP64_Softkey.tmp
```

NOTE: The HP64_Softkey application defaults file is re-created each time Graphical User Interface software is installed or updated. You can use the UNIX **diff** command to check for differences between the new HP64_Softkey application defaults file and the old application defaults file that is saved as /usr/hp64000/lib/X11/HP64_schemes/old/HP64_Softkey.

- 2 Modify the temporary file.

Modify the resource that defines the behavior or appearance that you wish to change.

For example, to change the number of lines in the main display area to 36:

```
vi HP64_Softkey.tmp
```

Search for the string "HP64_Softkey.lines". You should see lines similar to the following.

```
!-----  
! The lines and columns set the vertical and horizontal dimensions of the  
! main display area in characters, respectively. Minimum values are 18 lines  
! and 80 columns. These minimums are silently enforced.  
!  
! Note: The application cannot be resized by using the window manager.  
  
!HP64_Softkey.lines: 24  
!HP64_Softkey.columns: 85
```

Edit the line containing "HP64_Softkey.lines" so that it is uncommented and is set to the new value:

```
!-----  
! The lines and columns set the vertical and horizontal dimensions of the  
! main display area in characters, respectively. Minimum values are 18 lines
```

Chapter 6: Setting X Resources To modify the Graphical User Interface resources

! and 80 columns. These minimums are silently enforced.

!

! Note: The application cannot be resized by using the window manager.

```
HP64_Softkey.lines: 36  
!HP64_Softkey.columns: 85
```

Save your changes and exit the editor.

- 3 If the RESOURCE_MANAGER property exists (as is the case with HP VUE — if you're not sure, you can check by entering the **xrdb -query** command), use the **xrdb** command to add the resources to the RESOURCE_MANAGER property. For example:

```
xrdb -merge -nocpp HP64_Softkey.tmp
```

Otherwise, if the RESOURCE_MANAGER property does not exist, append the temporary file to your \$HOME/.Xdefaults file. For example:

```
cat HP64_Softkey.tmp >> $HOME/.Xdefaults
```

- 4 Remove the temporary file.
- 5 Start or restart the Graphical User Interface.

After you have completed the above steps, you must either start, or restart by exiting and starting again, the Graphical User Interface. Starting and exiting the Graphical User Interface is described in the "Starting and Exiting HP 64700 Interfaces" chapter.

To use customized scheme files

Scheme files are used to set platform specific resources that deal with color, fonts and sizes, mouse and keyboard operation, and labels and titles. You can create and use customized scheme files by following these steps.

- 1 Create the `$HOME/.HP64_schemes/<platform>` directory.

For example:

```
mkdir $HOME/.HP64_schemes  
mkdir $HOME/.HP64_schemes/HP-UX
```

- 2 Copy the scheme file to be modified to the `$HOME/.HP64_schemes/<platform>` directory.

Label scheme files are not platform specific; therefore, they should be placed in the `$HOME/.HP64_schemes` directory. All other scheme files should be placed in the `$HOME/.HP64_schemes/<platform>` directory.

For example:

```
cp /usr/hp64000/lib/X11/HP64_schemes/HP-UX/Softkey.Color  
$HOME/.HP64_schemes/HP-UX/Softkey.MyColor
```

Note that if your custom scheme file has the same name as the default scheme file, the load order requires resources in the custom file to explicitly override resources in the default file.

- 3 Modify the `$HOME/.HP64_schemes/<platform>/Softkey.<scheme>` file.

For example, you could modify the `"$HOME/.HP64_schemes/HP-UX/Softkey.MyColor"` file to change the defined foreground and background colors. Also, since the scheme file name is different than the default, you could comment out various resource settings to cause general foreground and background color definitions to apply to the Graphical User Interface. At least one resource must be defined in your color scheme file for it to be recognized.

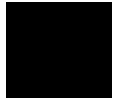
- 4 If your custom scheme file has a different name than the default, you must modify the scheme resource definitions.

The Graphical User Interface application defaults file contains resources that specify which scheme files are used. If your custom scheme files are named differently than the default scheme files, you must modify these resource settings so that your customized scheme files are used instead of the default scheme files.

For example, to use the "\$HOME/.HP64_schemes/HP-UX/Softkey.MyColor" color scheme file you would set the "HP64_Softkey.colorScheme" resource to "MyColor":

```
HP64_Softkey.colorScheme: MyColor
```

Refer to the previous "To customize Graphical User Interface resources" section for more detailed information on modifying resources.



To set up custom action keys

- Modify the "actionKeysSub.keyDefs" resource.

The "actionKeysSub.keyDefs" resource defines a list of paired strings. The first string defines the text that should appear on the action key pushbutton. The second string defines the command that should be sent to the command line area and executed when the action key is pushed.

A pair of parentheses (with no spaces, that is "()") can be used in the command definition to indicate that text from the entry buffer should replace the parentheses when the command is executed.

Action keys that use the entry buffer should always include the entry buffer symbol, "()", in the action key label as a visual cue to remind you to place information in the entry buffer before clicking the action key.

Shell commands can be executed by using an exclamation point prefix. A second exclamation point ends the command string and allows additional options on the command line.

Also, command files can be executed by placing the name of the file in the command definition.

Finally, an empty action ("") means to repeat the previous operation, whether it came from a pulldown, a dialog, a popup, or another action key.

Examples

To set up custom action keys when the graphical interface is used with the 68000 emulator, modify the "*m68000*actionKeysSub.keyDefs" resource:

```
*m68000*actionKeysSub.keyDefs: \  
"Make"      "cd /users/project2/68000; !make! in_browser" \  
"Load Pgm"  "load configuration config.EA; load program2" \  
"Run Pgm"   "run from reset" \  
"Trace after ( )" "trace after (); display trace" \  
"Step Source" "" "set source on; display memory mnemonic; step source" \  
"Again"
```

Refer to the previous "To modify Graphical User Interface resources" section for more detailed information on modifying resources.

To set initial recall buffer values

- Modify the "entries" resource for the particular recall buffer.

There are six popup recall buffers present in the Graphical User Interface. The resources for these popup recall buffers are listed in the following table.

The window manager resource "*transientDecoration" controls the borders around dialog box windows. The most natural setting for this resource is "title."

Popup Recall Buffer Resources		
Recall Popup	Resources	Description
File→Context→Directory ...	*dirSelect.textColumns *dirSelect.listVisibleItemCount *dirSelectSub.entries	The default number of text columns in the popup is 50.
File→Context→Symbols ...	*symSelect.textColumns *symSelect.listVisibleItemCount *symSelectSub.entries	The default number of visible lines in the popup is 12.
Trace→Trace Spec ...	*modtrace.textColumns *modtrace.listVisibleItemCount *modtraceSub.entries	The "entries" resource is defined as a list of strings (see the following example).
Entry Buffer ():	*recall.textColumns *recall.listVisibleItemCount *recallSub.entries	Up to 40 unique values are saved in each of the recall buffers (as specified by the resource settings
Command Line command recall	*recallCmd.textColumns *recallCmd.listVisibleItemCount *recallCmdSub.entries	"*XcRecall.maxDepth: 40" and "*XcRecall.onlyUnique: True").
Command Line pod/simio recall	*recallKbd.textColumns *recallKbd.listVisibleItemCount *recallKbdSub.entries	

Chapter 6: Setting X Resources

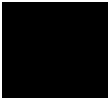
To set initial recall buffer values

Examples

To set the initial values for the directory selection dialog box when the Graphical User Interface is used with 68000 emulators, modify the "*m68000*dirSelectSub.entries" resource:

```
*m68000*dirSelectSub.entries: \  
"$HOME"\  
".."\  
"/users/project1"\  
"/users/project2/68000"
```

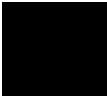
Refer to the previous "To modify the Graphical User Interface resources" section for more detailed information on modifying resources.



To set up demos or tutorials

You can add demos or tutorials to the Graphical User Interface by modifying the resources described in the following tables.

Demo Related Component Resources		
Resource	Value	Description
*enableDemo	False True	Specifies whether Help→Demo appears in the pulldown menu.
*demoPopupSub.indexFile	./Xdemo/Index-topics	Specifies the file containing the list of topic and file pairs.
*demoPopup.textColumns	30	Specifies the width, in characters, of the demo topic list popup.
*demoPopup.listVisibleItemCount	10	Specifies the length, in lines, of the demo topic list popup.
*demoTopic	About demos	Specifies the default topic in the demo popup selection buffer.



Chapter 6: Setting X Resources
To set up demos or tutorials

Tutorial Related Component Resources		
Resource	Value	Description
*enableTutorial	False True	Specifies whether Help → Tutorial appears in the pulldown menu.
*tutorialPopupSub.indexFile	./Xtutorial/Index-topics	Specifies the file containing the list of topic and file pairs.
*tutorialPopup.textColumns	30	Specifies the width, in characters, of the tutorial topic list popup.
*tutorialPopup.listVisibleItemCount	10	Specifies the length, in lines, of the tutorial topic list popup.
*tutorialTopic	About tutorials	Specifies the default topic in the tutorial popup selection buffer.

The mechanism for providing demos and tutorials in the graphical interface is identical. The following steps show you how to set up demos or tutorials in the Graphical User Interface.

- 1 Create the demo or tutorial topic files and the associated command files.

Topic files are simply ASCII text files. You can use "\I" to produce inverse video in the text, "\U" to produce underlining in the text, and "\N" to restore normal text.

Command files are executed when the "Press to perform demo (or tutorial)" button (in the topic popup dialog) is pushed. A command file must have the same name as the topic file with ".cmd" appended. Also, a command file must be in the same directory as the associated topic file.

2 Create the demo or tutorial index file.

Each line in the index file contains first a quoted string that is the name of the topic which appears in the index popup and second the name of the file that is raised when the topic is selected. For example:

```
"About demos" /users/guest/gui_demos/general  
"Loading programs" /users/guest/gui_demos/loadprog  
"Running programs" /users/guest/gui_demos/runprog
```

You can use absolute paths (for example, /users/guest/topic1), paths relative to the directory in which the interface was started (for example, mydir/topic2), or paths relative to the product directory (for example, ./Xdemo/general where the product directory is something like /usr/hp64000/inst/emul/64751A).

3 Set the "*enableDemo" or "*enableTutorial" resource to "True".

4 Define the demo index file by setting the "*demoPopupSub.indexFile" or "*tutorialPopupSub.indexFile" resource.

For example:

```
*demoPopupSub.indexFile: /users/guest/gui_demos/index
```

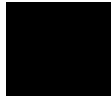
You can use absolute paths (for example, /users/guest/Index), paths relative to the directory in which the interface was started (for example, mydir/indexfile), or paths relative to the product directory (for example, ./Xdemo/Index-topics where the product directory is something like /usr/hp64000/inst/emul/64751A).

5 If you wish to define a default topic to be selected, set the "*demoTopic" or "*tutorialTopic" resource to the topic string.

For example:

```
*demoTopic: "About demos"
```

Refer to the previous "To customize Graphical User Interface resources" section for more detailed information on modifying resources.

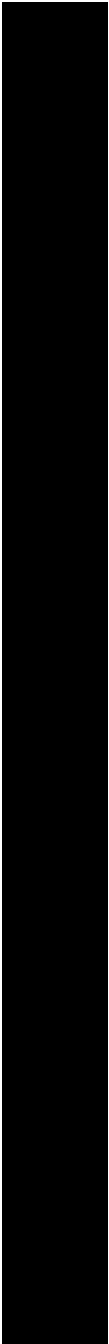




Part 3

Concept Guide

Part 3



7

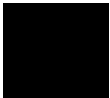


X Resources and the Graphical User Interface

X Resources and the Graphical User Interface

This chapter contains more detailed information about X resources and scheme files that control the appearance and operation of the Graphical User Interface. This chapter:

- Describes the X Window concepts surrounding resource specification.
- Describes the Graphical User Interface's implementation of scheme files.



X Resource Specifications

An X resource specification is a resource name and a value. The resource name identifies the element whose appearance or behavior is to be defined, and the value specifies how the element should look or behave. For example, consider the following resource specification:

```
Application.form.row.done.background: red
```

The resource name is "Application.form.row.done.background:" and the value is "red".

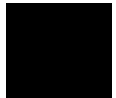
Resource Names Follow Widget Hierarchy

A *widget* is an OSF/Motif graphic device from which X applications are built. For example, pushbuttons and menu bars are Motif widgets. Applications are built using a hierarchy of widgets, and the application's X resource names follow this hierarchy. For example:

```
Application.form.row.done.background: red
```

In the resource name above, the top-level widget is named after the application. One of the top-level widget's children is a form widget, one of the form widget's children is a row-column manager widget, and one of the row-column manager widget's children is a pushbutton widget. Resource names show a path in the widget hierarchy.

Each widget in the hierarchy is a member of a widget class, and the particular instance of the widget is named by the application programmer.



Class Names or Instance Names Can Be Used

When specifying resource names, you can use either instance names or class names. For example, a "Done" pushbutton may have an instance name of "done" and a class name of "XmPushButton". To set the background color for a hypothetical "Done" pushbutton, you can use:

```
Application.form.row.done.background: red
```

Or, you can use:

```
Application.form.row.XmPushButton.background: red
```

Applications also have class and instance names. For example, an application may have an instance name of "applic1" and a class name of "Application". To set the background color for a hypothetical "Done" pushbutton only in the "applic1" application, you can use:

```
applic1.form.row.done.background: red
```

Note that instance names are more specific than class names. That is, class names may apply to many instances of the widget.

The class and instance names for the widgets in the Graphical User Interface can be displayed by choosing **Help**→**X Resource Names** and clicking on the "All names" button.

Wildcards Can Be Used

A wildcard may be used to match a resource specification to many different widgets at once. For example, to set the background color of all pushbuttons, you can use:

```
Application*XmPushButton.background: red
```

Note that resource names with wildcards are more general than those without wildcards.

Specific Names Override General Names

A more specific resource specification will override a more general one when both apply to a particular widget or application.

The names for the application and the main window widget in HP64_Softkey applications have been chosen so that you may specify custom resource values that apply in particular situations:

- 1 Apply to ALL HP64_Softkey applications:

HP64_Softkey*<resource>: <value>

- 2 Apply to specific types of HP64_Softkey applications:

emul*<resource>: <value> (for the emulator)

perf*<resource>: <value> (for the performance analyzer)

- 3 Apply to all HP64_Softkey applications, but only when they are connected to a particular type of microprocessor:

m68000<resource>: <value> (for the 68000)

m68020<resource>: <value> (for the 68020)

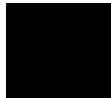
- 4 Apply to a specific HP64_Softkey application connected to a specific processor:

perf.m68000*<resource>: <value> (for the 68000 perf. analyzer)

emul.m68020*<resource>: <value> (for the 68020 emulator)

If all four examples above are used for a particular resource, #3 will override #2 for all applications connected to a 68000 emulator, and #4 will override #2, but only for the specifically mentioned type of microprocessor.

When modifying resources, your resource paths must either match, or be more specific than, those found in the application defaults file.



How X Resource Specifications are Loaded

When the Graphical User Interface starts up, it loads resource specifications from a set of configuration files located in system directories as well as user-specific locations.

Application Default Resource Specifications

Default resource specifications for an application are placed in a system directory:

HP-UX /usr/lib/X11/app-defaults

SunOS /usr/openwin/lib/X11/app-defaults

The name of the Graphical User Interface application defaults file is `HP64_Softkey` (same as the application class name). This file is well-commented and contains information about each of the X resources you can modify. You can easily view this file by choosing **Help**→**Topic** and selecting the "X Resources: App Default File" topic. Do not modify the application defaults file; any changes to this file will affect the appearance and behavior of the application for all users.

User-Defined Resource Specifications

User-defined resources (for any X application) are located in the X server's `RESOURCE_MANAGER` property or in the user's `$HOME/.Xdefaults` file.

Load Order

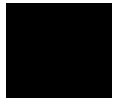
Resource specifications are loaded from the following places in the following order:

- 1 The application defaults file. For example, `/usr/lib/X11/app-defaults/HP64_Softkey` when the operating system is HP-UX or `/usr/openwin/lib/X11/app-defaults/HP64_Softkey` when the operating system is SunOS.
- 2 The `$XAPPLRESDIR/HP64_Softkey` file. (The `XAPPLRESDIR` environment variable defines a directory containing system-wide custom application defaults.)
- 3 The server's `RESOURCE_MANAGER` property. (The `xrdb` command loads user-defined resource specifications into the `RESOURCE_MANAGER` property.)

If no `RESOURCE_MANAGER` property exists, user defined resource settings are read from the `$HOME/.Xdefaults` file.

- 4 The file named by the `XENVIRONMENT` environment variable.
If the `XENVIRONMENT` variable is not set, the `$HOME/.Xdefaults-host` file is read (typically contains resource specifications for a specific remote host).
- 5 Resource specifications included in the command line with the `-xrm` option.

When specifications with identical resource names appear in different places, the latter specification overrides the former.



Scheme Files

Several of the Graphical User Interface's X resources identify *scheme files* that contain additional X resource specifications. Scheme files group resource specifications for different displays, computing environments, and languages.

Resources for Graphical User Interface Schemes

There are five X resources that identify scheme files:

HP64_Softkey.labelScheme:

Names the scheme file to use for labels and button text. Values can be: Label, \$LANG, or a custom scheme file name. The default uses the \$LANG environment variable if it is set and if a scheme file named Softkey.\$LANG exists in one of the directories searched for scheme files; otherwise, the default is Label.

HP64_Softkey.platformScheme:

Names the subdirectory for the platform specific color, size, and input scheme files. This resource should be set to the platform on which the X server is running (and displaying the Graphical User Interface) if it is different than the platform where the application is running. Values can be: HP-UX, SunOS, pc-xview, or a custom platform scheme directory name.

HP64_Softkey.colorScheme:

Names the color scheme file. Values can be: Color, BW, or a custom scheme file name.

HP64_Softkey.sizeScheme:

Names the size scheme file which defines the fonts and the spacing used. Values can be: Large, Small, or a custom scheme file name.

HP64_Softkey.inputScheme:

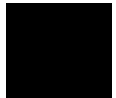
Names the input scheme file which specifies mouse and keyboard operation. Values can be: Input, or a custom scheme file name.

The actual scheme file names take the form: "Softkey.<value>".

Scheme File Names

There are six scheme files provided with the Graphical User Interface. Their names and brief descriptions of the resources they contain follow.

Softkey.Label	Defines the labels for the fixed text in the interface. Such things as menu item labels and similar text are in this file. If the \$LANG environment variable is set, the scheme file "Softkey.\$LANG" is loaded if it exists; otherwise, the file "Softkey.Label" is loaded.
Softkey.BW	Defines the <i>color scheme</i> for black and white displays. This file is chosen if the display cannot produce at least 16 colors.
Softkey.Color	Defines the <i>color scheme</i> for color displays. This file is chosen if the display can produce 16 or more colors.
Softkey.Large	Defines the <i>size scheme</i> (that is, the window dimensions and fonts) for high resolution displays (1000 pixels or more vertically).
Softkey.Small	Defines the <i>size scheme</i> (that is, the window dimensions and fonts) for low resolution displays (less than 1000 pixels vertically).
Softkey.Input	Defines the <i>input scheme</i> (that is, the button and key bindings for the mouse and keyboard).



Load Order for Scheme Files

Scheme files are searched for in the following directories and in the following order:

- 1 System scheme files in directory /usr/hp64000/lib/X11/HP64_schemes.
- 2 System-wide custom scheme files located in directory \$XAPPLRESDIR/HP64_schemes.
- 3 User-defined scheme files located in directory \$HOME/.HP64_schemes (note the dot in the directory name).

Custom Scheme Files

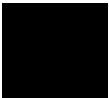
You can modify scheme files by copying them to the directory for user-defined schemes and changing the resource specifications in the file. For example, if you wish to modify the color scheme, and your platform is HP-UX, you can copy the /usr/hp64000/lib/X11/HP64_schemes/HP-UX/Softkey.Color file to \$HOME/.HP64_schemes/HP-UX/Softkey.Color and modify its resource specifications.

You can create custom scheme files by modifying the X resource for the particular scheme and by placing the custom scheme file in the directory for user-defined schemes. For example, if the following resource specifications are made:

```
HP64_Softkey.platformScheme: HP-UX  
HP64_Softkey.colorScheme: MyColor
```

The custom scheme file would be:

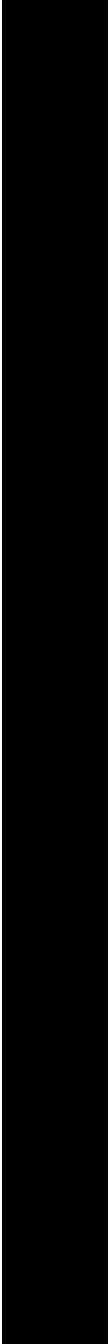
```
$HOME/.HP64_schemes/HP-UX/Softkey.MyColor
```



Part 4

Installation Guide

Part 4



8



Installation

Installation at a Glance

Before you can use the Graphical User Interface, you may need to install emulator hardware, and you have to install the interface software. You also need to verify the installation of the interface software and understand how to start the Graphical User Interface for the first time.

This chapter is not intended to be a complete installation guide for all of the just-mentioned tasks. This chapter concentrates on information, not found in other places, that is necessary for the installation or operation of the interface.

Installation Overview for HP 9000 Hosted Systems

Users of HP 9000 hosted systems should follow the instructions in the section titled "Installation for HP 9000 Hosted Systems". Briefly, those instructions tell you to do the following:

- 1 If necessary, install emulator, analyzer, or memory cards in the HP 64700 Series Cardcage according to the instructions found in the *HP 64700 Series Installation/Service Guide*.
- 2 Connect the emulator to your system and configure the emulator to communicate via the LAN (or RS-422 or RS-232) with the HP 9000 according to instructions also found in the *HP 64700 Series Installation/Service Guide*.
- 3 Install the Graphical User Interface and supporting HP 64700 Series software according to instructions found in this chapter. Alternatively, you may install the Softkey Interface and choose not to install the Graphical User Interface.
- 4 Verify the software installation according to instructions given in the "Installation for HP 9000 Hosted Systems" section of this chapter.
- 5 Start the interface according to instructions given in the "Installation for HP 9000 Hosted Systems" section of this chapter.
- 6 Exit the interface and go on to other chapters in this book.

Minimum HP 9000 Hardware and System Requirements

The following is a set of minimum hardware and system recommendations for operation of the Graphical User Interface on HP 9000 Series 300/400 and Series 700 workstations.

HP-UX For Series 9000/300 and Series 9000/400 workstations, the minimum supported version of the operating system is 7.03 or later. For Series 9000/700 workstations, the minimum supported version of the operating system is version 8.01.

64700 Operating Environment The Graphical User Interface requires version A.04.10 of the 64700 Operating Environment. (The Graphical User Interface version is A.04.00).

Motif/OSF For Series 9000/700 workstations, you must also have the Motif 1.1 dynamic link libraries installed. They are installed by default, so you do not have to install them specifically for this product, but you should consult your HP-UX documentation for confirmation and more information.

Hardware and Memory Any workstation used with the Graphical User Interface should have a minimum of 16 megabytes of memory (32 megabytes or more is recommended). Series 300 workstations should have a minimum performance equivalent to that of a HP 9000/350. A color display is also highly recommended.

From here, you should proceed to the section titled "Installation for HP 9000 Hosted Systems" for instructions on how to install, verify, and start the Graphical User Interface on HP 9000 systems.

Installation Overview for Sun SPARCsystems

Users of Sun SPARCsystems should follow the instructions in the section titled "Installation for Sun SPARCsystems". Briefly, those instructions tell you to do the following:

- 1 If necessary, install emulator, analyzer, or memory cards in the HP 64700 Series Cardcage according to the instructions found in the *HP 64700 Series Installation/Service Guide*.
- 2 Connect the emulator to your system and configure the emulator to communicate via the LAN with the hosted workstation according to instructions also found in the *HP 64700 Series Installation/Service Guide*.
- 3 Install the Graphical User Interface and supporting HP 64700 Series software according to instructions found in this chapter. Alternatively, you may install the Softkey Interface and choose not to install the Graphical User Interface.

Chapter 8: Installation

Installation at a Glance

- 4 Verify the software installation according to instructions given in the "Installation for Sun SPARCsystems" section of this chapter.
- 5 Start the interface according to instructions given in the "Installation for Sun SPARCsystems" section of this chapter.
- 6 Exit the interface and go on to other chapters in this book.

Minimum Sun SPARCsystem Hardware and System Requirements

The following is a set of minimum hardware and system recommendations for operation of the Graphical User Interface on Sun SPARCsystem workstations.

SunOS The Graphical User Interface software is designed to run on a Sun SPARCsystem with SunOS version 4.1 or 4.1.1 or greater (Solaris 1.X). At the time this manual was printed, some Graphical User Interface software would not run on Solaris 2.3 and above; ask your Hewlett-Packard Sales Office if there are plans to support your user interface software on Solaris 2.3. The tape uses the QIC-24 data format.

64700 Operating Environment The Graphical User Interface requires version A.04.10 or greater of the 64700 Operating Environment. (The minimum Graphical User Interface version required is A.04.00.)

Hardware and Memory Any workstation used with the Graphical User Interface should have a minimum of 16 megabytes of memory (32 megabytes or more is recommended). A color display is also highly recommended.

From here, you should proceed to the section titled "Installation for Sun SPARCsystems" for instructions on how to install, verify, and start the Graphical User Interface on SPARCsystem workstations.

Installation for HP 9000 Hosted Systems

Follow these instructions to install the Graphical User Interface on HP 9000 workstations. You can also follow these instructions through Step 4 to find out how not to install the Graphical User Interface if you want to use just the Softkey Interface.

Step 1. Install the hardware in the HP 64700 Series Cardcage

Turn to the *HP 64700 Series Installation/Service Guide* and follow the instructions for installing emulator, memory, or analyzer cards in the HP 64700 Series Cardcage. It may be that you already have installed the cards in the cardcage or your cardcage came with cards already installed.

If you have already installed the hardware and software and connected the emulator to your host system, skip to Step 5 to verify the software installation. Otherwise, continue with Step 2 of these instructions.

Step 2. Configure the emulator for the communication channel

Turn to the *HP 64700 Series Installation/Service Guide* and follow the instructions for configuring the emulator to communicate via LAN, RS-422, or RS-232. (RS-422 and RS-232 are only supported on HP 9000 Series 300/400 machines.)

When you have configured the emulator to communicate via the channel you have chosen, continue with Step 3 of these instructions.

Step 3. Connect the emulator to your system

Turn to the *HP 64700 Series Installation/Service Guide* and follow the instructions for connecting the emulator to your system. You can connect the emulator via LAN, RS-422, or RS-232.

When you have connected the emulator to your host system, continue with Step 4 of these instructions.

Step 4. Install the software

The tape that contains the Graphical User Interface software may contain several products. Usually, you will want to install all of the products on the tape. However, to save disk space, or for other reasons, you can choose to install selected filesets.

If you plan on using the Softkey Interface instead of the Graphical User Interface, you can save about 3.5 megabytes of disk space by not installing the XUI suffixed filesets in the "64700 Operating Environment" and "<processor-type> Emulation Tools" partitions. (Also, if you choose not to install the Graphical User Interface, you will not have to use a special command line option to start the Softkey Interface.)

Refer to the information on updating HP-UX in your HP-UX documentation for instructions on viewing partitions and filesets and marking filesets that should not be loaded.

The following sub-steps assume that you want to install all products on the tape.

- 1 Become the root user on the system you want to update.
- 2 Make sure the tape's write-protect screw points to SAFE.

- 3 Put the product media into the tape drive that will be the *source device* for the update process.
- 4 Confirm that the tape drive BUSY and PROTECT lights are on. If the PROTECT light is not on, remove the tape and confirm the position of the write-protect screw. If the BUSY light is not on, check that the tape is installed correctly in the drive and that the drive is operating correctly.
- 5 When the BUSY light goes off and stays off, start the update program by entering
/etc/update
at the HP-UX prompt.
- 6 When the HP-UX update utility main screen appears, confirm that the source and destination devices are correct for your system. Refer to the information on updating HP-UX in your HP-UX documentation if you need to modify these values.
- 7 Select "Load Everything from Source Media" when your source and destination directories are correct.
- 8 To begin the update, press the softkey <Select Item>. At the next menu, press the softkey <Select Item> again. Answer the last prompt with
y
It takes about 20 minutes to read the tape.
- 9 When the installation is complete, read /tmp/update.log to see the results of the update.



Step 5. Verify the software installation

A number of new filesets were installed on your system during the software installation process. This and following steps assume that you chose to load the Graphical User Interface filesets.

You can use this step to further verify that the filesets necessary to successfully start the Graphical User Interface have been loaded and that customize scripts have run correctly. Of course, the update process gives you mechanisms for verifying installation, but these checks can help to double-check the installation process.

- 1 Verify the existence of the **HP64_Softkey** file in the **/usr/lib/X11/app-defaults** subdirectory by entering
ls /usr/lib/X11/app-defaults/HP64_Softkey at the HP-UX prompt.

Finding this file verifies that you loaded the correct fileset and also verifies that the customize scripts executed because this file is created from other files during the customize process.

- 2 Examine **/usr/lib/X11/app-defaults/HP64_Softkey** near the end of the file to confirm that there are resources specific to your emulator.

Near the end of the file, there will be resource strings that contain references to specific emulators. For example, if you installed the Graphical User Interface for the 68000 emulator, resource name strings will have **m68000** embedded in them.

After you have verified the software installation, you must start the X server and an X window manager (if you are not currently running an X server). If you plan to run the Motif Window Manager (mwm), or similar window manager, continue with Step 6a of these instructions. If you plan to run HP VUE, skip to Step 6b of these instructions.

Step 6a. Start the X server and the Motif Window Manager (mwm)

If you are not already running the X server and a window manager, do so now. The X server is required to use the Graphical User Interface because it is an X Windows application. A window manager is not required to execute the interface, but, as a practical matter, you must use some sort of window manager with the X server.

- Start the X server by entering **x11start** at the HP-UX prompt.

Consult the X Window documentation supplied with the HP-UX operating system documentation if you do not know about using X Windows and the X server.

After starting the X server and Motif Window Manager, continue with step 7 of these instructions.

Step 6b. Start HP VUE

If you are running the X server under HP VUE and have not started HP VUE, do so now.

HP VUE is a window manager for the X Window system. The X server is executing underneath HP VUE. Unlike the Motif Window Manager, HP VUE provides a login shell and is your default interface to the HP 9000 workstation.

Step 7. Set the necessary environment variables

The DISPLAY environment variable must be set before the Graphical User Interface will start. Also, you should modify the PATH environment variable to include the "/usr/hp64000/bin" directory, and, if you have installed software in a directory other than "/", you need to set the HP64000 environment variable.

The following instructions show you how to set these variables at the UNIX prompt. Modify your ".profile" or ".login" file if you wish these environment variables to be set when you log in. The following instructions also assume that you're using "sh" or "ksh"; if you're using "csh", environment variables are set using the "setenv <VARIABLE> <value>" command.

- 1 Set the DISPLAY environment variable by entering

```
DISPLAY=<hostname>:<server_number>.<screen_number>  
export DISPLAY
```

For example:

```
DISPLAY=myhost:0.0; export DISPLAY
```

Consult the X Window documentation supplied with the UNIX system documentation for an explanation of the DISPLAY environment variable.

- 2 Set the HP64000 environment variable.

For example, if you installed the HP 64000 software relative to the root directory, "/", you would enter

```
HP64000=/usr/hp64000; export HP64000
```

If you installed the software relative to a directory other than the root directory, it is strongly recommended that you use a symbolic link to make the software appear to be under /usr/hp64000. For example, if you installed the software relative to directory /users/team, you would enter

```
ln -s /users/team/usr/hp64000 /usr/hp64000
```

If you do not wish to establish a symbolic link, you can set the HP64000 variable to the full path that contains the HP 64000 software. Again, if you installed relative to /users/team, you would enter

```
HP64000=/users/team/usr/hp64000; export HP64000
```

- 3 Set the PATH environment variable to include the **usr/hp64000/bin** directory by entering

```
PATH=$PATH:$HP64000/bin; export PATH
```

Including **usr/hp64000/bin** in your PATH relieves you from prefixing HP 64700 executables with the directory path.

- 4 Set the MANPATH environment variable to include the **usr/hp64000/man** and **usr/hp64000/contrib/man** directories by entering

```
MANPATH=$MANPATH:$HP64000/man:$HP64000/contrib/man  
export MANPATH
```

Including these directories in your MANPATH variable lets you access the on-line "man" page information included with the software.



Step 8. Determine the logical name of your emulator

The *logical name* of an emulator is a label associated with a set of communications parameters in the **\$HP64000/etc/64700tab.net** file. The 64700tab.net file is placed in the directory as part of the installation process.

- 1 Display the 64700tab.net file by entering **more /usr/hp64700/etc/64700tab.net** at the HP-UX prompt.
- 2 Page through the file until you find the emulator you are going to use.

This step will require some matching of information to an emulator, but it should not be difficult to determine which emulator you want to address.

Examples

A typical entry for an 68000 emulator connected to the LAN would appear as follows:

```
#-----  
# Channel| Logical | Processor | Remainder of Information for the Channel  
# Type | Name | Type | (IP address for LAN connections)  
#-----  
lan: em68000 m68000 21.17.9.143
```

A typical entry for an 68000 emulator connected to an RS-422 port would appear as follows:

```
#-----  
# | | | | |Xpar|Parity|Flow|Stop|Char  
# Channel| Logical | Processor | Host | Physical |Mode| | |Bits|Size  
# Type | Name | Type | Name | Device | | |XON| |  
# | | | | |OFF | NONE |RTS | 2 | 8  
#-----  
serial: em68000 m68000 myhost /dev/emcom23 OFF NONE RTS 2 8
```


Step 9. Start the interface with the **emul700** command

- 1 Apply power to the emulator you wish to access after making sure the emulator is connected to the LAN or to your host system.

On the HP 64700 Series Emulator, the power switch is located on the front panel near the bottom edge. Push the switch in to turn power on to the emulator.

- 2 Wait a few seconds to allow the emulator to complete its startup initialization.
- 3 Choose a terminal window from which to start the Graphical User Interface.
- 4 Start the Graphical User Interface by entering the **emul700** command and giving the logical name of the emulator as an argument to the command, as in

```
$HP64000/bin/emul700 <logical_name> &
```

or

```
emul700 <logical name> &
```

if **\$HP64000/bin** is in your path.

If you are running the X server, if the Graphical User Interface is installed, and if your DISPLAY environment variable is set, the **emul700** command will start the Graphical User Interface. Otherwise, **emul700** starts the Softkey Interface.

You should include an ampersand ("&") with the command to start the Graphical User Interface as a background process. Doing so frees the terminal window where you started the interface so that the window may still be used.

- 5 Optionally start additional Graphical User Interface windows into the same emulation session by repeating the previous step.

Chapter 8: Installation
Installation for HP 9000 Hosted Systems

You can also choose to use the Softkey Interface under X Windows, but you must include a command line argument to **emul700** to override the default Graphical User Interface. Start the Softkey Interface by entering

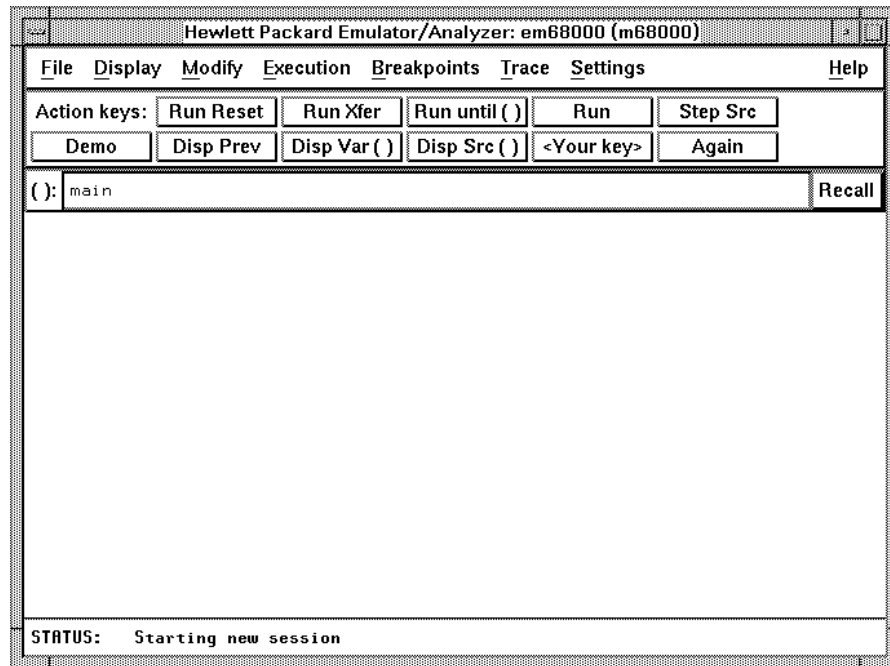
emul700 -u skemul <logical name>

Example

Suppose you have discovered that the logical name for a 68000 emulator connected to the LAN is "em68000". To start the Graphical User Interface and begin communicating with that emulator, enter (assuming your \$PATH includes **\$HP64000/bin**)

emul700 em68000

After a few seconds, the Graphical User Interface Emulator/Analyzer window should appear on your screen. The window will be similar to the following:



Step 10. Exit the Graphical User Interface

- 1 Position the mouse pointer over the pulldown menu named "File" on the menu bar at the top of the interface screen.
- 2 Press and hold the *command select* mouse button until the File menu appears.
- 3 While continuing to hold the mouse button down, move the mouse pointer down the menu to the "Exit" menu item.
- 4 Display the Exit cascade menu by moving the mouse pointer to the right edge of the Exit menu choice. There is an arrow on the right edge of the menu item.
- 5 Choose "Released" from the cascade menu.

The interface will terminate and release the emulator for use by others.



Installation for Sun SPARCsystems

Follow these instructions to install the Graphical User Interface on Sun SPARCsystem workstations. You can also follow these instructions through Step 4 to find out how to prevent installation of the Graphical User Interface if you only plan to use the Softkey Interface.

Step 1. Install the hardware in the HP 64700 Series Cardcage

Turn to the *HP 64700 Series Installation/Service Guide* and follow the instructions for installing emulator, memory, or analyzer cards in the HP 64700 Series Cardcage. It may be that you already have installed the cards in the cardcage or your cardcage came with cards already installed.

If you have already installed the hardware and software and connected the emulator to your host system, skip to Step 5 to verify the software installation. Otherwise, continue with Step 2 of these instructions.

Step 2. Configure the emulator for the communication channel

Turn to the *HP 64700 Series Installation/Service Guide* and follow the instructions for configuring the emulator to communicate via LAN. (RS-422 and RS-232 are only supported on HP 9000 Series 300/400 machines.)

When you have configured the emulator to communicate via LAN, continue with Step 3 of these instructions.

Step 3. Connect the emulator to your system

Turn to the *HP 64700 Series Installation/Service Guide* and follow the instructions for connecting the emulator to your system. You can connect the emulator via LAN.

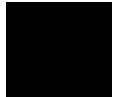
When you have connected the emulator to your host system, continue with Step 4 of these instructions.

Step 4. Install the software

The tape that contains the Graphical User Interface software may contain several products. Usually, you will want to install all of the products on the tape. However, to save disk space, or for other reasons, you can choose to install selected filesets.

If you plan on using the Softkey Interface instead of the Graphical User Interface, you can save about 3.5 megabytes of disk space by not installing the XUI suffixed filesets. (Also, if you choose not to install the Graphical User Interface, you will not have to use a special command line option to start the Softkey Interface.)

Refer to the *Software Installation Notice* for software installation instructions. After you are done installing the software, return here.



Step 5. Start the X server and OpenWindows

If you are not already running the X server, do so now. The X server is required to run the Graphical User Interface because it is an X application.

- Start the X server by entering `/usr/openwin/bin/openwin` at the UNIX prompt.

Consult the OpenWindows documentation if you do not know about using OpenWindows and the X server.

Step 6. Set the necessary environment variables

The DISPLAY environment variable must be set before the Graphical User Interface will start. Also, you should modify the PATH environment variable to include the "usr/hp64000/bin" directory, and, if you have installed software in a directory other than "/", you need to set the HP64000 environment variable.

The following instructions show you how to set these variables at the UNIX prompt. Modify your ".profile" or ".login" file if you wish these environment variables to be set when you log in. The following instructions also assume that you're using "csh"; if you're using "sh", environment variables are set in the "<VARIABLE>=<value>; export <VARIABLE>" form.

- 1 The DISPLAY environment variable is usually set by the **openwin** startup script. Check to see that DISPLAY is set by entering

```
echo $DISPLAY
```

If DISPLAY is not set, you can set it by entering

```
setenv DISPLAY=<hostname>:<server_number>.<screen_number>
```

For example:

```
setenv DISPLAY=myhost:0.0
```

Consult the OpenWindows documentation for an explanation of the DISPLAY environment variable.

2 Set the HP64000 environment variable.

For example, if you installed the HP 64000 software relative to the root directory, "/", you would enter

```
setenv HP64000 /usr/hp64000
```

If you installed the software relative to a directory other than the root directory, it is strongly recommended that you use a symbolic link to make the software appear to be under /usr/hp64000. For example, if you installed the software relative to directory /users/team, you would enter

```
ln -s /users/team/usr/hp64000 /usr/hp64000
```

If you do not wish to establish a symbolic link, you can set the HP64000 variable to the full path that contains the HP 64000 software; also set the LD_LIBRARY_PATH variable to the directory containing run-time libraries used by the HP 64000 products. Again, if you installed relative to /users/team, you would enter

```
setenv HP64000 /users/team/usr/hp64000  
setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:${HP64000}/lib
```

3 Set the PATH environment variable to include the **usr/hp64000/bin** directory by entering

```
setenv PATH ${PATH}:${HP64000}/bin
```

Including **usr/hp64000/bin** in your PATH relieves you from prefixing HP 64700 executables with the directory path.

Chapter 8: Installation
Installation for Sun SPARCsystems

- 4 Set the MANPATH environment variable to include the **usr/hp64000/man** and **usr/hp64000/contrib/man** directories by entering

```
setenv MANPATH ${MANPATH}:${HP64000}/man
setenv MANPATH ${MANPATH}:${HP64000}/contrib/man
```

Including these directories in your MANPATH variable lets you access the on-line "man" page information included with the software.

- 5 If the Graphical User Interface is to run on a SPARCsystem computer that is not running OpenWindows, include the **/usr/openwin/lib** directory in LD_LIBRARY_PATH.

```
setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:/usr/openwin/lib
```

Step 7. Verify the software installation

A number of product filesets were installed on your system during the software installation process. Due to the complexity of installing on NFS mounted file systems, a script that verifies and customizes these products was also installed. This stand alone script may be run at any time to verify that all files required by the products are in place in the file system. If required files are not found, this script will attempt to symbolically link them from the \$HP64000 install directory to their proper locations.

- Run the script **\$HP64000/bin/envinstall**.

Step 8. Map your function keys

If you are using the Softkey Interface, map your function keys by following the steps below.

- 1 Copy the function key definitions by typing:

```
cp $HP64000/etc/ttyswrc ~/.ttyswrc
```

This creates key mappings in the .ttyswrc file in your \$HOME directory.

- 2 Remove or comment out the following line from your .xinitrc file:

```
xmodmap -e 'keysym F1 = Help'
```

If any of the other keys F1-F8 are remapped using xmodmap, comment out those lines also.

- 3 Add the following to your .profile or .login file:

```
stty erase ^H  
setenv KEYMAP sun
```

The erase character needs to be set to backspace so that the Delete key can be used for "delete character."

If you want to continue using the F1 key for HELP, you can use F2-F9 for the Softkey Interface. All you have to do is set the KEYMAP variable. If you use OpenWindows, type:

```
setenv KEYMAP sun.2-9
```

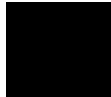
If you use xterm windows (the xterm window program is located in the directory /usr/openwin/demo), type:

```
setenv KEYMAP xterm.2-9
```

Reminder: If you are using OpenWindows, add /usr/openwin/bin to the end of the \$PATH definition, and add the following line to your .profile:

```
setenv OPENWINHOME /usr/openwin
```

After you have mapped your function keys, you must start the X server and an X window manager (if you are not currently running an X server).



Step 9. Determine the logical name of your emulator

The *logical name* of an emulator is a label associated with a set of communications parameters in the **\$HP64000/etc/64700tab.net** file. The 64700tab.net file is placed in the directory as part of the installation process.

- 1 Display the 64700tab.net file by entering **more \$HP64000/etc/64700tab.net** at the UNIX prompt.
- 2 Page through the file until you find the emulator you are going to use.

This step will require some matching of information to an emulator, but it should not be difficult to determine which emulator you want to address.

Examples

A typical entry for an 68000 emulator connected to the LAN would appear as follows:

```
#-----  
# Channel| Logical | Processor | Remainder of Information for the Channel  
# Type | Name | Type | (IP address for LAN connections)  
#-----  
lan: em68000 m68000 21.17.9.143
```

Step 10. Start the interface with the **emul700** command

- 1 Apply power to the emulator you wish to access after making sure the emulator is connected to the LAN.

On the HP 64700 Series Emulator, the power switch is located on the front panel near the bottom edge. Push the switch in to turn power on to the emulator.

- 2 Wait a few seconds to allow the emulator to complete its startup initialization.
- 3 Choose a terminal window from which to start the Graphical User Interface.
- 4 Start the Graphical User Interface by entering the **emul700** command and giving the logical name of the emulator as an argument to the command, as in

```
$HP64000/bin/emul700 <logical_name> &
```

or

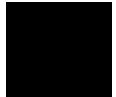
```
emul700 <logical name> &
```

if **\$HP64000/bin** is in your path.

If you are running the X server, if the Graphical User Interface is installed, and if your **DISPLAY** environment variable is set, the **emul700** command will start the Graphical User Interface. Otherwise, **emul700** starts the Softkey Interface.

You should include an ampersand ("&") with the command to start the Graphical User Interface as a background process. Doing so frees the terminal window where you started the interface so that the window may still be used.

- 5 Optionally start additional Graphical User Interface windows into the same emulation session by repeating the previous step.



Chapter 8: Installation
Installation for Sun SPARCsystems

You can also choose to use the Softkey Interface in a terminal emulation window, but you must include a command line argument to **emul700** to override the default Graphical User Interface. Start the Softkey Interface by entering

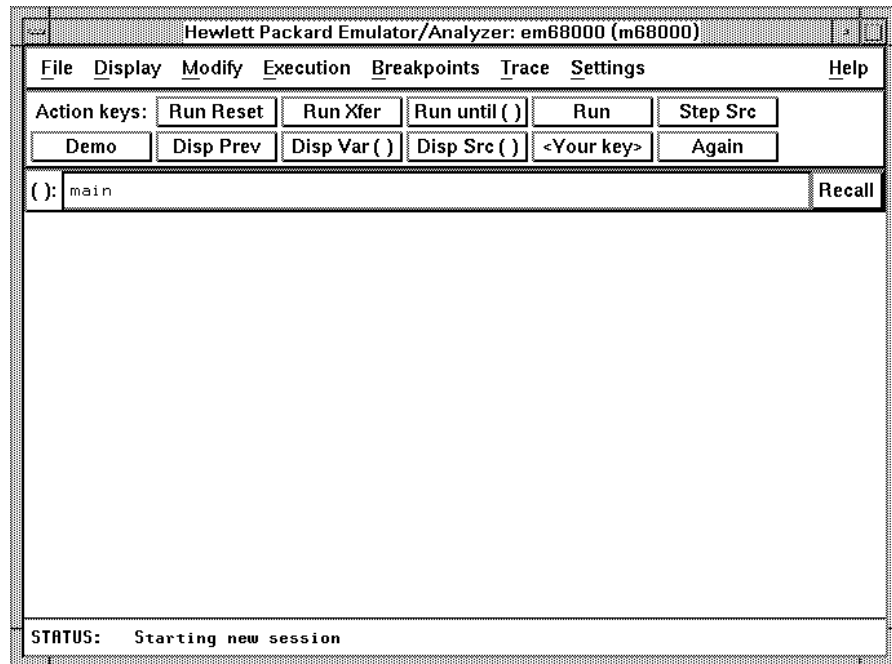
emul700 -u skemul <logical name>

Example

Suppose you have discovered that the logical name for a 68000 emulator connected to the LAN is "em68000". To start the Graphical User Interface and begin communicating with that emulator, enter (assuming your \$PATH includes \$HP64000/bin)

emul700 em68000

After a few seconds, the Graphical User Interface Emulator/Analyzer window should appear on your screen. The window will be similar to the following:



Step 11. Exit the Graphical User Interface

- 1 Position the mouse pointer over the pulldown menu named "File" on the menu bar at the top of the interface screen.
- 2 Press and hold the *command select* mouse button until the File menu appears.
- 3 While continuing to hold the mouse button down, move the mouse pointer down the menu to the "Exit" menu item.
- 4 Display the Exit cascade menu by moving the mouse pointer to the right edge of the Exit menu choice. There is an arrow on the right edge of the menu item.
- 5 Choose "Released" from the cascade menu.

The interface will terminate and release the emulator for use by others.





Index

- A**
 - absolute files, **111**
 - action keys, **19**
 - custom, **192**
 - operation, **65**
 - with command files, **192**
 - with entry buffer, **63, 65**
 - app-defaults directory
 - HP 9000 computers, **206**
 - Sun SPARCsystem computers, **206**
 - application resource
 - See X resource*

- B**
 - break, **133**
 - break to monitor from program, **133**
 - breakpoints, **27, 135-145**
 - clearing, **144**
 - clearing all, **145**
 - deactivating, **142**
 - enable/disable, **137**
 - permanent, setting, **139**
 - re-activating, **142**
 - screen to file, **178**
 - setting, **141**
 - setting all, **141**
 - breakpoints list, displaying, **136**

- C**
 - cascade menu, **56**
 - changing
 - column width, **167**
 - directory context in configuration window, **96, 107**
 - class name, X applications, **204**
 - client, X, **184**
 - closing emulator/analyzer windows, **51**
 - color scheme, **186, 190, 209**
 - columns in main display area, **187**
 - command buttons, **20**

- command files
 - other things to know about, **79**
 - passing parameters, **78**
- command line, **20**
 - Command Recall dialog box, **20**
 - Command Recall dialog box, operation, **75**
 - copy-and-paste to from entry buffer, **64**
 - editing entry area with keyboard, **75**
 - editing entry area with popup menu, **74**
 - editing entry area with pushbuttons, **73**
 - entering commands, **71**
 - entry area, **20**
 - executing commands, **71**
 - help, **76**
 - recalling commands with dialog box, **75**
 - turning on or off, **70, 187**
- command paste* mouse button, **22**
- Command Recall dialog box operation, **66**
- command select* mouse button, **22**
- commands
 - editing in command line entry area, **73-75**
 - entering in command line, **71**
 - executing in command line, **71**
 - line erase, **75**
 - recall, **75**
 - recalling with dialog box, **75**
 - word selection, **75**
- configuration help for items in dialog boxes, **97**
- configuration context, displaying from configuration window, **96, 107**
- configuration, emulator
 - exiting the interface, **98, 108**
 - loading from file, **98, 108**
 - modifying a section, **92, 104**
 - starting the interface, **90, 102**
 - storing, **95, 106**
- configuring the emulator, **87, 89-108**
- context
 - changing directory in configuration window, **96, 107**
 - changing directory in emulator/analyzer window, **120**
 - changing symbol, **120**
 - displaying directory from configuration window, **96, 107**

- context (continued)
 - displaying directory from emulator/analyzer window, **119**
 - displaying symbol, **119**
- copy-and-paste
 - addresses, **61**
 - from entry buffer, **64**
 - multi-window, **61, 64**
 - symbol width, **61**
 - to entry buffer, **60**
- copying
 - breakpoints screen to file, **178**
 - data values screen to file, **177**
 - display area to file, **177**
 - emulator status screen to file, **178**
 - error log to file, **178**
 - event log to file, **178**
 - global symbols to file, **178**
 - local symbols to file, **178**
 - memory to file, **177**
 - pod commands screen to file, **83, 178**
 - registers to file, **178**
 - trace listing to file, **177**
- cursor buttons, **20**
- D**
 - data values, **125-126**
 - adding items to the existing display, **126**
 - clearing the display and adding a new item, **126**
 - displaying, **31, 125**
 - screen to file, **177**
 - default trace specification, **147**
 - demos, setting up, **195-197**
 - dequeuer, how it works, **165**
 - device table file, **25, 45-46**
 - dialog box, **65**
 - Command Recall, operation, **66, 75**
 - Directory Selection, **120**
 - Directory Selection, operation, **65, 68**
 - Entry Buffer Recall, operation, **63, 66**
 - File Selection, operation, **66-67**
 - Ttrace Options, **162**
 - Trace Specification Selection, operation, **155**



- directory context
 - changing in configuration window, **96, 107**
 - changing in emulator/analyzer window, **120**
 - displaying from configuration window, **96, 107**
 - displaying from emulator/analyzer window, **119**
- Directory Selection dialog box operation, **65, 68**
- display area, **19**
 - columns, **187**
 - lines, **187-188**
 - screen to file, **177**
- display command
 - memory mnemonic, **26**
 - symbols, **113**
- display trace absolute command, **166**
- display trace absolute status binary command, **166**
- display trace absolute status hex command, **166**
- display trace absolute status mnemonic command, **166**
- display trace command, **161-171**
- display trace count absolute command, **168**
- display trace count command, **168**
- display trace count relative command, **168**
- display trace depth command, **170**
- display trace dequeue off command, **165**
- display trace dequeue on command, **165**
- display trace disassemble_from_line_number command, **164**
 - align_data_from_line option, **165**
- display trace mnemonic command, **164**
- display trace offset_by command, **168**
- displaying
 - registers, **127**
 - simulated io screen, **179, 181**
 - trace listing, **147**
- E** edit
 - command line entry area with popup menu, **74**
 - command line entry area with pushbuttons, **73**
 - file, **173**
 - file at address, **173**
 - file at program counter, **173**
 - file at symbol from symbols screen, **173**
 - file from memory display screen, **173**

edit command line entry area with keyboard, **75**

editing

- file, **187**
- file at address, **187**

emul700, command to start the emulator/analyzer interface, **45**

emulation configuration, **87, 89-108**

- configuration context, **107**
- exiting the configuration interface, **98, 108**
- fully graphical form, **89-100**
- help for items in dialog boxes, **97**
- help for making entries, **97**
- help for the softkey-based configuration, **108**
- loading a softkey-based configuration file, **108**
- loading an existing configuration file, **98**
- loading from file, **98, 108**
- mapping memory in graphical configuration, **99**
- modifying a configuration section, **92, 104**
- modifying a softkey-based section, **104**
- modifying the fully graphical form, **92**
- softkey-based configuration interface, **101-108**
- starting the configuration interface, **90, 102**
- starting the softkey-based configuration, **102**
- storing, **95, 106**
- storing a softkey-based configuration file, **106**
- storing changes to a file, **95**
- the directory context, **96**

emulation session, exiting, **52**

emulator

- configuring the, **88**
- device table file, **25, 45-46**
- running from target reset, **131**

emulator processor

- reset, **134**

emulator status

- displaying, **178**

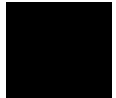
emulator/analyzer interface

- exiting, **40, 51-52**
- running in multiple windows, **45**
- starting, **45-48**

Index

- emulator/analyzer window
 - changing directory context, **120**
 - changing symbol context, **120**
- end command, **40**
- entry
 - pod commands, **83**
 - simulated io, **180**
- entry buffer, **19**
 - address copy-and-paste to, **61**
 - clearing, **60**
 - copy-and-paste from, **64**
 - copy-and-paste to, **60**
 - Entry Buffer Recall dialog box, **19**
 - Entry Buffer Recall dialog box, operation, **63**
 - multi-window copy-and-paste from, **64**
 - multi-window copy-and-paste to, **61**
 - operation, **63**
 - recall button, **19**
 - recalling entries, **63**
 - symbol width and copy-and-paste to, **61**
 - text entry, **60**
 - with action keys, **63, 65**
 - with pulldown menus, **63**
- Entry Buffer Recall dialog box operation, **66**
- environment variables (UNIX)
 - HP64KPATH, **81**
 - PATH, **45**
- error log
 - displaying, **175**
 - to file, **178**
- event log
 - displaying, **176**
 - to file, **178**
- event_log, **49-50**
- exit, emulator/analyzer interface, **40, 51-52**
- exiting
 - emulation session, **52**
 - emulator/analyzer windows, **51**

- F** file
- breakpoints screen to, **178**
 - copying the pod commands screen, **83**
 - data values screen to, **177**
 - display area to, **177**
 - editing, **173**
 - editing at address, **173**
 - editing at program counter, **173**
 - editing at symbol from symbols screen, **173**
 - editing from memory display screen, **173**
 - emulator configuration, **95, 106**
 - emulator configuration load, **98, 108**
 - emulator status screen to, **178**
 - error log to, **178**
 - event log to, **178**
 - global symbols to, **178**
 - local symbols to, **178**
 - memory to, **177**
 - pod commands screen to, **178**
 - registers to, **178**
 - trace data load, **156**
 - trace data to, **156**
 - trace listing to, **177**
 - trace specification load, **157**
 - trace specification to, **157**
- file extension .EA configuration files, **95**
- File Selection dialog box operation, **66-67**
- formal parameters (command files), **78**
- functions, step over, **122**
- G** global symbols, **26**
- display command, **113**
 - to file, **178**
- grayed-out, **72-73**



- H**
 - hand pointer, **19, 59**
 - hardware
 - HP 9000 memory needs, **215**
 - HP 9000 minimum performance, **215**
 - HP 9000 minimums overview, **214**
 - SPARCsystem memory needs, **216**
 - SPARCsystem minimum performance, **216**
 - SPARCsystem minimums overview, **216**
 - help
 - command line, **76**
 - help index, **69**
 - for emulation configuration entries, **97**
 - HP 9000
 - 700 series Motif libraries, **215**
 - HP-UX minimum version, **215**
 - installing software, **217-227**
 - minimum system requirements overview, **214**
 - HP-UX, minimum version, **215**
 - HP64KPATH, UNIX environment variable, **81**
- I**
 - IEEE-695 absolute file format, **111**
 - input
 - pod commands, **83**
 - simulated io, **180**
 - input scheme, **186, 209**
 - installation
 - at a glance, **214-216**
 - HP 9000 overview, **214**
 - HP 9000 specific instructions, **217-227**
 - SPARCsystem specific instructions, **228-237**
 - SPARCsystems overview, **215**
 - instance name, X applications, **203-204**
 - interface
 - exiting, **52**
 - fully graphical emulation configuration, **89-100**
 - softkey-based emulation configuration, **101-108**
 - interface, emulator configuration
 - exiting, **98, 108**
 - modifying a section, **92, 104**
 - starting, **90, 102**
 - inverse video, graphical interface demo/tutorial files, **196**

- K** keyboard
 - accelerators, **58**
 - choosing menu items, **57**
 - focus policy, **58**
 - pod commands, **83**
 - simulated io, **180**

- L** label scheme, **186, 190, 209**
 - LANG environment variable, **209**
 - LD_LIBRARY_PATH environment variable, **232**
 - libraries, Motif for HP 9000/700, **215**
 - line numbers (source file), symbol display, **114**
 - lines in main display area, **187-188**
 - loading
 - trace data file, **156**
 - trace specification file, **157**
 - local symbols
 - display command, **114**
 - to file, **178**

- M** memory
 - displaying, **121**
 - displaying at an address, **123**
 - displaying repetitively, **124**
 - mnemonic format display, **122**
 - modifying, **124**
 - to file, **177**
 - memory recommendations
 - HP 9000, **215**
 - SPARCsystem, **216**
 - menu, popup menu in trace list, **163**
 - menus
 - editing command line with popup, **74**
 - hand pointer means popup, **19, 59**
 - pull-down operation with keyboard, **57**
 - pull-down operation with mouse, **56-57**
 - mnemonic memory display, **26, 122**
 - mnemonic memory display, setting the source/symbol modes, **158**
 - modes, source/symbol, **158**
 - modify, registers, **128**

- monitor, breaking to from program, **133**
- Motif, HP 9000/700 requirements, **215**
- mouse buttons, **22**
- mouse, choosing menu items, **56-57**
- multi-window
 - copy-and-paste from entry buffer, **64**
 - copy-and-paste to entry buffer, **61**
- N**
 - nesting command files, **77**
 - NOT TAKEN in trace list, **165**
 - notes, breakpoint locations must contain opcodes, **139, 141**
- O**
 - offset addresses in trace list, **168**
 - operating system
 - HP-UX minimum version, **215**
 - SunOS minimum version, **216**
 - overview
 - HP 9000 installation, **214**
 - installation, **214-216**
 - SPARCsystems installation, **215**
- P**
 - parameter passing in command files, **78**
 - parent symbol, displaying from symbols screen, **116**
 - paste* mouse button, **22**
 - PATH, UNIX environment variable, **45**
 - platform
 - HP 9000 memory needs, **215**
 - HP 9000 minimum performance, **215**
 - SPARCsystem memory needs, **216**
 - SPARCsystem minimum performance, **216**
 - platform scheme, **186, 208**
 - pod commands
 - copying screen to a file, **83**
 - display screen, **83**
 - keyboard input, **83**
 - screen to file, **178**
 - popup menu in trace list, **163**
 - popup menus
 - command line editing with, **74**
 - hand pointer indicates presence, **19, 59**

- prestore qualifier, **153**
- processor type, **46**
- processor, emulator reset, **134**
- program
 - breaking to monitor, **133**
 - running until address reached, **131**
- program counter
 - mnemonic memory display, **27**
 - running from, **130**
- programs
 - loading, **111**
 - loading only symbols, **112**
 - loading without symbols, **112**
 - stepping assembly-level instructions, **133**
 - stepping high-level source lines, **132**
- pull-down menus
 - choosing with keyboard, **57**
 - choosing with mouse, **56-57**
- pushbutton select* mouse button, **22**

- Q** qualifier
 - storage, using pull-downs, **152**
 - trigger, **148-151**

- R** recall buffer, **19**
 - columns, **193**
 - initial content, **193-194**
 - lines, **193**
 - recalling entries, **63**
- recall, command, **75**
 - dialog box, **75**
- recalling trace specifications with dialog box, **155**
- registers
 - displaying, **32, 127**
 - modify, **128**
 - to file, **178**
- release_system, end command option, **40**
- repetitive
 - display of memory, **124**
 - trace, **155**



- reset (emulator), commands which cause exit from, **134**
- reset emulator processor, **134**
- reset trace display defaults, **169**
- reset, run from, **131**
- resource
 - See* X resource
 - RESOURCE_MANAGER property, **206-207**
- run command, **130**
 - from reset, **131**
- run program until address, **131**
- S**
 - scheme files (for X resources), **185, 208-210**
 - color scheme, **186, 190, 209**
 - custom, **190-191, 210**
 - input scheme, **186, 209**
 - label scheme, **186, 190, 209**
 - platform scheme, **186, 208**
 - size scheme, **186, 209**
 - scroll bar, **19**
 - select* mouse button, **22**
 - server, X, **184, 206**
 - set command, **161-171**
 - set default command, **169**
 - set source off command, **167**
 - set source on command, **167**
 - set source only command, **167**
 - set symbols all command, **166**
 - set symbols high command, **166**
 - set symbols low command, **166**
 - set symbols off command, **166**
 - set symbols on command, **166**
 - set width label command, **167**
 - set width mnemonic command, **167**
 - set width source command, **167**
 - shell variables, **79**
 - simulated io
 - displaying screen, **179, 181**
 - keyboard input, **180**
 - size scheme, **186, 209**

- softkey pushbuttons, **20**
- softkey-based emulation configuration, **101-108**
- software
 - installation for HP 9000, **217-227**
 - installation for SPARCsystems, **228-237**
- software breakpoints, opcode locations, **139, 141**
- source lines
 - display in trace list, **167**
 - symbol display, **114**
- source/symbol modes, setting, **158**
- SPARCsystems
 - installing software, **228-237**
 - minimum system requirements overview, **216**
 - SunOS minimum version, **216**
- specify trace dequeuing options, **165**
- start
 - default trace, **147**
 - trace storing address, **152**
- states, change the number available for display, **170**
- status line, **19**
- status line (display), **50**
- status, displaying, **177**
- status, emulator screen to file, **178**
- step command, **28**
- step over, **122**
- stepping programs
 - by assembly instructions, **133**
 - by high-level source lines, **132**
- stop
 - trace, **154**
 - trace until, **154**
- storing
 - trace data to file, **156**
 - trace specification to file, **157**
 - using pulldowns to define qualifier, **152**
- SunOS, minimum version, **216**
- switching directory context in configuration window, **96, 107**
- symbol context
 - changing, **120**
 - displaying, **119**

- symbol file, loading, **113**
- symbols, **113**
 - display local, **114**
 - displaying, **113**
 - displaying in trace list, **166**
 - displaying parent from symbols screen, **116**
 - global to file, **178**
 - loading only, **112**
 - loading with program, **111**
 - local to file, **178**
- system requirements
 - HP 9000 overview, **214**
 - HP-UX minimum version, **215**
 - OSF/Motif HP 9000/700 requirements, **215**
 - SPARCsystem overview, **216**
 - SunOS minimum version, **216**

T **TAKEN, NOT TAKEN, and ?TAKEN? in trace list, 165**

- target memory ROM, symbols for, **112**
- terminal emulation window, opening, **178**
- terminal interface, copying screen to a file, **83**
- trace
 - about an address, **151**
 - after an address, **148**
 - before an address, **149**
 - before an address, break on trigger, **150**
 - default specification, **147**
 - displaying count information, **168**
 - displaying listing, **147**
 - displaying without disassembly, **166**
 - everything, **147**
 - listing to file, **177**
 - loading trace data from file, **156**
 - loading trace specification from file, **157**
 - recalling trace specifications, **155**
 - repetitively, **155**
 - reset display defaults, **169**
 - stopping, **154**
 - storage qualifier using pulldowns, **152**
 - storage qualifier with prestore, **153**
 - storing trace data to file, **156**

- trace (continued)
 - storing trace specification to file, **157**
 - Trace Specification Selection dialog box, **155**
 - until stop, **154**
 - trace counting off command, **168**
 - trace dequeueing, specifying options, **165**
 - trace display, setting the source/symbol modes, **158**
 - trace list
 - disassembly, **164**
 - display around specific line number, **169**
 - display source lines, **167**
 - displaying, **161-171**
 - offset addresses, **168**
 - popup menu, **163**
 - trace options dialog box, **162**
 - transfer address, **130**
 - trigger position
 - bottom of trace, **149-150**
 - middle of trace, **151**
 - top of trace, **148**
 - tutorials, setting up, **195-197**
- U** user program
 - breaking to monitor, **133**
 - running until address reached, **131**
 - stepping assembly-level instructions, **133**
 - stepping high-level source lines, **132**
- W** wait command
 - command files, using in, **77**widget resource
 - See* X resourcewindow
 - exiting emulator/analyzer, **51**window, terminal emulation, opening, **178**windows
 - opening additional emulator/analyzer, **49**
 - running the emulator/analyzer interface in multiple, **45**

- workstation
 - HP 9000 memory needs, **215**
 - HP 9000 minimum performance, **215**
 - SPARCsystem memory needs, **216**
 - SPARCsystem minimum performance, **216**
- X** X client, **184**
- X resource, **184**
 - \$XAPPLRESDIR directory, **207**
 - \$XENVIRONMENT variable, **207**
 - .Xdefaults file, **206**
 - /usr/hp64000/lib/X11/HP64_schemes, **209**
 - app-defaults file, **206**
 - class name for applications, **204**
 - class name for widgets, **204**
 - command line options, **207**
 - commonly modified graphical interface resources, **186**
 - defined, **203-205**
 - general form, **203**
 - instance name for applications, **204**
 - instance name for widgets, **203**
 - loading order, **207**
 - modifying resources, generally, **186-189**
 - RESOURCE_MANAGER property, **207**
 - scheme file system directory, **209**
 - scheme files, Graphical User Interface, **208-210**
 - scheme files, named, **209**
 - schemes, forcing interface to use certain, **208**
 - Softkey.BW, **209**
 - Softkey.Color, **209**
 - Softkey.Input, **209**
 - Softkey.Label, **209**
 - Softkey.Large, **209**
 - Softkey.Small, **209**
 - wildcard character, **204**
 - xrdb, **207**
 - xrm command line option, **207**
- X server, **184, 206**
- X Window System, **45**

Certification and Warranty

Certification

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

Warranty

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

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

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

Limitation of Warranty

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

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

Exclusive Remedies

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

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

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