User's Guide

HP Debug 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.

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

Printing History

New editions are complete revisions of the manual. The date on the title page changes only when a new edition is published.

A software code may be printed before the date; this indicates the version level of the software product at the time the manual was issued. Many product updates and fixes do not require manual changes, and manual corrections may be done without accompanying product changes. Therefore, do not expect a one-to-one correspondence between product updates and manual revisions.

Edition 1 B3750-97000, July 1996

Edition 2 B3750-97004, March 1997

Certification and Warranty

Certification and warranty information can be found at the end of this manual on the pages before the back cover.

The HP Debug User Interface — Overview

The HP Debug User Interface is a multi-platform application that lets you debug C language programs for embedded microprocessor systems. The interface is available on Microsoft Windows 95, HP-UX, SunOS, and Solaris.

The debugger controls HP emulators and analyzers on the local area network (LAN). It takes full advantage of the emulator's real-time capabilities to allow effective debug of C programs while running in real time.

Work in a window-based application.

- You can display different types of debugger information in different windows, just as you display other windows in applications running on the platform you work with.
- You can complete a wide variety of debug-related tasks without exiting the debugger. For example, you can edit files or compile your programs without exiting.
- You can cut or copy text from a window to the entry buffer, and copied contents may be pasted into other windows or dialog boxes.

Communicate at high speeds.

- You can use the emulator's LAN connection for high-speed data transfer (including program download). These connections give you an efficient debugging environment.

Debug programs in C context.

- You can display C language source files (optionally with intermixed assembly language code).
- You can display program symbols.
- You can display the stack backtrace.
- You can display and edit the contents of program variables.
- You can step through programs, either by source line or assembly language instruction.
- You can step over functions.
- You can run programs until the current function returns.

- You can run programs up to a particular source line or assembly language instruction.
- You can set breakpoints in the program.

Display and modify processor resources.

- You can display and edit the contents of memory locations in hexadecimal or as C variables.
- You can display and edit the contents of microprocessor registers, including on-chip peripheral registers.

Trace program execution.

- You can trace control flow within a function at the C statement level.
- You can trace all C statements that access a variable.
- You can trace before, and break program execution on, a C variable being set to a specified value.
- You can make custom trace specifications.

Debug your program while it runs continuously at full speed.

- You can configure the debugger to prevent it from automatically initiating any action that may interrupt user program execution. This ensures that the user program executes in real time, so you can debug your design while it runs in a real-world operating mode.
- You can inspect and modify C variables and data structures without interrupting execution.
- You can set and clear breakpoints without interrupting execution.
- You can perform all logic analysis functions, observing C program and variable activity, without interrupting program execution.

In This Book

This book documents the HP Debug User Interface:

Chapter 1 quickly shows you how to use the debugger.
Chapter 2 shows you how to use the debugger interface.
Chapter 3 shows you how to debug programs.
Chapter 4 describes commands that appear in the Debug window.
Chapter 5 describes commands that appear in the windows that can be invoked from the Debug window.
Chapter 6 describes commands that appear in pop-up menus.
Chapter 7 summarizes the debugger commands as they are used in command files.
Chapter 8 describes the format for expressions used in commands.
Chapter 9 contains conceptual (and more detailed) information on various topics.
Chapter 10 describes how to customize the HP Debug User Interface.

Glossary of terms is summarized in the end of this book.

Contents

1 Getting Started

Step 1. Start the debugger 17
Step 2. Set the hardware options 19
Step 3. Map memory for the demo program 20
Step 4. Load the demo program 21
Step 5. Display the source file 22
Step 6. Set a breakpoint 23
Step 7. Run the demo program 24
Step 8. Delete the breakpoint 25
Step 9. Step over a function 26
Step 10. Single-step one line 27
Step 11. Run the program to a specified line 28
Step 12. Run the program until the current function returns 29
Step 13. Display a variable 30
Step 14. Edit a variable 31
Step 15. Display register contents 32
Step 16. Trace accesses to a variable 33
Step 17. Exit the debugger 34

2 Using the Debugger Interface

Starting and Exiting the Debugger 37

To start the Debugger 37
To exit the Debugger 38

Working with Debugger Windows 39

To open debugger windows 39
To copy window contents to the file 40
To set tabstops in the Debug window 40

Using the Entry Buffer 41

History Function 41

Using Action Buttons 42

Using Command Files 43

To create a command file 43
To execute a command file 44

3 Debugging Programs

Loading and Displaying Programs 49

To load user programs 50
To display source files by their names 51
To display source code specifying a heading line 51
To display source code from the current program counter 52
To display source code or mnemonics only 52
To display source code mixed with assembly instructions 53
To highlight source code 54
To specify source file directories 55

Running, Stepping, and Stopping the Program 56

To run the program from the current program counter 57
To run the program from a specified address 57
To run the program from the start address 58
To run the program from target reset 58
To run the program to a specified address 58
To run the program until the current function return s 59
To step a single line or instruction from the current program counter 60
To step a single line or instruction from a specified address 61
To step a single line or instruction from the start address 61
To step over a function 62
To stop program execution 64
To reset the processor 65

Using Breakpoints 66

To set a breakpoint 67
To list the breakpoints 69
To disable a breakpoint 69
To delete breakpoints 70

Displaying and Editing Variables 71

To display a variable 71

To edit a variable 73

Displaying and Editing Memory 74

To display memory 75
To edit memory 77
To fill memory 78

Displaying and Editing Peripheral Registers 79

To display peripheral registers 79
To edit peripheral registers 80

Displaying and Editing Registers 81

To display registers 82
To edit registers 83

Tracing Program Execution 84

How the tracing function works 85
To trace accesses to every state 87
To start tracing from a specified address 87
To trace including a specified address in the center of the trace list 88
To end tracing at a specified address 89
To trace accesses to a specified address 89
To trace accesses to a specified address and break 90
To trigger tracing on an access to a specified variable 91
To trace until the command is halted 93
To stop a running trace 93
To repeat the last trace 93
To search trigger/string in the trace list 94
To specify the trace clock 95

Setting Display Mode for Tracing Results 96

To display bus cycles 97
To specify display mode for the trace count 98
To display symbol information in the trace list 98
To display the trace list with function names 99
To display the trace list with line numbers 99
To highlight source code in the trace list 99

Setting Up Custom Trace Specifications 100

To set up an appropriate trace specification 100

To set up a Condition trace specification 101
To set up a Sequence trace specification 103
To store and load trace specifications 107

Saving and Loading Configurations 108

To save the current emulator configuration 108
To load the emulator configuration 109

4 Debug Window Commands

File→Load→Configuration... (ALT, F, L, C) 114
File→Load→Program... (ALT, F, L, P) 115
File→Store→Configuration... (ALT, F, S, C) 117
File→Copy→Display... (ALT, F, P, D) 118
File→Log→Playback... (ALT, F, O, P) 119
File→Log→Record... (ALT, F, O, R) 121
File→Log→Stop (ALT, F, O, S) 122
Execution→Run→From PC (ALT, E, R, P) 123
Execution→Run→From () (ALT, E, R, F) 123
Execution→Run→From Start Address (ALT, E, R, S) 124
Execution→Run→From Reset (ALT, E, R, R) 124
Execution→Run→Until () (ALT, E, R, U) 125
Execution→Run→Return to Caller (ALT, E, R, C) 126
Execution→Step→From PC (ALT, E, S, P) 127
Execution→Step→From () (ALT, E, S, F) 127
Execution→Step→From Start Address (ALT, E, S, S) 128
Execution→Step→Over Procedure Call (ALT, E, S, O) 129
Execution→Break (ALT, E, B) 130
Execution→Reset (ALT, E, T) 131
Execution→Breakpoints... (ALT, E, P) 132
Display→Program At PC (ALT, D, P) 135
Display→Program At () (ALT, D, A) 135
Display→Source Files... (ALT, D, S) 136
Display→Find... (ALT, D, F) 137
Window→Source (ALT, W, S) 139
Window→Register (ALT, W, R) 139
Window→Memory (ALT, W, M) 139
Window→Variable (ALT, W, V) 140
Window→Peripheral (ALT, W, P) 140

Window→Backtrace (ALT, W, B) 141
Settings→Display Mode→Source Only (ALT, S, D, S) 142
Settings→Display Mode→Mixed (ALT, S, D, M) 142
Settings→Display Mode→Mnemonics Only (ALT, S, D, N) 143
Settings→Display Mode→Symbols (ALT, S, D, Y) 143
Settings→Display Mode→Highlight Source Lines (ALT, S, D, H) 144
Settings→Source View... (ALT, S, S) 145
Settings→Configuration→Hardware... (ALT, S, C, H) 147
Settings→Configuration→Memory Map... (ALT, S, C, M) 147

5 Window Control Menu Commands

Common Control Menu Commands 151

File→Copy→Display... (ALT, F, P, D) 152
File→Close (ALT, F, C) 152

HP Debug User I/F Window Commands 153

File→Exit (ALT, F, X) 154
Connect→Emulator... (ALT, C, E) 155
Connect→Logic Analyzer... (ALT, C, L) 156
Window→Tile (For PC) (ALT, W, T) 158
Window→Cascade (For PC) (ALT, W, C) 158
Window→Arrange Icons (For PC) (ALT, W, A) 158
Help→Contents (ALT, H, C) 159
Help→Search for Help on... (ALT, H, S) 159
Help→How to Use Help (ALT, H, H) 159
Help→Tutorial (ALT, H, T) 160
Help→Version... (ALT, H, V) 160

Trace Window Commands 161

Trace→Everything (ALT, T, E) 162
Trace→After () (ALT, T, F) 162
Trace→About () (ALT, T, A) 163
Trace→Before () (ALT, T, B) 164
Trace→Only () (ALT, T, O) 164
Trace→Until () (ALT, T, U) 165
Trace→Until Halt (ALT, T, H) 165
Trace→Variable... (ALT, T, V) 166

Trace→Condition... (ALT, T, C) 168
Trace→Sequence... (ALT, T, S) 170
Trace→Again (ALT, T, G) 173
Trace→Halt (ALT, T, T) 173
Display→Trigger (ALT, D, T) 174
Display→Find... (ALT, D, F) 175
Settings→Display Mode→Source Only (ALT, S, D, S) 177
Settings→Display Mode→Mixed (ALT, S, D, M) 177
Settings→Display Mode→Bus Cycles Only (ALT, S, D, B) 178
Settings→Display Mode→Symbols (ALT, S, D, Y) 178
Settings→Display Mode→Function Names (ALT, S, D, F) 179
Settings→Display Mode→Line Numbers (ALT, S, D, L) 179
Settings→Display Mode→Highlight Source Lines (ALT, S, D, H) 180
Settings→Trace Count→Relative (ALT, S, C, R) 181
Settings→Trace Count→Absolute (ALT, S, C, A) 182
Settings→Trace Clock Speed→Very Fast (ALT, S, S, V) 183
Settings→Trace Clock Speed→Fast (ALT, S, S, F) 184
Settings→Trace Clock Speed→Slow (ALT, S, S, S) 185

Source Window Commands 186

Display→Program At () (ALT, D, A) 187
Display→Source Files... (ALT, D, S) 188
Display→Find... (ALT, D, F) 189
Settings→Display Mode→Source Only (ALT, S, D, S) 191
Settings→Display Mode→Mixed (ALT, S, D, M) 191
Settings→Display Mode→Mnemonics Only (ALT, S, D, N) 192
Settings→Display Mode→Symbols (ALT, S, D, Y) 192
Settings→Display Mode→Highlight Source Lines (ALT, S, D, H) 193

Register Window Commands 194

Settings→Auto Update (ALT, S, A) 194

Memory Window Commands 195

Display→From () (ALT, D, F) 196
Display→Blocked→Byte (ALT, D, B, B) 196
Display→Blocked→Word (ALT, D, B, W) 197
Display→Blocked→Long (ALT, D, B, L) 197
Display→Absolute→Byte (ALT, D, A, B) 198

Display→Absolute→Word (ALT, D, A, W) 198
Display→Absolute→Long (ALT, D, A, L) 199
Display→Absolute→Float (ALT, D, A, F) 199
Modify→Memory... (ALT, M, M) 200
Settings→Auto Update (ALT, S, A) 202

Variable Window Commands 203

Display→Variable () (ALT, D, V) 204
Display→Address Of (ALT, D, A) 205
Display→Contents Of (ALT, D, C) 205
Modify→Variable... (ALT, M, V) 206
Settings→Display Base→Default (ALT, S, D, D) 208
Settings→Display Base→Decimal (ALT, S, D, C) 208
Settings→Display Base→Hexadecimal (ALT, S, D, H) 209
Settings→Display Base→Float (ALT, S, D, F) 209
Settings→Display Base→String (ALT, S, D, S) 210
Settings→Auto Update (ALT, S, A) 211

Peripheral Window Commands 212

Display→ < register class name > 212
Settings→Auto Update (ALT, S, A) 213

Backtrace Window Commands 214

[Source at Stack Level] 214

6 Window Pop-up Commands

Debug Window Pop-up Commands 217

Set Breakpoint 217
Clear Breakpoint 217
Run to Here 217
Evaluate () 218

Source Window Pop-up Commands 219

Set Breakpoint 219
Clear Breakpoint 219
Evaluate () 219

Backtrace Window Pop-up Commands 220
Source at Stack Level 220

7 Command File Command Summary

8 Expressions in Commands

Numeric Constants 229
Symbols 230
C Operators 231
Address Range 232

9 Concepts

Debugger Windows 235
The HP Debug User I/F Window 236
The Debug Window 237
The Trace Window 240
The Source Window 242
The Register Window 243
The Memory Window 244
The Variable Window 246
The Peripheral Window 248
The Backtrace Window 249

10 Customizing the HP Debug User Interface

General information on customizing 253

Customizing global setting of Debug User Interface 254

Customizing each windows 255

Customizing the Action Buttons 259

Customizing the Debugger Interface's Appearance (for workstations) 260

Glossary

Index

1

Getting Started

Getting Started

This tutorial helps you get comfortable with the software by showing you how to perform some measurements on a demo program, such as how:

Step 1.	To start the debugger.
Step 2.	To set the hardware options.
Step 3.	To map memory for the demo program.
Step 4.	To load the demo program.
Step 5.	To display the source file.
Step 6.	To set a breakpoint.
Step 7.	To run the demo program.
Step 8.	To delete a breakpoint.
Step 9.	To step over a function.
Step 10.	To single-step one line.
Step 11.	To run the program to a specified line.
Step 12.	To run the program until the current function returns.
Step 13.	To display a variable.
Step 14.	To edit a variable.
Step 15.	To display register contents.
Step 16.	To trace accesses to a variable.
Step 17.	To exit the debugger.

The SAMPLE demo program, which is a simple C program that does case conversion on a few strings, is included with the HP Debug User Interface.

Also available is the ECS demo program, which is somewhat more complex C program for an environment control system.

The demo program directory contains a README file that describes the program, and batch files that show you how the object files were made.

This tutorial shows you how to perform some measurements on the SAMPLE demo program.

Step 1. Start the debugger

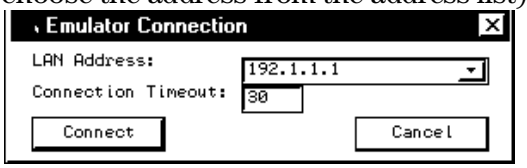
For workstations:

1 Enter the following command from the command line and press Return:

```
$ netrap &
```

Under normal operation, the HP Debug User I/F window opens, then the Copyright dialog box and the Emulator Connection dialog box appear.

2 Enter the LAN address for the emulator in the Emulator Connection dialog box (or choose the address from the address list).



You can add the LAN address in the address list to specify it in the initialization file. See Chapter 10, "Customizing the HP Debug User Interface," for details.

3 Click the Connect button.

When the connection has been made, the Debug window and the Trace window open.

Or:

1 If you know the LAN address for the emulator, you may enter the following command:

```
$ netrap -a <emulator_host_name> &
```

You can check the detailed LAN address for the emulator using the /etc/hosts file. The file is found on the local computer that runs the HP Debug User Interface.

For PC:

1 Choose HP B37XXX in the HP Debug User Interface group from the Windows 95 Start menu.

Or:

a. Click the Windows 95 Start button and choose the Run (ALT, S, R) command.

b. Enter the debugger startup file name C:\hpntr\b37xxa\Netrap.exe (if C:\hpntr\b37xxa was the installation path chosen during installation).

c. Click the OK button. Under normal operation, the HP Debug User I/F window opens, then the Copyright dialog box and the Emulator Connection dialog box appear.

You can accelerate start-up operations by specifying the IP address for the emulator when installing the debugger, which enables the debugger to connect to the emulator without opening the Emulator Connection dialog box.

2 Enter the LAN address for the emulator in the Emulator Connection dialog box (or choose the address from the address list).

You can add the LAN address in the address list to specify it in the initialization file. See Chapter 10, "Customizing the HP Debug User Interface," for details.

3 Click the Connect button.
When the connection has been made, the Debug window and the Trace window open.

Step 2. Set the hardware options

The HP Debug User Interface comes with the file sample.cfg, which contains configurations for specific emulators. Loading this file configures the system hardware options automatically.

1 Choose the File→Load→Configuration... (ALT, F, L, C) command.

The file selection dialog box appears.

2 Select the file sample.cfg, then click the OK button.

The configurations specified in sample.cfg are loaded.
You can check the hardware options configured with sample.cfg by choosing the Settings→Configuration→Hardware... (ALT, S, C, H) command from the Debug window.

Step 3. Map memory for the demo program

Emulation memory for the demo program must be mapped before you can use the emulator. Loading the sample.cfg file (step 2) maps emulation memory automatically.

You can check the memory map specified by sample.cfg by choosing the Settings→Configuration→Memory Map... (ALT, S, C, M) command from the Debug window.

Step 4. Load the demo program

1 Choose the File→Load→Program... (ALT, F, L, P) command.

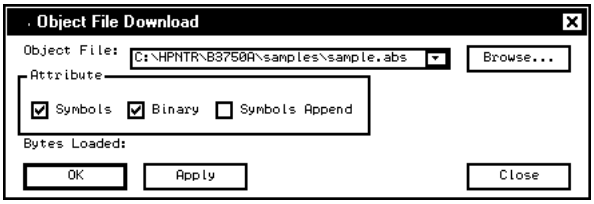
The Object File Download dialog box appears.

2 Click the Browse button and select the sample program object file sample.abs.

The file resides in the following directory (if C:\hpntr\b37xxa (for PC) or /usr/hpntr/hp700_9 (for workstations) was the installation path chosen during installation):

For workstations: /usr/hpntr/hp700_9/samples

For PC: C:\hpntr\b37xxa\samples



3 Select the Symbols option and the Binary option.

4 Click the OK button.

Note

Click the Apply button instead of the OK button to check how many bytes of data will be loaded. The number of bytes to be loaded is displayed below the Attribute group box. Click the Close button to close the dialog box.

Step 5. Display the source file

To display sample.c, the source file of sample.abs, starting from the main function in the Debug window:

1 Confirm that "main" is displayed in the entry buffer of the Debug window.

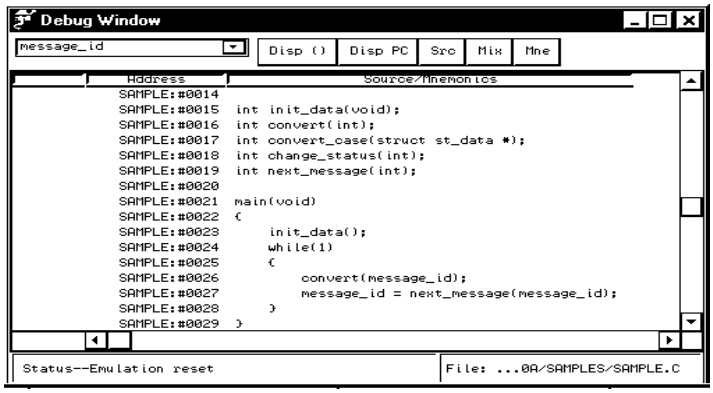
You can define the initial value of the entry buffer in the initialization file. See Chapter 10, "Customizing the HP Debug User Interface," for details.

2 Click the Disp () action button.

The window displays the sample.c source file, starting from the main function.

3 From the Debug window's control menu, choose the Settings→Display Mode→Source Only (ALT, S, D, S) command.

The window displays the sample.c source file in source-only mode.



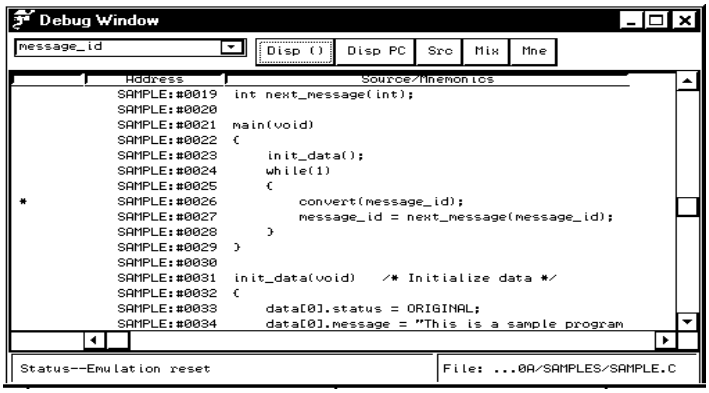
Step 6. Set a breakpoint

To set a breakpoint on the line where the "convert" function is called:

1 Cursor-select the line containing "convert" and press the right mouse button.

The pop-up menu appears.

2 Choose Set Breakpoint from the pop-up menu.



Note that the line containing "convert" is marked with an "*", indicating that a breakpoint has been set on the line.

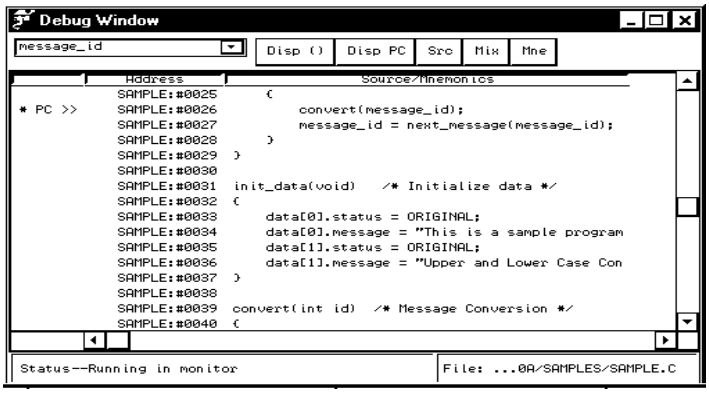
You can set breakpoints from the Source window as well as from the Debug window.

Step 7. Run the demo program

To run the demo program from the start address:

1. From the Debug window, choose the Execution→Reset (ALT, E, T) command, followed by the Execution→Break (ALT, E, B) command, to initialize the stack pointer.

2. Choose the Execution→Run→From Start Address (ALT, E, R, S) command.



Note that the demo program runs until the line marked with an "*". The "PC>>" marker indicates the current location of the program counter.

Step 8. Delete the breakpoint

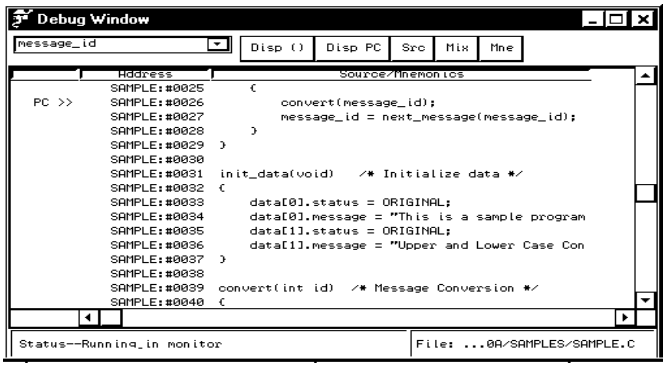
To delete the breakpoint you set in step 7:

1 Cursor-select the line marked with an "*" and press the right mouse button.

The pop-up menu appears.

2 Choose Clear Breakpoint from the pop-up menu.

The "*" marker disappears from the Debug window.

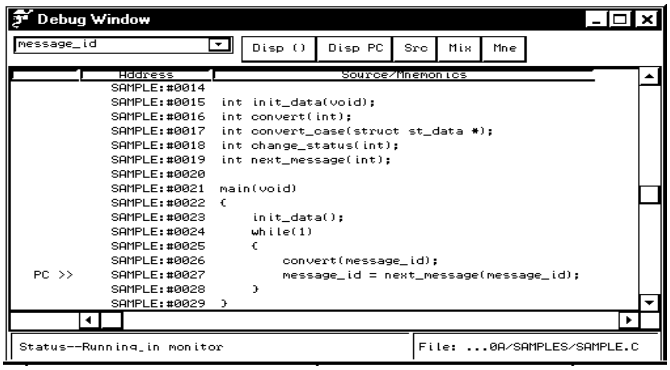


Step 9. Step over a function

To step over the "convert" function:

- From the Debug window, choose the Execution→Step→Over Procedure Call (ALT, E, S, O) command.

The "convert" function executes, and the program counter moves to the next line.

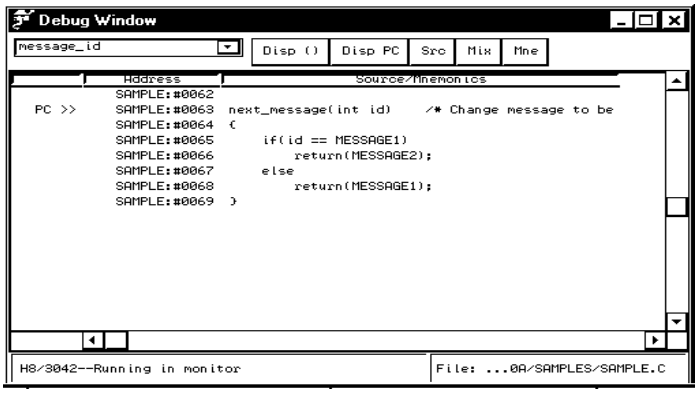


Step 10. Single-step one line

To single-step the demo program from the current program counter:

- From the Debug window's control menu, choose the Execution→Step→From PC (ALT, E, S, P) command.

  Note that the C statement executes, and the program counter moves to the "next_message" function.



Step 11. Run the program to a specified line

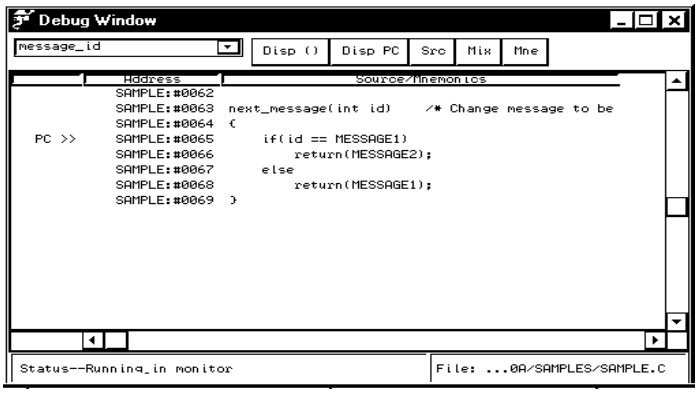
To execute the demo program to the first line of the "next_message" function:

1 Cursor-select the *if* statement in the "next_message" function and press the right mouse button.

The pop-up menu appears.

2 Choose Run to Here from the pop-up menu.

The program executes and stops immediately before the line that you specified.

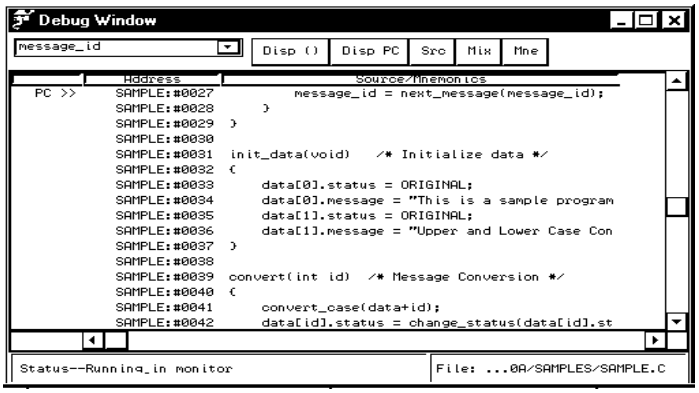


Step 12. Run the program until the current function returns

To execute the program until the "next_message" function (the current PC function) returns to its caller:

- From the Debug window, choose the Execution→Run→Run to Caller (ALT, E, R, C) command.

The program executes until it reaches the line that called "next_message".



Step 13. Display a variable

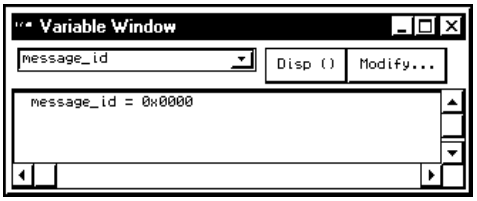
To display the contents of the "message_id" variable:

1 Double-click to highlight "message_id" on the line showing the "PC>>" marker in the Debug window.

The entry buffer displays "message_id".

2 From the Debug window's control menu, choose the Window→Variable (ALT, W, V) command.

The Variable window appears. The text box displays "message_id". Note that the list box displays the contents of "message_id".



Step 14. Edit a variable

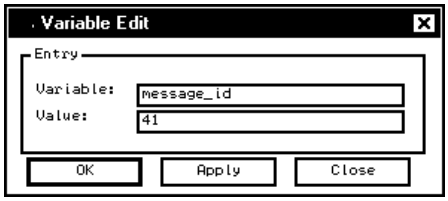
To edit the contents of the "message_id" variable:

1 From the Variable window, choose the Modify→Variable... (ALT, M, V) command.

The Variable Edit dialog box appears.

2 Enter "41" in the Value: text box.

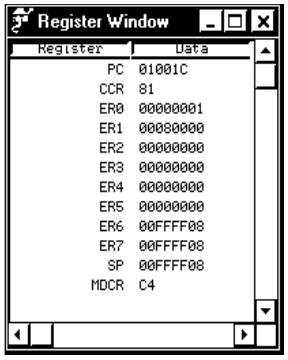
3 Click the Apply button.



Step 15. Display register contents

- From the Debug window's control menu, choose the Window→Register (ALT, W, R) command.

  The Register window opens and displays the register contents.



Step 16. Trace accesses to a variable

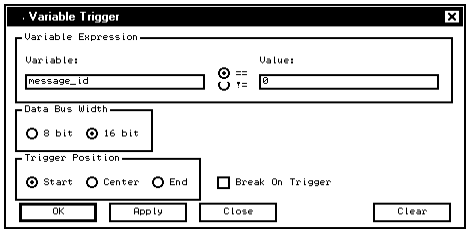
Note

If you use HP B3755A/56A Debug User Interface, the logic analyzer should be connected on LAN to perform the trace function. See "Connect→Logic Analyzer... (ALT, C, L)" in Chapter 5 for details.

To trace accesses to the "message_id" variable:

1. Double-click to highlight "message_id" on the line showing the "PC>>" marker in the Debug window.

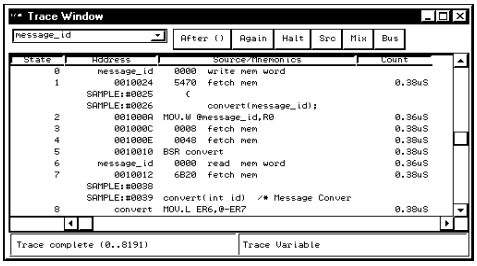
2. From the Trace window, choose the Trace→Variable... (ALT, T, V) command.



3. Enter "0" in the Value: text box.

4. Click the OK button.

The Trace window becomes active and displays the lines that accessed "message_id".



Step 17. Exit the debugger

- From the HP Debug User I/F window, choose the File→Exit (ALT, F, X) command.

  This will end your HP Debug User Interface session.

2

Using the Debugger Interface

Using the Debugger Interface

This chapter contains general information about using the Debugger interface:

- Starting and exiting the debugger.
- Working with debugger windows.
- Using the entry buffer.
- Using action buttons.
- Using command files.

Starting and Exiting the Debugger

This section shows you how:

- To start the debugger.
- To exit the debugger.

To start the Debugger

For Workstations:

1 Enter the following command and press the Return key:

```
$ netrap &
```

Use the following options to accelerates the start-up operation:

-a	Specify with the IP address for the emulator.
-l	Specify with the IP address for the logic analyzer.
-s	Specify with the slot ID indicating the logic analyzer module you wish to use.

The option "-s" is useful if your logic analyzer contains more than one modules. You can use the letters "A" to "J" or "a" to "j" for the slot ID. Starting the Debugger without the "-s" option will default to the "AUTOSELECT" mode, in which modules are searched from the upper slot and the first-detected module will be used.

2 Enter the host name of the emulator to be connected in the Emulator Connection dialog box and click the Connect button.

For PC:

1 Choose B37XXX in the HP Debug User Interface group from the Windows 95 Start menu.

2 Enter the host name of the emulator to be connected in the Emulator Connection dialog box and click the Connect button.

You can accelerate start-up operations by specifying the IP address for the emulator when you install the Debugger, which enables the Debugger to connect to the emulator without opening the Emulator Connection dialog box.

To exit the Debugger

- From the HP Debug User I/F window, choose the File→Exit (ALT, F, X) command.

This will end your Debugger session.

Working with Debugger Windows

This section shows you how:

- To open debugger windows.
- To copy window contents to the file.
- To set tabstops in the Debug window.

To open debugger windows

1 From the Debug window, choose the Window (ALT, W) command.

2 Select a window you want to open from the pull-down menu.

You can open the following windows from the Debug window:

- Source window
- Register window
- Memory window
- Variable window
- Peripheral window
- Backtrace window

To enhance your debugging efficiency, you can open multiple instances for each type of window. For example, you can analyze execution of a program by monitoring several variables at one time, or debug a program with source codes displayed for each module.

To copy window contents to the file

- From the window's control menu, choose the File→Copy→Display... (ALT, F, P, D) command.

  This command copies the information shown in the window to a specified file. Selecting this command invokes the File Selection dialog box, which you use to specify the name of the file.

To set tabstops in the Debug window

1. From the control menu, choose the Settings→Source View... (ALT, S S) command.

   The Source View Settings dialog box appears.

2. Enter the number of columns for tabstops in the Tab Width: box.

3. Click the OK button.

Using the Entry Buffer

Some of the Debugger's windows have an entry buffer, located to the left of the menu bar.

For commands that include "()", such as the Execution→Run→From () command in the Debug window, you must enter a value in the buffer. The entered value is then passed to the command when it is executed.

To display value or string in the entry buffer select it by double-clicking in the Debugger's window. It is easier to double-click the data to specify a value in the edit box than to enter it from the keyboard.

For all the windows and dialog boxes, their entry buffer will have the same copy of the data entered in the edit box of one of the windows and dialog boxes by double-clicking, when they are opened. This capability greatly facilitates command input. For example, when tracing accesses to a variable in a program, double-click a variable name in one of the Debugger's windows, choose the Trace→Variable... (ALT, T, V) command in the Trace window, specify a value, and choose the OK button. In this case, you do not need to enter a variable name in the edit box in the Variable Trigger dialog box.

History Function

In the Debugger's windows and dialog boxes, the entry buffers having a ▼ at the right side support the history function, which contains values previously entered.

When specifying a value in the entry buffer, clicking the ▼ displays a pull-down menu showing values that have been specified.

Using pull-down menus to specify a value eliminates the need to re-enter the same values and avoids typing errors.

Using Action Buttons

Some of the Debugger's windows have action buttons, located to the right of the entry buffer.

Using action buttons enables you to execute a command with a click of the mouse button instead of choosing the command from the menu.

For example, in the Debug window, you can display a source code (or a mnemonic code), starting from the address specified in the entry buffer, by pressing the Disp () action button. Without the action button, you would have to choose the Display command from the control menu and then choose the Program At () command.

By default, action buttons are defined for common debugging operations. In some windows, you can define action buttons to customize operations.

See Also

"Customizing the Action Buttons" in Chapter 10, "Customizing the HP Debug User Interface"

Using Command Files

This section shows you how:

- To create a command file.
- To execute a command file.

A command file is an ASCII text file containing one or more debugger commands. Executing this file enables you to perform routine tasks or multiple commands as a batch process. The command syntax is simple, and you can edit a command file using an ASCII editor. For details about the format of each debugger command, see Chapter 7, "Command File Command Summary."

To create a command file

1 From the Debug window's control menu, choose the File→Log→Record... (ALT, F, O, R) command.

The Log Record dialog box appears.

2 Enter the command file name.

3 Choose the commands to be stored in the command file.

4 When you complete all necessary operations, choose the File→Log→Stop (ALT, F, O, S) command.

To execute a command file

1 From the Debug window's control menu, choose the File→Log→Playback... (ALT, F, O, P) command.

The Log Playback dialog box appears.

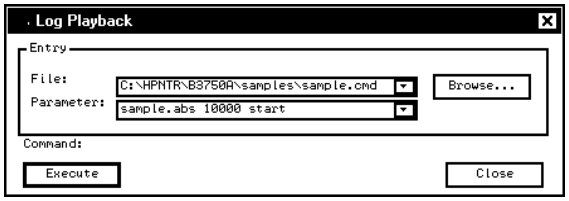
2 Select the command file to be executed.

3 Click the Execute button.

You can execute command files that have been created by logging commands.

Example

Specifying a command file to be executed:



Setting Parameters

You can execute a command file with up to five parameters specified.

To pass a parameter you specify to the command, enter the command in the command file, replacing the parameter field with %1, %2, %3, %4, or %5.

Example

```
file binary %1

bp set %2

run %3
```

The above example shows a command file with three parameters specified. To execute these commands, choose the command file in the Log Playback dialog box, specify "sample.abs 10000 start," using single spaces as the delimiter for the parameters, and click the Execute button. The debugger loads the file sample.abs, sets a breakpoint at address 10000, and runs the program from the start address.

Note

3

Debugging Programs

Debugging Programs

This chapter contains information on loading and debugging programs:

- Loading and displaying programs.
- Running, stepping, and stopping the program.
- Using breakpoints.
- Displaying and editing variables.
- Displaying and editing memory.
- Displaying and editing peripheral registers.
- Displaying and editing registers.
- Tracing program execution.
- Setting display mode for tracing results.
- Setting up custom trace specifications.
- Saving and loading configurations.

Loading and Displaying Programs

This section shows you how:

- To load user programs.
- To display source files by their names.
- To display source code specifying a heading line.
- To display source code from the current program counter.
- To display source code/mnemonics only.
- To display source code mixed with assembly instructions.
- To highlight source code.
- To specify source file directories.

To load user programs

1 From the Debug window's control menu, choose the File→Load→Program... (ALT, F, L, P) command.

The Object File Download dialog box appears.

2 Select the file to be loaded.

3 Specify options according to the attributes of the file.

4 Click the OK button to load the program.

The dialog box will automatically close when the program is loaded.

Clicking the Apply button instead of the OK button will load the program without closing the dialog box. This function is useful when you wish to add several programs or symbol information files. When you complete loading, click the Close button to close the dialog box.

With PC version, you can use the Load action button in the HP Debug User I/F window as the short cut of the File→Load→Program... (ALT, F, L, P) command.

To display source files by their names

1 From the Debug window's control menu, choose the Display→Source Files... (ALT, D, S) command.

The Source Search dialog box appears.

2 Select the file you wish to display.

3 Click the OK button.

This command is also available with the Source window.

Note For some language tools, the contents of assembly source files cannot be displayed.

To display source code specifying a heading line

1 Enter the address you want to display in the entry buffer of the Debug window.

2 Click the Disp () action button in the Debug window.

Or:

Choose the Display→Program At () (ALT, D, A) command from the Debug window's control menu.

For this command, the entry buffer accepts symbol in the source code as well as an address.
This command is also available with the Source window.

To display source code from the current program counter

- Click the Disp PC action button in the Debug window.

  Or:

- Choose the Display→Program At PC (ALT, D, P) command from the Debug window's control menu.

  Use this command to display the source code from the position of the current program counter.
  This command is also available with the Source window.

To display source code or mnemonics only

- From the Debug window's control menu, choose the Settings→Display Mode→Source Only (ALT, S, D, S) command to display only source code.

- To display only mnemonics, choose the Settings→Display Mode→Mnemonics Only (ALT, S, D, N) command instead.

  The Debug window has three display modes: C source-only mode, mnemonics-only mode, and C source/mnemonic mixed mode.
  In the C source-only mode, source code appears with line numbers.
  These commands are also available with the Source window, which provides Src and Mne action buttons for faster access.

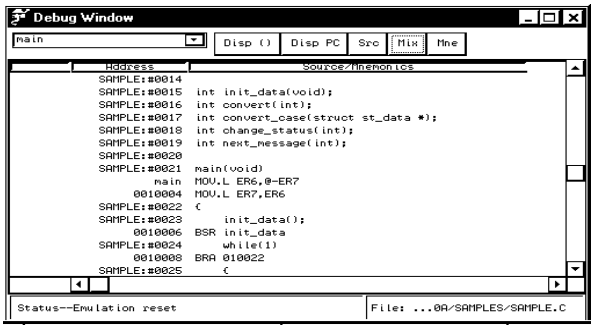
To display source code mixed with assembly instructions

- From the Debug window's control menu, choose the Settings→Display Mode→Mixed (ALT, S, D, M) command.

  In this display mode, the Debug window contains the address, and disassembled instruction mnemonics intermixed with the C source lines. This command is also available with the Source window, which provides a Mix action button for faster access.

Example

Source/mnemonic mixed mode display:



To highlight source code

- From the Debug window's control menu, choose the Settings→Display Mode→Highlight Source Lines (ALT, S, D, H) command.

  This command enables you to show the source code in a different color than the mnemonics. You may want to use this command when you select the C source/mnemonic mixed mode.

  You can specify the color for highlighting (WS version only). See "Customizing the Debugger Interface's Appearance" in Chapter 10 for details.

  This command is also available with the Source window and the Trace window.

To specify source file directories

If the source files associated with the loaded object file are in different directories than the object file, you must identify the directories in which the source files can be found. You can specify as many directories as you use for the program.

1 From the Debug window's control menu, choose the Settings→Source View... (ALT, S, S) command.

The Source View Settings dialog box appears. The list box in the dialog box contains previously specified directories.

2 Enter the directory name in the Source Path: text box.

3 Click the Insert button.

4 If you have a directory that is no longer in use, specify it in the list box and click the Delete button.

5 Click the Close button to close the Source View Settings dialog box.

Running, Stepping, and Stopping the Program

This section shows you how:

- To run the program from the current program counter.
- To run the program from a specified address.
- To run the program from the start address.
- To run the program from target reset.
- To run the program to a specified address.
- To run the program until the current function returns.
- To step a single line or instruction from the current program counter.
- To step a single line or instruction from a specified address.
- To step a single line or instruction from the start address.
- To step over a function.
- To stop program execution.
- To reset the processor.

To run the program from the current program counter

For Workstations:

- Click the Continue action button in the Debug window.

Or:

- Choose the Execution→Run→From PC (ALT, E, R, P) command from the Debug window's control menu.

For PC:

- Click the Continue action button in the HP Debug User I/F window.

Or:

- Choose the Execution→Run→From PC (ALT, E, R, P) command from the Debug window's control menu.

To run the program from a specified address

1 Specify an address in the entry buffer in the Debug window.

2 Choose the Execution→Run→From () (ALT, E, R, F) command from the Debug window's control menu.

This command executes the user program starting from the address specified in the entry buffer.

To run the program from the start address

- Choose the Execution→Run→From Start Address (ALT, E, R, S) command from the Debug window's control menu.

  This command executes the user program from the start address defined in the object file.

To run the program from target reset

- Choose the Execution→Run→From Reset (ALT, E, R, R) command from the Debug window's control menu.

  This command causes the emulator to wait for a RESET signal from the target system. The RESET signal begins executing from the reset vector.

To run the program to a specified address

1. Specify an address in the entry buffer in the Debug window.
2. Choose the Execution→Run→Until () (ALT, E, R, U) command from the Debug window's control menu.

   If the selected source line is not reached within the time (milliseconds) specified by stepTimeout in the initialization file (.netrap.ini for the WS version, or netrap.ini for the PC version), a message box appears telling you that the stepping is aborted and the emulator returns to "running in monitor" status.

To run the program until the current function return s

- **Choose the Execution→Run→Return to Caller (ALT, E, R, C) command from the Debug window's control menu.**

  This command executes the user program until the current function returns to its caller.

  Because this command determines where to stop execution based on stack frame data and object file function information, there are restrictions on using this command. See "Execution→Run→Return to Caller (ALT, E, R, C)" in Chapter 4 for details.

To step a single line or instruction from the current program counter

For Workstations:

- Click the Step action button in the Debug window.

Or:

- Choose the Execution→Step→From PC (ALT, E, S, P) command from the Debug window's control menu.

For PC:

- Click the Step action button in the HP Debug User I/F window.

Or:

- Choose the Execution→Step→From PC (ALT, E, S, P) command from the Debug window's control menu.

To step a single line or instruction from a specified address

1 Specify an address in the entry buffer in the Debug window.

2 Choose the Execution→Step→From () (ALT, E, S, F) command from the Debug window's control menu.

A single source line is executed when in the source-only display mode, unless no source is available or an assembly language program is loaded. In these cases, a single assembly language instruction is executed.

When in the source/mnemonic mixed display mode or in the mnemonic-only display mode, a single assembly language instruction is executed.

To step a single line or instruction from the start address

- Choose the Execution→Step→From Start Address (ALT, E, S, S) command from the Debug window's control menu.

This command executes a single source line or a single assembly language instruction from the start address.

To step over a function

For Workstations:

- Click the Over action button in the Debug window.

Or:

- Choose the Execution→Step→Over Procedure Call (ALT, E, S, O) command from the Debug window's control menu.

For PC:

- Click the Over action button in the HP Debug User I/F window.

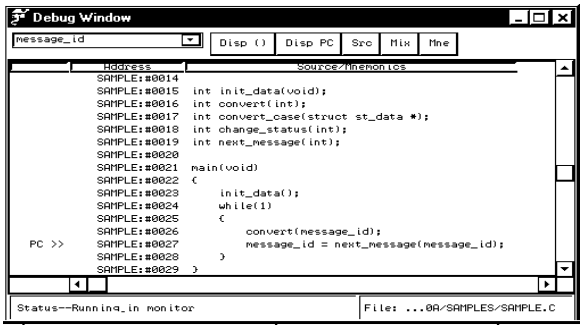
Or:

- Choose the Execution→Step→Over Procedure Call (ALT, E, S, O) command from the Debug window's control menu.

This command executes a single source line or a single assembly language instruction at the current program counter address. If an instruction or source line makes a subroutine or function call, the entire subroutine or function is executed.

Example

When the current program counter is at the line where the function "convert" is called, choosing the Execution→Step→Over Procedure Call (ALT, E, S, O) command steps over the function. Once the function has been stepped over, the program counter indicates the next to the line containing "convert".



Note

Execution may fail in single-stepping source lines containing loop statements such as "while," "for," or "do while" statements.

To stop program execution

For Workstations:

- Click the Break action button in the Debug window.

Or:

- Choose the Execution→Break (ALT, E, B) command from the Debug window's control menu.

For PC:

- Click the Break action button in the HP Debug User I/F window.

Or:

- Choose the Execution→Break (ALT, E, B) command from the Debug window's control menu.

This command stops user program execution and starts monitor program execution.

This command can also be used to break to the monitor when the processor is in "reset" status.

Note

If the clock is not supplied to the emulator, breaking to the monitor is not possible.

To reset the processor

For Workstations:

- Click the Reset action button in the Debug window.

Or:

- Choose the Execution→Reset (ALT, E, T) command from the Debug window's control menu.

For PC:

- Click the Reset action button in the HP Debug User I/F window.

Or:

- Choose the Execution→Reset (ALT, E, T) command from the Debug window's control menu.

This command stops program execution and resets the processor.

Using Breakpoints

This section shows you how:

- To set a breakpoint.
- To list the breakpoints.
- To disable a breakpoint.
- To delete breakpoints.

A breakpoint is an address or symbol you identify in the user program where program execution is to stop. Breakpoints let you look at the state of the user program and the target system at specific points in the program.

Because breakpoints are set by replacing opcodes in the program, you cannot set breakpoints in programs stored in the target system's ROM.

To set a breakpoint

1 Position the cursor on the line where you wish to set a breakpoint and press the right mouse button to display the pop-up menu.

2 Choose the Set Breakpoint command from the pop-up menu.

Or:

1 Select the symbol where you wish to set a breakpoint by double-clicking or dragging the mouse pointer.

2 Choose the Execution→Breakpoint...(ALT, E, P) command from the Debug window's control menu.

The Software Breakpoints dialog box appears. The selected symbol is displayed in the Address: entry box.

3 Click the Set button.

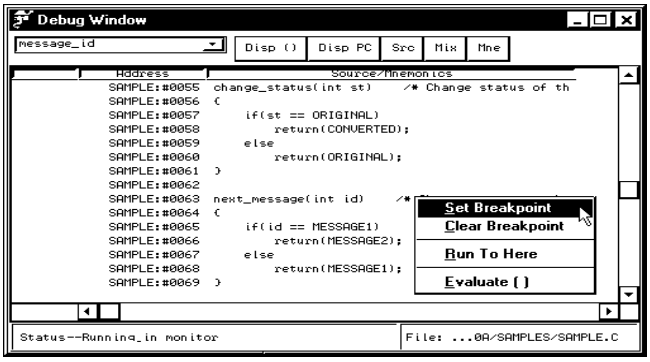
You can find the breakpoint you have set in the Current Breakpoints list box.

4 Click the Close button.

When a breakpoint is reached, program execution stops immediately before executing the instruction or source code line where the breakpoint is set.

Example

To set a breakpoint at the line containing "next_message":



To list the breakpoints

- Choose the Execution→Breakpoints... (ALT, E, P) command.

The Software Breakpoints dialog box appears. The Current Breakpoints list box displays the current breakpoints.

You can enable, disable, or delete breakpoints from this dialog box.

To disable a breakpoint

1. Choose the Execution→Breakpoints... (ALT, E, P) command.

   The Software Breakpoints dialog box appears. The Current Breakpoints list box displays the current breakpoints.

2. Select the breakpoint to be disabled.

3. Click the Disable button.

4. Click the Close button.

You can re-enable a breakpoint in the same manner by choosing the Execution→Breakpoints... (ALT, E, P) command, selecting a disabled breakpoint from the list, and clicking the Enable button.

Breakpoints are set by replacing opcodes in the user program with the processor-specific break instruction. When "Enable Recognition of Software Breakpoints" in the dialog box is set to off, the emulator recognizes break instructions as assembler break instructions instead of software breakpoints.

To delete breakpoints

1. Position the cursor on the line where you wish to set a breakpoint and press the right mouse button to display the pop-up menu.
2. Choose the Clear Breakpoint command from the pop-up menu.

Or:

1. Choose the Execution→Breakpoints... (ALT, E, P) command.

   The Software Breakpoints dialog box appears. The Current Breakpoints list box displays the current breakpoints.
2. Select the breakpoint to be deleted.
3. Click the Clear button.
4. Click the Close button to close the dialog box.

   Clicking the Clear All button deletes all current breakpoints at once.

Displaying and Editing Variables

This section shows you how:

- To display a variable
- To edit a variable

To display a variable

1 Position the mouse pointer over the variable in the window and double-click the left mouse button.

2 Click the right mouse button to display the pop-up menu and choose the Evaluate () command from the menu.

Or:

1 Position the mouse pointer over the variable in the window and double-click the left mouse button.

2 Choose the Window→Variable (ALT, W, V) command.

The Variable window appears. The selected variable and its value are displayed in the window.

If you previously opened the Variable window, you may do the following:

1 Double-click the variable you want to display.

The selected variable name appears in the edit box in the Variable window.

2 Click the Disp () button in the Variable window.

If you want to display the address or value of the pointer variable, choose the Display→Address Of (ALT, D, A) or Display→Contents Of (ALT, D, C) command, respectively, from the Variable window's control menu.

If you want to change the base for display, choose the Settings→Display Base (ALT, S, D) command and then choose one of the following bases from the Variable window's control menu.

Default format	Default (D)
Decimal format	Decimal (C)
Hexadecimal format	Hexadecimal (H)
Floating point format	Float (F)
String format	String (S)

To periodically update the Variable window, choose the Settings→Auto Update (ALT, S, A). See "Settings→Auto Update (ALT, S, A)" in "Variable Window Commands" in Chapter 5 for details on updating options.

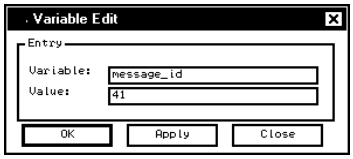
To change the displayed variable, enter the variable name in the entry buffer in the Variable window, then press the Disp () action button. To re-display the previously displayed variable, click the ▼ to the right of the entry buffer to open the pull-down menu and select it from the menu for faster access.

To edit a variable

1. Position the mouse pointer over the variable in the Debug window and double-click the left mouse button.
2. Choose the Window→Variable (ALT, W, V) command from the Debug window's control menu.

   The Variable window appears. The selected variable and its value are displayed in the window.
3. Choose the Modify→Variable... (ALT, M, V) command from the Variable window's control menu.

   The Variable Edit dialog box appears.
4. Type the desired value in the Value: text box.



5. Click the Apply button.
6. Click the Close button.

   You can click the OK button to change the variable value and close the Variable Edit dialog box at the same time.

Displaying and Editing Memory

This section shows you how:

- To display memory.
- To edit memory.
- To fill memory.

To display memory

1 Double-click the starting address or symbol from which the memory contents are displayed.

2 Choose the Window→Memory (ALT, W, M) command from the Debug window's control menu.

The Memory window appears. The memory contents from the selected address or symbol are displayed in the window.

If you previously opened the Memory window, you may do the following:

1 Double-click the symbol you want to display.

The selected symbol appears in the entry buffer in the Memory window.

2 Click the Disp () button in the Memory window.

To specify the data size for display, choose Display (ALT, D) from the Memory window's control menu to open the pull-down menu, then choose one of the following data sizes from the menu:

- Block Format

8-bit format	Display→Blocked→Byte (ALT, D, B, B)
16-bit format	Display→Blocked→Word (ALT, D, B, W)
32-bit format	Display→Blocked→Long (ALT, D, B, L)

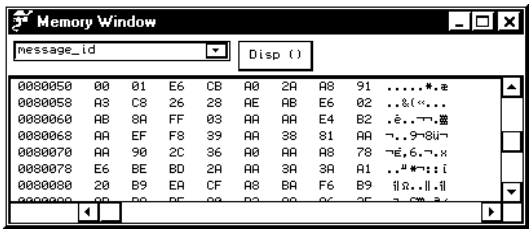
- Absolute Format

8-bit format	Display→Absolute→Byte (ALT, D, A, B)
16-bit format	Display→Absolute→Word (ALT, D, A, W)
32-bit format	Display→Absolute→Long (ALT, D, A, L)
Floating-point format	Display→Absolute→Float (ALT, D, A, F)

To periodically update the Memory window, choose the Settings→Auto Update (ALT, S, A). See "Settings→Auto Update (ALT, S, A)" in "Memory Window Commands" in Chapter 5, for details on updating options.

Example

Memory display in byte format:



To edit memory

1 Position the mouse pointer over the symbol in the Debug window and double-click the left mouse button.

2 Choose the Window→Memory (ALT, W, M) command from the Debug window's control menu.

The Memory window appears. The memory contents starting from the selected symbol are displayed in the window.

3 Choose the Modify→Memory... (ALT, M, M) command from the Memory window's control menu.

The Memory Modification dialog box appears.

4 Type the desired value in the Data: text box and specify the data size in the Size: text box.

5 Click the Apply button.

6 Click the Close button.

You can click the OK button to change the variable value and close the Memory Modification dialog box at the same time.

Memory can be also edited directly;position the mouse pointer on the value in the Memory window, enter the desired value, and press the Return key.

Editing the contents of target system memory temporarily interrupts user program execution. You cannot modify the contents of target memory while the emulator is running the user program and monitor intrusion is disallowed.

To fill memory

1 From the Memory window's control menu, choose the Modify→Memory... (ALT, M, M) command.

The Memory Modification dialog box appears.

2 Enter the desired address range in the Address: text box.

The format is <*Start_address*>..<*End_address*>.

3 Select one of the Size options.

4 Click the Apply button.

You can click the OK button to fill the memory and close the Memory Modification dialog box at the same time.

Displaying and Editing Peripheral Registers

This section shows you how:

- To display the peripheral registers.
- To edit the peripheral registers.

To display peripheral registers

1 Choose the Window→Peripheral (ALT, W, P) command from the Debug window's control menu.

The Peripheral window appears.

2 Choose the Display command from the Peripheral window's control menu and select the register classes you want to display.

To periodically update the Peripheral window display, choose the Settings→Auto Update (ALT, S, A) command from the Peripheral window's control menu. See "Settings→Auto Update (ALT, S, A)" in "Peripheral Window Commands" in Chapter 5 for details.

For details on the classes that can be displayed in the window, see *HP Debug User Interface User's Guide* specific to your target system and emulator.

To edit peripheral registers

1 Choose the Window→Peripheral (ALT, W, P) command from the Debug window's control menu.

The Peripheral window appears.

2 Choose the Display command from the Peripheral window's control menu and select the register classes you want to edit.

3 Double-click the value to be changed.

4 Use the keyboard to enter a new value.

5 Press the Return key.

Modifying register contents temporarily interrupts program execution. You cannot modify register contents while the user program is running and monitor intrusion is disallowed.

Displaying and Editing Registers

This section shows you how:

- To display registers.
- To edit registers.

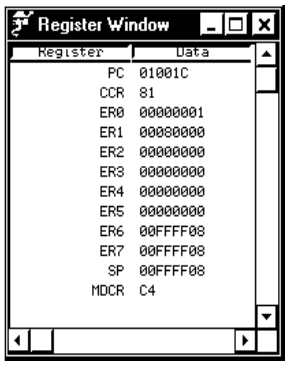
To display registers

- Choose the Window→Register (ALT, W, R) command from the Debug window's control menu.

To periodically update the Register window display, choose the Settings→Auto Update (ALT, S, A) command from the Register window's control menu. See "Settings→Auto Update (ALT, S, A)" in Chapter 5, "Register Window Commands," for details.

Example

Register contents displayed in the Register window:



To edit registers

1. Choose the Window→Register (ALT, W, R) command from the Debug window's control menu.
2. Double-click the value to be changed.
3. Use the keyboard to enter a new value.
4. Press the Return key.

Modifying register contents temporarily interrupts program execution. You cannot modify register contents while the user program is running and monitor intrusion is disallowed.

Note that register values are not actually changed until the Return key is pressed.

Tracing Program Execution

This section shows you how:

- The tracing function works.
  - trace window contents
  - notes for specifying symbols
- To trace accesses to every state.
- To start tracing from a specified address.
- To trace including a specified address in the center of the trace list.
- To end tracing at a specified address.
- To trace accesses to a specified address.
- To trace accesses to a specified address and break.
- To trigger tracing on an access to a specified variable.
- To trace until the command is halted.
- To stop a running trace.
- To repeat the last trace.
- To search trigger/string in the trace list.
  - Search for trigger
  - Search for strings
- To specify the trace clock.

Note

If you use HP B3755A/56A Debug User Interface, the logic analyzer should be connected on LAN to perform the trace function. See "Connect→Logic Analyzer... (ALT, C, L)" in Chapter 5 for details.

How the tracing function works

When you trace program execution, the analyzer captures microprocessor address bus, data bus, and control signal values at each clock cycle. The values captured for one clock cycle are collectively called a state. A trace is a collection of these states stored in analyzer memory (also called trace memory).

The trigger condition tells the analyzer when to store states in trace memory. The trigger position specifies whether states are stored before, after, or about the state that satisfies the trigger condition.

The store condition limits the kinds of states that are stored in trace memory.

When the states stored are limited by the store condition, states that satisfy the prestore condition may be stored when they occur before the states that satisfy the store condition.

Note

If the logic analyzer is integrated as a tracer, which your emulation system does not have, the prestore function is not available.

After a captured state satisfies the trigger condition, a trace is completed when trace memory is filled with states that satisfy the store and prestore conditions.

Trace Window Contents

When traces are completed, the trace results are automatically displayed in the Trace window.

Each line in the trace shows the trace buffer state number, the module name + line number or the function name + offset, the source code, and the count information for the state (relative time to the other lines, by default).

When bus cycles are displayed, the address, data, and disassembled instruction or bus cycle status mnemonics are shown.

You can choose the display mode for the Trace window for your own purpose. See "Setting Display Mode for Tracing Results".

Notes for specifying symbols

When you use symbols to specify trace conditions, you must pay attention to whether the Debugger regards the specified symbols as an address range or an address value.

The Debugger recognizes symbols defined for a command containing "()" in their name as an address range. For example, specifying "main" in the entry buffer and choosing the Trace→Only () command traces the *whole* main () function.

To specify a symbol that may be considered an address range as an explicit address value, add "+0" (zero) to the end of the symbol (for example, "main+0").

To trace accesses to every state

- Choose the Trace→Everything (ALT, T, E) command from the Trace window's control menu.

  This command with continuously trace all states. It does not set the trigger condition and starts tracing immediately after the command is given.

To start tracing from a specified address

1 Specify an address in the Trace window's entry buffer.

2 Click the After () action button in the Trace window.

Or:

1 Specify an address in the Trace window's entry buffer.

2 Choose the Trace→After () (ALT, T, F) command.

This command sets the trace condition so that the instruction specified by the address in the entry buffer is located at the beginning of the trace list. When the trace conditions are met, the emulator begins storing trace data.

Note

The Debugger recognizes symbols defined for a command containing "()" in their name as an address range.
To specify a symbol that may be considered an address range as an explicit address value, add "+0" (zero) to the end of the symbol (for example, "main+0").

To trace including a specified address in the center of the trace list

1. Specify an address in the Trace window's entry buffer.
2. Choose the Trace→About () (ALT, T, A) command.

   This command sets the trace condition so that the instruction specified by the address in the entry buffer is located at the center of the trace list.

Note

The Debugger recognizes symbols defined for a command containing "()" in their name as an address range.
To specify a symbol that may be considered an address range as an explicit address value, add "+0" (zero) to the end of the symbol (for example, "main+0").

To end tracing at a specified address

1. Specify an address in the Trace window's entry buffer.
2. Choose the Trace→Before () (ALT, T, B) command.

   This command sets the trace condition so that the instruction specified by the address in the entry buffer is located at the end of the trace list. The emulator always stores trace data and stops storing when the trace conditions are met.

Note

The Debugger recognizes symbols defined for a command containing "()" in their name as an address range.
To specify a symbol that may be considered an address range as an explicit address value, add "+0" (zero) to the end of the symbol (for example, "main+0").

To trace accesses to a specified address

1. Specify a symbol in the Trace window's entry buffer.
2. Choose the Trace→Only () (ALT, T, O) command.

   This command traces accesses to the range of memory specified by the symbol in the entry buffer enabling you to create a trace list for particular tasks.

To trace accesses to a specified address and break

1. Specify an address in the Trace window's entry buffer.
2. Choose the Trace→Until () (ALT, T, U) command.

   This command sets the trace condition so that the instruction specified by the address in the entry buffer ends the trace. Program execution stops when the condition is met.

Note

The Debugger recognizes symbols defined for a command containing "()" in their name as an address range.
To specify a symbol that may be considered an address range as an explicit address value, add "+0" (zero) to the end of the symbol (for example, "main+0").

To trigger tracing on an access to a specified variable

1 Double-click the variable.

2 Choose the Trace→Variable... (ALT, T, V) command from the Trace window's control menu.

The Variable Trigger dialog box appears.
The Variable: text box contains the variable name selected by double-clicking.

3 Enter a value in the Value: text box.

4 Choose "==" to trigger the analyzer when a write operation occurs to the variable with the specified value. Choose "!=" to trigger the analyzer when a write operation occurs with values other than the specified value.

5 Specify the data bus width for the variable address.

6 Select the trigger position.

7 Select the Break On Trigger check box to stop program execution on the occurrence of a trigger.

8 Click the OK button.

Clicking the Apply button instead of the OK button allows you to execute tracing with the dialog box open. This option is useful when you want to change conditions and initiate successive traces. To close the dialog box, click the Close button.

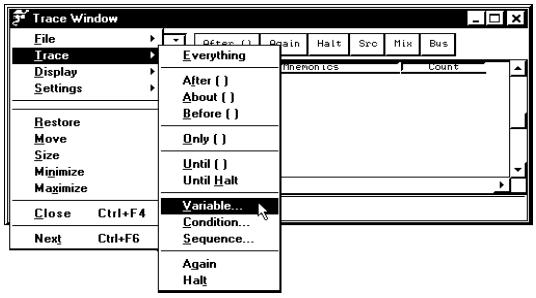
The Trace→Variable... (ALT, T, V) command executes tracing triggered when the specified value is written in the specified global variable. Selecting the Break On Trigger check box stops program execution on the occurrence of a trigger.

Example

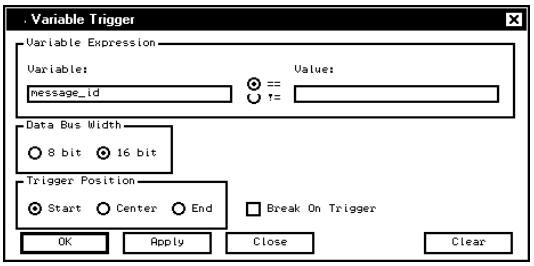
To stop execution when the variable "message_id" becomes "0":

Double-click "message_id".

Choose the Trace→Variable... (ALT, T, V) command.



Enter "0" in the Value: text box and choose "==".
Choose the bus width. Choose "End" for "Trigger Position" and select the Break On Trigger check box.



Click the OK button.

When the value of the variable "message_id" becomes "0," program execution stops. The Trace window displays the trace result.

To trace until the command is halted

1. To start the trace, choose the Trace→Until Halt (ALT, T, H) command.
2. When you are ready to stop the trace, choose the Trace→Halt (ALT, T, T) command.

This command is useful for tracing program execution that leads to a processor halt or to a break to the monitor.

To stop a running trace

- Choose the Trace→Halt (ALT, T, T) command.

The command is used to:

Stop a trace initiated with the Trace→Until Halt (ALT, T, H) command.

Force termination of a trace that cannot be completed due to absence of the specified state.

Stop a trace before the trace buffer becomes full.

To repeat the last trace

- Choose the Trace→Again (ALT, T, G) command.

The Trace→Again (ALT, T, G) command traces program execution using the last trace specifications stored.

To search trigger/string in the trace list

The HP Debug User Interface enables you to search the trace result for information you need.

Search for trigger

To search the trace result for the trigger positions:

- Choose the Display→Trigger (ALT, D, T) command from the Trace window's control menu.

Search for strings

To search the trace result for a specified string:

- Choose the Display→Find... (ALT, D, F) command from the Trace window's control menu.

The search starts from the current cursor position in the Trace window. You can set the search direction using the Direction option button in the Find String dialog box. Marking the Match Case check box enables case-sensitive search.

A value or string can be selected (in other words, copied in the entry buffer) from another window before choosing the command; it will automatically appear in the dialog box that is opened.

To specify the trace clock

1 From the Trace window's control menu, choose Settings→Trace Clocks.

2 Choose one of Very Fast, Fast, and Slow from the pull-down menu.

Specify the trace clock for the clock frequency of your emulation processor, as shown in the following list.

Emulation Processor Clock Frequency	Trace Clock to Be Specified
26 MHz or less	Choose Very Fast. The trace clock is set to the fastest speed.
20 MHz or less	Choose Fast. The trace clock is set to the fast speed.
16 MHz or less	Choose Slow. The trace clock is set to the slow speed.

You can select the trace clock only if your emulator is the HP 64704/706A. If your emulator is the HP 64794A/C/D, this command is displayed in gray and cannot be chosen.

Note

If you use the HP 64704/706A analyzer board and set the trace clock to Fast or Very Fast, states are counted instead of time. If you want to measure trace time with a 16-MHz or faster clock, you must use the HP 64794A/C/D analyzer board.

Setting Display Mode for Tracing Results

This section shows you how:

- To display bus cycles.
- To specify display mode for the trace count.
- To display symbol information in the trace list.
- To display the trace list with function names.
- To display the trace list with line numbers.
- To highlight source code in the trace list.

To display bus cycles

- Click the Bus action button or Mix action button in the Trace window.

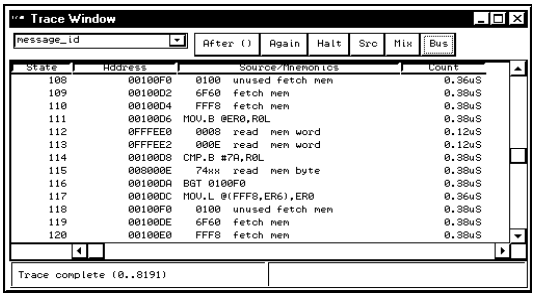
  Or:

- From the Trace window's control menu, choose the Settings→Display Mode→Bus Cycles Only (ALT, S, D, B) or Settings→Display Mode→Mixed (ALT, S, D, M) command.

  The Settings→Display Mode→Mixed (ALT, S, D, M) command displays the bus cycles associated with each of the source lines. The Settings→Display Mode→Bus Cycles Only (ALT, S, D, B) command displays only bus cycles. To hide the bus cycles, click the Src action button in the Trace window, or choose the Settings→Display Mode→Source Only (ALT, S, D, S) command from the Trace window's control menu.

Example

Bus cycles displayed in the trace window.



To specify display mode for the trace count

- From the Trace window, choose the Settings→Trace Count→Absolute (ALT, S, C, A) command or the Settings→Trace Count→Relative (ALT, S, C, R) command.

  To count in relative mode, choose the Settings→Trace Count→Relative (ALT, S, C, R) command. The count for interval between the current and previous states is displayed.

  To count in absolute mode, choose the Settings→Trace Count→Absolute (ALT, S, C, A) command. The total count obtained since the trigger of the trace is displayed.

Note If you use the HP 64704/706A analyzer board and set the trace clock to Fast or Very Fast, states are counted instead of time. If you want to measure trace time with a 16-MHz or faster clock, you must use the HP 64794A/C/D analyzer board.

Note If you use HP B3755A/56A Debug User Interface, maximum time count between states is 34 seconds.

To display symbol information in the trace list

- Choose the Settings→Display Mode→Symbols (ALT, S, D, Y) command from the Trace window's control menu.

  Symbol information is displayed in the trace list.

To display the trace list with function names

- Choose the Settings→Display Mode→Function Names (ALT, S, D, F) command from the Trace window's control menu.

  Function names are displayed in the trace result. This command causes line numbers to disappear.

To display the trace list with line numbers

- Choose the Settings→Display Mode→Line Numbers (ALT, S, D, L) command from the Trace window's control menu.

  Line numbers are displayed in the trace result. This command causes function names to disappear.

To highlight source code in the trace list

- Choose the Settings→Display Mode→Highlight Source Lines (ALT, S, D, H) command from the Trace window's control menu.

  Source lines are highlighted.

  This command is available only when source code is displayed in the trace result.

  The HP Debug User Interface allows you to change the color of highlighted source lines (WS version only). See "Customizing the Debugger Interface's Appearance" in Chapter 10, "Customizing the HP Debug User Interface," for details.

Setting Up Custom Trace Specifications

This section shows you how:

- To set up an appropriate trace specification.
- To set up a Condition trace specification.
- To set up a Sequence trace specification.
- To store and load trace specifications.

To set up an appropriate trace specification

When you want to trigger the analyzer on the occurrence of one state, use the Trigger Store Condition dialog box to set up the trace specification.
When you want to trigger the analyzer on a sequence of more than two states, use the Sequence dialog box to set up the trace specification.

To set up a Condition trace specification

The HP Debug User Interface's Condition trace function is useful when you want to trace states that satisfy condition B after condition A is satisfied. Condition A is called the *trigger condition* and condition B is called the *store condition*.

1 Choose the Trace→Condition... (ALT, T, C) command from the Trace window's control menu.

The Trace Trigger Store Condition dialog box appears.

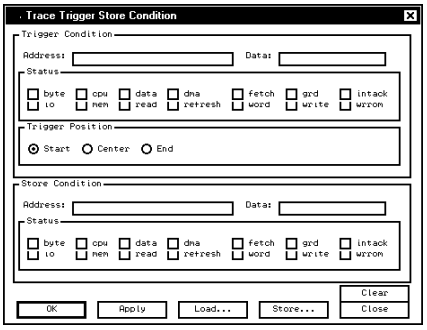
2 Specify the *trigger condition* using the Address: and Data: text boxes and the Status check box within the Trace Condition group box.

3 Specify the *trigger position* by selecting the trigger start, trigger center, or trigger end option in the Trigger Position group box.

4 Specify the *store condition* using the Address: and Data: text boxes and the Status check box within the Store Condition group box.

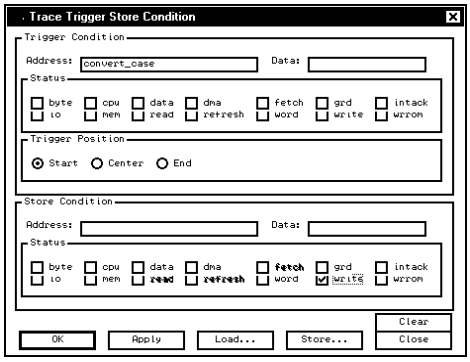
5 Click the OK button to set up the analyzer and start the trace.

You can store the setting specified in the dialog box in a file and load it when you want to use the same or a similar setting.



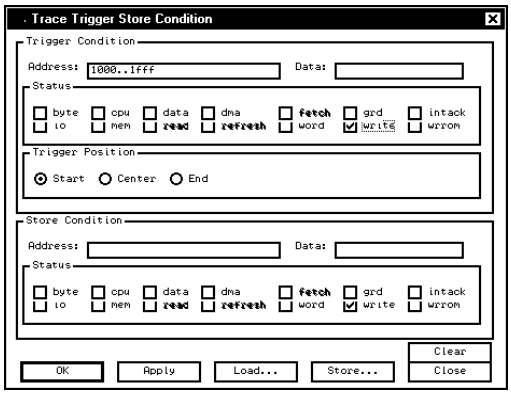
Example

To set access to the "convert_case" function as the trigger condition and store only states with write status before and after the trigger condition:



Example

To specify the trigger condition as any address where write cycles occur in the range 1000h through 1fffh:



To set up a Sequence trace specification

Sequence trace specifications let you trigger the analyzer on a sequence of several captured states.

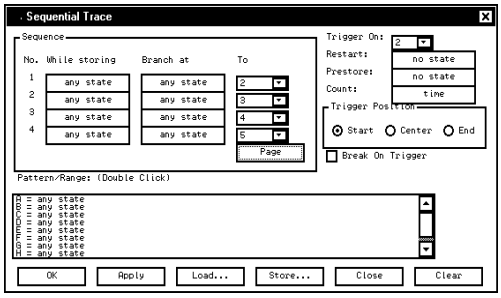
The HP Debug User Interface supports eight sequence levels. When a trace is started, the first sequence level is active. Entry into one of the other sequence levels (you specify which) will trigger the analyzer. Each level lets you specify one condition that, when satisfied by a captured state, will cause branches to other levels.

1. Choose the Trace→Sequence... (ALT, T, S) command from the Trace window's control menu.
2. Specify While storing, Branch at, and To for each sequence level you will use.
3. Specify which sequence level to trigger on.

   The analyzer triggers on entry to the specified level. Therefore, the condition that causes a branch to the specified level actually triggers the analyzer.
4. Specify the restart, prestore, and count conditions.
5. Choose the resources used for the condition specified above from the Pattern/Range: list box by double-clicking and specifying the address range, data range, or status to be traced and the trace pattern.
6. Specify the trigger position by selecting the trigger start, trigger center, or trigger end option.
7. If you want emulator execution to break to the monitor when the trigger condition occurs, select the Break On Trigger check box.
8. Click the OK button to set up the analyzer and start the trace.

The Trace→Sequence... (ALT, T, S) command calls the Sequential Trace dialog box, where you make the following trace specifications:

You can store the setting specified in the dialog box in a file and load it when you want to use the same or a similar setting.



Selecting While storing, Branch at, Restart, Prestore, or Count options opens the Condition dialog box.

To set the conditions using the Condition dialog box:

- Specify the condition type.

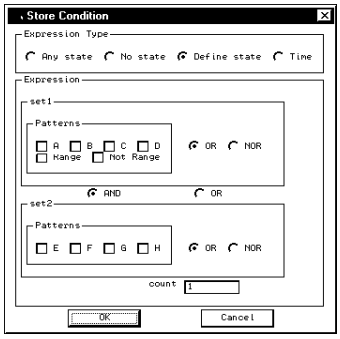
  Choose options in the Expression Type group box.

Any state	Any states satisfy the condition you set.
No state	No states satisfy the condition you set.
Define state	Combinations of trace patterns defined in the Expression group box satisfy the condition you set.
Time	Counts trace in time. This option is available only when you select Count.

When you choose Define state, follow the next steps:

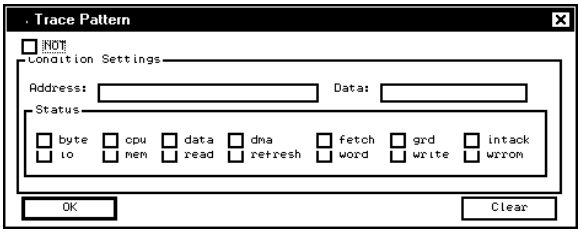
- Specify the condition.

  Choose Trace patterns "A" through "H", "Range", and "Not Range". You can specify two sets of patterns, and resources within a set may be combined using the OR and NOR logical operators. Resources in the two sets may be combined using the OR and AND logical operators.

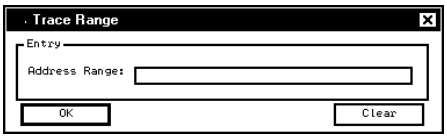


  The address or data range and trace pattern resources are defined by double-clicking on the resource name in the Pattern/Range list box.

  If you double-click on a pattern name in the Pattern/Range list box, the Trace Pattern dialog box appears to let you specify address, data, and status values. By selecting the NOT check box, you can specify all states other than those identified by the address, data, and status values.

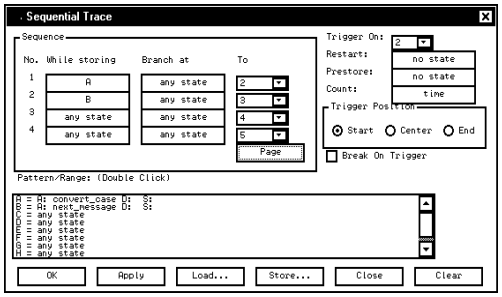


If you double-click on the range resource in the Pattern/Range list box, the Trace Range dialog box appears to let you select and address range for the specified range and enter the start and end addresses in the range. Enter the range with the format:<*Start_address*>..<*End_address*>.



Example

To specify execution of "convert_case" and "next_message" as the trigger sequence:



To store and load trace specifications

You can store trace specifications specified in the Trace Trigger Store Condition dialog box and Sequential Trace dialog box in a file and load them when you want to use the same or similar settings.

This function enables you to repeatedly use complicated trace specifications easily.

To store trace specifications:

The dialog box for storing the trace specification is assumed to be open.

1 Click the Store... button.

2 Specify the name of the trace specification file.

3 Click the OK button.

To load trace specifications:

The dialog box for loading the trace specification is assumed to be open.

1 Click the Load... button.

2 Select the desired trace specification file.

3 Click the OK button.

Saving and Loading Configurations

This section shows you how:

- To save the current emulator configuration.
- To load an emulator configuration.

To save the current emulator configuration

1 From the Debug window, choose the File→Store→Configuration... (ALT, F, S, C) command.

The Configuration File Selection dialog box appears. The displayed directory is the HP Debug User Interface start-up directory.

2 Choose the directory in which you want to store the file from the Directories: list box and double-click it.

3 Specify a file name in the Configuration File Name: box.

The ".cfg" extension is automatically attached to the file name.

4 Click the OK button.

To load the emulator configuration

1 From the Debug window, choose the File→Load→Configuration... (ALT, F, L, C) command.

The Configuration File Selection dialog box appears. The displayed directory is the HP Debug User Interface start-up directory. Files having the ".cfg" extension are displayed.

2 Choose the directory that contains the file you want to load from the Directories: list box and double-click it.

As necessary, change the contents of the Filter: box and click the Filter button to adjust the list of files.

3 Choose the file you want to load with the mouse and click the OK button.

Note

4

Debug Window Commands

Debug Window Commands

This chapter describes the following commands:

- File→Load→Configuration... (ALT, F, L, C)
- File→Load→Program... (ALT, F, L, P)
- File→Store→Configuration... (ALT, F, S, C)
- File→Copy→Display... (ALT, F, P, D)
- File→Log→Playback... (ALT, F, O, P)
- File→Log→Record... (ALT, F, O, R)
- File→Log→Stop (ALT, F, O, S)
- Execution→Run→From PC (ALT, E, R, P)
- Execution→Run→From () (ALT, E, R, F)
- Execution→Run→From Start Address (ALT, E, R, S)
- Execution→Run→From Reset (ALT, E, R, R)
- Execution→Run→Until () (ALT, E, R, U)
- Execution→Run→Return to Caller (ALT, E, R, C)
- Execution→Step→From PC (ALT, E, S, P)
- Execution→Step→From () (ALT, E, S, F)
- Execution→Step→From Start Address (ALT, E, S, S)
- Execution→Step→Over Procedure Call (ALT, E, S, O)
- Execution→Break (ALT, E, B)
- Execution→Reset (ALT, E, T)
- Execution→Breakpoints... (ALT, E, P)
- Display→Program At PC (ALT, D, P)
- Display→Program At () (ALT, D, A)
- Display→Source Files... (ALT, D, S)
- Display→Find... (ALT, D, F)
- Window→Source (ALT, W, S)
- Window→Register (ALT, W, R)
- Window→Memory (ALT, W, M)
- Window→Variable (ALT, W, V)
- Window→Peripheral (ALT, W, P)
- Window→Backtrace (ALT, W, B)
- Settings→Display Mode→Source Only (ALT, S, D, S)
- Settings→Display Mode→Mixed (ALT, S, D, M)

- Settings→Display Mode→Mnemonics Only (ALT, S, D, N)
- Settings→Display Mode→Symbols (ALT, S, D, Y)
- Settings→Display Mode→Highlight Source Lines (ALT, S, D, H)
- Settings→Source View... (ALT, S, S)
- Settings→Configuration→Hardware... (ALT, S, C, H)
- Settings→Configuration→Memory Map... (ALT, S, C, M)

File→Load→Configuration... (ALT, F, L, C)

Loads an emulator configuration command file.

This command opens a file selection dialog box from which you select the emulator configuration file.

Emulator configuration command files contain:

- Hardware configuration settings.
- Memory map configuration settings.

Command File Command

No command.

See Also

"Saving and Loading Configurations" in Chapter 3, "Debugging Program"

File→Load→Program... (ALT, F, L, P)

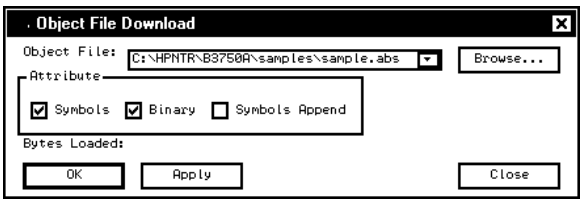
Loads the specified object file and symbolic information into the debugger.

Program code is loaded into the emulation memory or RAM on the target system.

For available object files and the software development tools that generate them, see *HP Debug User Interface User's Guide* specific to your target system and emulator.

Object File Download Dialog Box

Choosing the File→Load→Program... (ALT, F, L, P) command opens the following dialog box:



Object File	Lets you specify the object file to be loaded.
Browse...	Opens a file selection dialog box from which you can select the object file to be loaded.
Symbols	Loads only the symbolic information. This option is used when programs are already in the debugger memory (for example, when you exit and restart the debugger without turning off power to the target system, or when code resides in the target system's ROM).
Binary	Loads program code but not symbols.
Symbols Append	Appends the symbols from the specified object file to the currently loaded symbols.
Bytes Loaded:	Displays the loaded data in kilobytes.
OK	Starts loading the specified object file.

Apply	Loads program codes or symbol information. Since clicking this button does not close the dialog box, you can successively load other object files and symbol information.
Close	Closes the dialog box without loading the object file.

Command File Commands

```
fil(e) obj(ect) filename
```

Loads the specified object file and symbols into the Debugger.

```
fil(e) sym(bol) filename
```

Loads only the symbolic information from the specified object file.

```
fil(e) bin(ary) filename
```

Loads only the program code from the specified object file.

```
fil(e) obj(ect) filename app(end)
fil(e) sym(bol) filename app(end)
```

Appends the symbol information from the specified object file to the currently loaded symbol information.

See Also

"To load user programs" under "Loading and Displaying Programs" in Chapter 3, "Debugging Programs"

File→Store→Configuration... (ALT, F, S, C)

Saves the current hardware configuration to a command file.

This command opens a file selection dialog box from which you select the emulator configuration file used for storing the configuration.

The following information is saved in the emulator configuration file:

- Hardware configuration settings.
- Memory map configuration settings.

Command File Command

No command.

See Also

"To save the current emulator configuration" under "Saving and Loading Configurations" in Chapter 3, "Debugging Programs"

File→Copy→Display... (ALT, F, P, D)

Copies the current contents of the Debugger window to a file in ASCII format.

This command opens a file selection dialog box from which you select the name of the output file.

Command files are stored as ASCII text files so they can be created or edited with ASCII text editors.

Command File Command

No command.

See Also

"To copy window contents to the file" under "Working with Debugger Windows" in Chapter 2, "Using the Debugger Interface"

File→Log→Playback... (ALT, F, O, P)

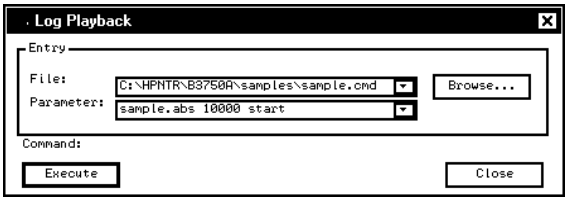
Executes the specified command file.

Command files can be created with the File→Log→Record... (ALT, F, O, R) command.

Command files are stored as ASCII text files so they can be created or edited with ASCII text editors.

Log Playback Dialog Box

Choosing the File→Log→Record... (ALT, F, O, R) command opens the following dialog box:



File	Lets you enter the name of the command file to be executed.
Parameter:	Lets you specify up to five parameters that replace placeholders in the command file.
Browse...	Opens a file selection dialog box from which you can select the command file name.
Command	Shows the command being executed.
Execute	Executes the specified command file.
Close	Closes the dialog box.

Setting Parameters

You can execute a command file with up to five parameters.

To pass a parameter you specify to the command, the command should appear in the command file with the field of the parameter replaced by %1, %2, %3, %4, or %5.

Example

```
file binary %1

bp set %2

run %3
```

The above example shows the command file with three parameters specified. To execute these commands, choose the command file in the Log Playback dialog box, specify "sample.abs 10000 start" using single spaces as the delimiter for the parameters, and click the Execute button. The debugger loads the file sample.abs, sets a breakpoint at address 10000, then runs the program from the start address.

Command File Command

No command.

See Also

"To execute a command file" under "Using Command Files" in Chapter 2, "Using the Debugger Interface"

Chapter 7, "Command File Command Summary"

File→Log→Record... (ALT, F, O, R)

Starts logging of command execution.

This command opens a file selection dialog box from which you can select the destination command log file. If you select a filename that already exists, executed commands are added to the file. You can also create a new command file. Command log files have a ".cmd" extension.

Only command file commands defined for the HP Debug User Interface are recorded.

The File→Log→Stop (ALT, F, O, S) command stops the logging of command execution.

The File→Log→Playback... (ALT, F, O, P) command executes the file created with this command.

Command File Commands

```
log set filename

log on
```

See Also

"To create a command file" under "Using Command Files" in Chapter 2, "Using the Debugger Interface"

Chapter 7, "Command File Command Summary"

File→Log→Stop (ALT, F, O, S)

Stops logging of command execution.

Command File Command

```
log off
```

See Also

"To create a command file" under "Using Command Files" in Chapter 2, "Using the Debugger Interface"
Chapter 7, "Command File Command Summary"

Execution→Run→From PC (ALT, E, R, P)

Runs the program from the current program counter address.
This command drives the processor to "Running user program" status.

Command File Command

`run`

See Also

"To run the program from the current program counter" under "Running, Stepping, and Stopping the Program" in Chapter 3, "Debugging Programs"

Execution→Run→From () (ALT, E, R, F)

Runs the program from the specified address.
This command drives the processor to "Running user program" status.

Command File Command

`run fro(m) address`

See Also

"To run the program from a specified address" under "Running, Stepping, and Stopping the Program" in Chapter 3, "Debugging Programs"

Execution→Run→From Start Address (ALT, E, R, S)

Runs the program from the start address defined in the object file.
This command drives the processor to "Running user program" status.

Command File Command

```
run sta(rt)
```

See Also

"To run the program from the start address" under "Running, Stepping, and Stopping the Program" in Chapter 3, "Debugging Programs"

Execution→Run→From Reset (ALT, E, R, R)

Causes the emulator to wait for a reset signal from the target system. The reset signal begins executing from the reset vector.
This command drives the processor to "Awaiting target reset" status.

Command File Command

```
run res(et)
```

See Also

"To run the program from target reset" under "Running, Stepping, and Stopping the Program" in Chapter 3, "Debugging Programs"

Execution→Run→Until () (ALT, E, R, U)

Runs from the current program counter address up to the specified address. If the specified address is not reached within the time (milliseconds) specified by stepTimeout in the initialization file (.netrap.ini for the WS version, or netrap.ini for the PC version), a message box appears telling you that the stepping is aborted.

Command File Command

`run unt(il) address`

See Also

"To run the program from a specified address" under "Running, Stepping, and Stopping the Program" in Chapter 3, "Debugging Programs"

Execution→Run→Return to Caller (ALT, E, R, C)

Runs the program until the current function returns to its caller.

Because this command determines the address to stop execution based on stack frame data and object file function information, the following restrictions are imposed:

- A function cannot properly return immediately after its entry point because the stack frame for the function has not yet been generated. Use the Execution→Step→From PC command to single-step the function before using this command.
- An assembly language routine cannot properly return, even it follows C function call conventions, because there is no function information in the object file.
- An interrupt function cannot properly return because it uses a stack in a different fashion from standard functions.

Command File Command

```
ret(urn)
```

See Also

"To run the program until the current function returns" under "Running, Stepping, and Stopping the Program" in Chapter 3, "Debugging Programs"

Execution→Step→From PC (ALT, E, S, P)

Executes a single source line or a single assembly language instruction at the current program counter address.

A single source line is executed when in the source-only display mode, unless no source is available or an assembly language program is loaded; in these cases, a single assembly language instruction is executed.

When in the source/mnemonic mixed display mode or in the mnemonic-only display mode, a single assembly language instruction is executed.

Command File Command

`ste(p) count`

See Also

"To step a single line or instruction from the current program counter" under "Running, Stepping, and Stopping the Program" in Chapter 3, "Debugging Programs"

Execution→Step→From () (ALT, E, S, F)

Executes a single source line or a single assembly language instruction from the specified address or the symbol in the entry buffer.

A single source line is executed when in the source-only display mode, unless no source is available or an assembly language program is loaded; in these cases, a single assembly language instruction is executed.

When in the source/mnemonic mixed display mode or in the mnemonic-only display mode, a single assembly language instruction is executed.

Command File Command

`ste(p) count fro(m) address`

See Also

"To step a single line or instruction from a specified address" under "Running, Stepping, and Stopping the Program" in Chapter 3, "Debugging Programs"

Execution→Step→From Start Address (ALT, E, S, S)

Executes a single source line or a single assembly language instruction from the start address defined in the object file.

A single source line is executed when in the source-only display mode, unless no source is available or an assembly language program is loaded; in these cases, a single assembly language instruction is executed.

When in the source/mnemonic mixed display mode or in the mnemonic-only display mode, a single assembly language instruction is executed.

Command File Command

`ste(p) count sta(rt)`

See Also

"To step a single line or instruction from the start address" under "Running, Stepping, and Stopping the Program" in Chapter 3, "Debugging Programs"

Execution→Step→Over Procedure Call (ALT, E, S, O)

Executes a single source line or a single assembly language instruction at the current program counter address. If an instruction or source line makes a subroutine or function call, the entire subroutine or function is executed.

This command is identical to the Execution→Step→From PC (ALT, E, S, P) command except that the entire subroutine or function is executed if an executed source line makes a function call or an executed assembler instruction makes a subroutine call.

Note

Execution may fail in single-stepping source lines containing loop statements such as "while," "for," or "do while" statements.

Command File Command

`ove(r) count`

See Also

"To step over a function" under "Running, Stepping, and Stopping the Program" in Chapter 3, "Debugging Programs"

Execution→Break (ALT, E, B)

Stops user program execution and breaks to the monitor.

This command can also be used to break to the monitor when the processor is in the "Emulation reset" status.

This command drives the processor into "Running in monitor" status.

Note

If the clock is not supplied to the emulator, breaking to the monitor is not possible.

Command File Command

```
bre(ak)
```

See Also

"To stop program execution" under "Running, Stepping, and Stopping the Program" in Chapter 3, "Debugging Programs"

Execution→Reset (ALT, E, T)

Resets the emulation processor.

While the processor is in "Emulation reset" status, you cannot display or modify the contents of target system memory or registers. Before you can display or modify target system memory or processor registers, you must use the Execution→Break (ALT, E, B) command to break to the monitor.

Command File Command

```
res(et)
```

See Also

"To reset the processor" under "Running, Stepping, and Stopping the Program" in Chapter 3, "Debugging Programs"

Execution→Breakpoints... (ALT, E, P)

Sets, lists, or deletes breakpoints.

This command displays the Software Breakpoints dialog box, from which you can set, delete, enable, or disable breakpoints while checking the current breakpoint settings.

Note

You cannot enable or disable breakpoints from the pop-up menu.

The breakpoint marker "*" appears on lines at which breakpoints are set. A set breakpoint remains active until it is deleted.

When a breakpoint is reached, program execution stops immediately before executing the instruction or source code line at which the breakpoint is set.

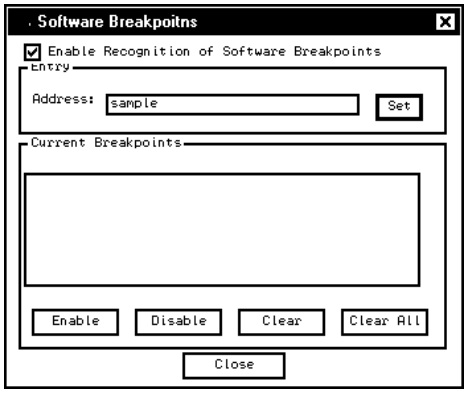
This command replaces the original program opcodes at the specified address with a break instruction. When the emulator detects the break instruction, it breaks to the monitor and restores the original instruction from the memory. When the emulator detects a break instruction that was not inserted as a breakpoint, the emulator breaks to the monitor and shows "Undefined breakpoint at address" message.

Breakpoints cannot be set in programs stored in target system ROM.

This command may cause "*" markers to appear at two or more addresses. This happens when a single instruction is associated with two or more source lines. You can select the mnemonic display mode in the Debug window to verify that the breakpoint is set at a single address.

Software Breakpoints Dialog Box

The Execution→Breakpoint... (ALT, E, P) command opens the following dialog box:



Enable Recognition of Software Breakpoints	Specifies whether to allow the emulator to recognize breakpoints you set. If you set this item to off, the emulator recognizes break instructions not as software breakpoints but as break instructions in the user program.
Address	Lets you specify the symbol or address at which to set a breakpoint.
Set	Sets a breakpoint.
Current Breakpoints	Displays the addresses and line numbers of current breakpoints.
Enable	Enables the selected breakpoint.
Disable	Disables the selected breakpoint.
Clear	Deletes the selected breakpoints from the Current Breakpoints list box.
Clear All	Deletes all the breakpoint from the Current Breakpoints list box.
Close	Closes the dialog box.

Command File Commands

`bp set` *`address`*

Sets a breakpoint.

`bp ena(ble)/dis(able)` *`address`*

Enables/Disables the specified breakpoint.

`bp ena(ble)/dis(able) all`

Enables/Disables all breakpoints.

`bp cle(ar)` *`address`*

Deletes the specified breakpoint.

`bp cle(ar) all`

Deletes all breakpoints.

`mod(e) bp ena(ble)/dis(able)`

Enables/Disables recognition of software breakpoints.

See Also

"Using Breakpoints" in Chapter 3, "Debugging Programs"

Display→Program At PC (ALT, D, P)

Displays the source code starting from the current program counter.

Command File Command

```
dis(play) fro(m) pc
```

See Also

"To display source code from the current program counter" under "Loading and Displaying Programs" in Chapter 3, "Debugging Programs"

Display→Program At () (ALT, D, A)

Displays the source code starting from the specified address. Enter the start address in the entry buffer on the tool bar and choose this command.
You can use symbols to specify addresses.

Command File Command

```
dis(play) fro(m) address
```

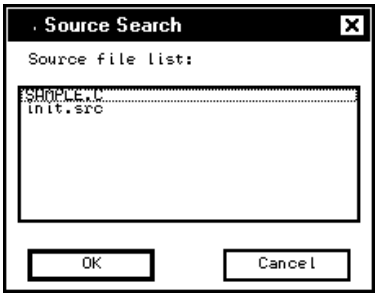
See Also

"To display source code specifying a heading line" under "Loading and Displaying Programs" in Chapter 3, "Debugging Programs"

Display→Source Files... (ALT, D, S)

Displays the specified source file in the Debug window.
This command is disabled before the object file is loaded or when no source is available for the loaded object file.

Source Search Dialog Box



Source file list	Lists source files associated with the loaded object file. You can select the source file to be displayed from this list.
OK	Switches the Debug window contents to the selected source file.
Cancel	Closes the dialog box.

Note

For some language tools, the contents of assembly source files cannot be displayed.

Command File Command

No command.

See Also

"To display source files by their names" under "Loading and Displaying Programs" in Chapter 3, "Debugging Programs"

Display→Find... (ALT, D, F)

Searches for and displays a specified string in the Debug window.

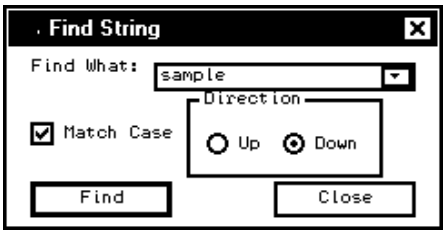
The search starts from the current cursor position in the Debug window. You can set the search direction using the Direction option button in the Find String dialog box. Checking the Match Case check box enables a case-sensitive search.

Before searching for strings using this command, you must set the emulator to source-only display mode.

The string can be selected from another window (that is, copied to the entry buffer) before choosing the command; it will automatically appear in the dialog box that is opened.

Find String Dialog Box

Choosing the Display→Find... (ALT, D, F) command opens the following dialog box:



Find What	Lets you enter the string to be searched in the Debug window.
Match Case	Enables or disables case matching.
Up	Specifies that the search be from the current cursor position backward.
Down	Specifies that the search be from the current cursor position forward.
Find	Searches for the string.
Close	Closes the dialog box.

Command File Command

No command.

Window→Source (ALT, W, S)

Opens the Source window.

Command File Command

```
dis(play) sou(rce)
```

Window→Register (ALT, W, R)

Opens the Register window.

Command File Command

```
dis(play) reg(ister)
```

See Also

"Displaying and Editing Registers" in Chapter 3, "Debugging Programs"

Window→Memory (ALT, W, M)

Opens the Memory window.

Command File Command

```
dis(play) mem(ory) address
```

See Also

"Displaying and Editing Memory" in Chapter 3, "Debugging Programs"

Window→Variable (ALT, W, V)

Opens the Variable window.

Command File Command

`dis(play) var(iable) variable`

See Also

"Displaying and Editing Variables" in Chapter 3, "Debugging Programs"

Window→Peripheral (ALT, W, P)

Opens the Peripheral window.

Command File Command

`dis(play) per(ipheral)`

See Also

"Displaying and Editing Peripheral Registers" in Chapter 3, "Debugging Programs"

Window→Backtrace (ALT, W, B)

Opens the Backtrace window.

Command File Command

```
dis(play) bac(ktrace)
```

See Also

"Backtrace Window Commands" in Chapter 5, "Window Control Menu Commands"

"The Backtrace Window" under "Debugger Windows" in Chapter 9, "Concepts"

Settings→Display Mode→Source Only (ALT, S, D, S)

Selects the source-only display mode.

Command File Command

```
mod(e) sou(rce) onl(y)
```

See Also

"To display source code/mnemonics only" under "Loading and Displaying Programs" in Chapter 3, "Debugging Programs"

Settings→Display Mode→Mixed (ALT, S, D, M)

Selects the source/mnemonic mixed display mode.

Command File Command

```
mod(e) sou(rce) on
```

See Also

"To display source code mixed with assembly instructions" under "Loading and Displaying Programs" in Chapter 3, "Debugging Programs"

Settings→Display Mode→Mnemonics Only (ALT, S, D, N)

Selects the mnemonic-only display mode.

Command File Command

```
mod(e) sou(rce) off
```

See Also

"To display source code/mnemonics only" under "Loading and Displaying Programs" in Chapter 3, "Debugging Programs"

Settings→Display Mode→Symbols (ALT, S, D, Y)

Displays symbols.

Command File Command

```
mod(e) sym(bols) on
mod(e) sym(bols) off
```

Settings→Display Mode→Highlight Source Lines (ALT, S, D, H)

Highlights the source lines.

The HP Debug User Interface allows you to change the color of highlighted source lines (workstation version only). See "Customizing the Debugger Interface's Appearance" in Chapter 2, "Using the Debugger Interface" for details.

Command File Command

```
mod(e) hig(hlight) on
mod(e) hig(hlight) off
```

See Also

"To highlight source code" under "Loading and Displaying Programs" in Chapter 3, "Debugging Programs""

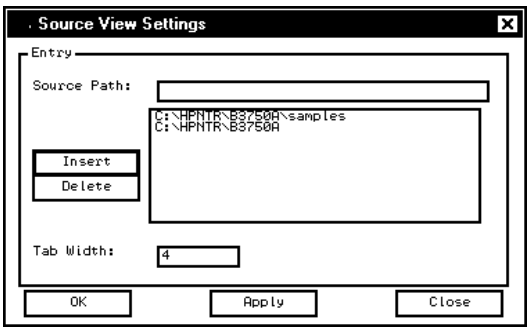
Settings→Source View... (ALT, S, S)

Displays the Source View Settings dialog box, from which you can specify the source file search path.

Source View Settings Dialog Box

Some object file formats do not contain information on source file directories. For these types of files, the Debugger cannot display the source code if the file does not exist in the current directory. Specify the source file search path in the Source View Settings dialog box.

In this dialog box, you can also specify the number of the tab code columns for displaying source codes.



Source Path	Lets you specify the directories you want to add to the source file search path. Directories in the current source file search path are listed in the dialog box.
Insert	Adds the directory entered in the Source Path text box to the source file search path.
Delete	Deletes the directory entered in the Source Path text box from the source file search path.
Tab Width:	Lets you specify the number of spaces between tabstops, from 1 to 32.
OK	Saves the settings and closes the dialog box.

Apply	Saves the settings without closing the dialog box.
Cancel	Closes the dialog box without saving the settings.
Close	Closes the dialog box.

Command File Command

`mod(e) tab` *`tabstops`*

See Also

"To specify source file directories" under "Loading and Displaying Programs" in Chapter 3, "Debugging Programs"

Settings→Configuration→Hardware... (ALT, S, C, H)

Displays the Hardware Configuration dialog box.

Setting items in the Hardware Configuration dialog box will differ depending on your processor type.

See *HP Debug User Interface User's Guide* specific to your target system and emulator.

Command File Commands

Available command file commands will differ depending on your processor type.

See *HP Debug User Interface User's Guide* specific to your target system and emulator.

Settings→Configuration→Memory Map... (ALT, S, C, M)

Displays the Memory Map dialog box for mapping ranges of memory to emulator and/or target memories.

For details on the mapping policies and available ranges for the emulator you use, see *HP Debug User Interface User's Guide* specific to your target system and emulator.

Command File Commands

Available command file commands will differ depending on your processor type.

See *HP Debug User Interface User's Guide* specific to your target system and emulator.

Note

5

Window Control Menu Commands

Window Control Menu Commands

The HP Debug User Interface provides several windows you can open from the Window menu in the Debug window.

This chapter describes the following commands:

- Common control menu commands.
- HP Debug User I/F window commands.
- Trace window commands.
- Source window commands.
- Register window commands.
- Memory window commands.
- Variable window commands.
- Peripheral window commands.
- Backtrace window commands.

Common Control Menu Commands

This section describes commands that appear in the control menus of most Debugger windows:

- File→Copy→Display... (ALT, F, P, D)
- File→Close (ALT, F, C)

The Debug window and Trace window, which automatically appear when the debugger is started, do not supply the File→Close (ALT, F, C) command.

Note

If you use HP B3755A/56A Debug User Interface, the Trace window does not appear when the debugger is started without connecting the logic analyzer. The window will appear after you connect the logic analyzer and can be closed using the File→Close (ALT, F, C) command.

File→Copy→Display... (ALT, F, P, D)

Copies the current window contents to the destination file specified.

This command displays the File Selection dialog box, from which you can specify the destination file.

Command File Command

No command.

See Also

"To copy window contents to the file" under "Working with Debugger Windows" in Chapter 2, "Using the Debugger Interface"

File→Close (ALT, F, C)

Closes the window.

Command File Command

No command.

HP Debug User I/F Window Commands

This section describes the following commands:

- File→Exit (ALT, F, X)
- Connect→Emulator... (ALT, C, E)
- Connect→Logic Analyzer... (ALT, C, L)
- Window→Tile (For PC) (ALT, W, T)*
- Window→Cascade (For PC) (ALT, W, C)*
- Window→Arrange Icons (For PC) (ALT, W, A)*
- Help→Contents (ALT, H, C)
- Help→Search for Help on... (ALT, H, S)
- Help→How to Use Help (ALT, H, H)
- Help→Tutorial (ALT, H, T)
- Help→Version... (ALT, H, V)

* Available with the PC version only.

File→Exit (ALT, F, X)

Exits the Debugger.

Command File Command

No command.

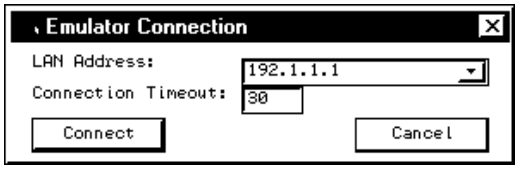
See Also

"To exit the Debugger" under "Starting and Exiting the Debugger" in Chapter 2, "Using the Debugger Interface"

Connect→Emulator... (ALT, C, E) emulator;communication

Opens the Emulator Connection Dialog Box, which lets you specify the host name or IP address of the emulator and connects it to the host computer.

Emulator Connection Dialog Box



LAN Address	Specifies the host name or the IP address of the emulator. The value is entered in integer dot notation (for example, 15.6.35.253).
Connection Timeout	Specifies, in seconds, the timeout period for connecting to the emulator. If the connection is not made within the specified time, the Debugger displays a connection error message and aborts the connection.
Connect	Connects the emulator to the host computers.
Cancel	Cancels the connection and closes the dialog box.

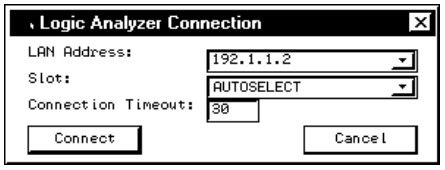
Command File Command

No command.

Connect→Logic Analyzer... (ALT, C, L)

Opens the Analyzer Connection Dialog Box, which lets you specify the host name or IP address of your logic analyzer and connects it to the host computer. This command is used if a logic analyzer is used as a tracer, which your emulation system does not originally have.

Analyzer Connection Dialog Box



LAN Address	Specifies the host name or the IP address of the analyzer. The value is entered in integer dot notation (for example, 15.6.35.253).
Slot:	Specify the slot ID indicating the logic analyzer module you wish to use if your analyzer contains more than one modules. You can select from "SLOT_A" through "SLOT_J" or "AUTOSELECT" for the slot ID. Default is the "AUTOSELECT", where modules are searched from the upper slot and the first-detected module will be used.
Connection Timeout	Specifies, in seconds, the timeout period for connecting to the analyzer. If the connection is not made within the specified time, the Debugger displays a connection error message and aborts the connection.
Connect	Connects the analyzer to the host computer.
Cancel	Cancels the connection and closes the dialog box.

Note

For emulators with trace functions, this menu option is displayed in gray and cannot be selected.

Command File Command

No command.

Window→Tile (For PC) (ALT, W, T)

Arranges and sizes windows so that none are overlapped.
Windows are sized evenly.

Command File Command

No command.

Window→Cascade (For PC) (ALT, W, C)

Arranges, sizes, and overlaps windows.
Windows are sized evenly as large as possible.

Command File Command

No command.

Window→Arrange Icons (For PC) (ALT, W, A)

Rearranges icons in the HP Debug User I/F window.
Icons are distributed evenly along the lower edge of the HP Debug User I/F window.

Command File Command

No command.

Help→Contents (ALT, H, C)

Displays the help contents.

Command File Command
No command.

Help→Search for Help on... (ALT, H, S)

Displays the help for the selected topic.
The Help→Contents command displays the contents of the help.

Command File Command
No command.

Help→How to Use Help (ALT, H, H)

Displays how to use help.

Command File Command
No command.

Help→Tutorial (ALT, H, T)

Displays the basic usage of the HP Debug User Interface.

Command File Command

No command.

Help→Version... (ALT, H, V)

Displays the Version/Copyright dialog box.

Version/Copyright Dialog Box

The Version/Copyright dialog box displays the version and copyright information on the HP Debug User Interface. The HP Debug User Interface install directory and the startup directory are also displayed.

Command File Command

No command.

Trace Window Commands

This section describes the following commands:

- Trace→Everything (ALT, T, E)
- Trace→After () (ALT, T, F)
- Trace→About () (ALT, T, A)
- Trace→Before () (ALT, T, B)
- Trace→Only () (ALT, T, O)
- Trace→Until () (ALT, T, U)
- Trace→Until Halt (ALT, T, H)
- Trace→Variable... (ALT, T, V)
- Trace→Condition... (ALT, T, C)
- Trace→Sequence... (ALT, T, S)
- Trace→Again (ALT, T, G)
- Trace→Halt (ALT, T, T)
- Display→Trigger (ALT, D, T)
- Display→Find... (ALT, D, F)
- Settings→Display Mode→Source Only (ALT, S, D, S)
- Settings→Display Mode→Mixed (ALT, S, D, M)
- Settings→Display Mode→Bus Cycles Only (ALT, S, D, B)
- Settings→Display Mode→Symbols (ALT, S, D, Y)
- Settings→Display Mode→Function Names (ALT, S, D, F)
- Settings→Display Mode→Line Numbers (ALT, S, D, L)
- Settings→Display Mode→Highlight Source Lines (ALT, S, D, H)
- Settings→Trace Count→Relative (ALT, S, C, R)
- Settings→Trace Count→Absolute (ALT, S, C, A)
- Settings→Trace Clock Speed→Very Fast (ALT, S, S, V)
- Settings→Trace Clock Speed→Fast (ALT, S, S, F)
- Settings→Trace Clock Speed→Slow (ALT, S, S, S)

Trace→Everything (ALT, T, E)

Traces all states.

Command File Command

```
tra(ce) eve(rything)
```

See Also

"To trace accesses to every state" under "Tracing Program Execution" in Chapter 3, "Debugging Programs"

Trace→After () (ALT, T, F)

Traces the program so that the specified address or symbol is located at the beginning of the trace list.
Enter an address or symbol in the entry buffer and choose this command.

Note

The Debugger recognizes symbols defined for a command containing "()" in their name as an address range.
To specify a symbol that may be considered an address range as an explicit address value, add "+0" (zero) to the end of the symbol (for example, "main+0").

Command File Command

```
tra(ce) aft(er) address
```

See Also

"To start tracing from a specified address" under "Tracing Program Execution" in Chapter 3, "Debugging Programs"

Trace→About () (ALT, T, A)

Traces the program so that the specified address or symbol is located at the center of the trace list.

Enter an address or symbol in the entry buffer and choose this command.

Note

The Debugger recognizes symbols defined for a command containing "()" in their name as an address range.

To specify a symbol that may be considered an address range as an explicit address value, add "+0" (zero) to the end of the symbol (for example, "main+0").

Command File Command

`tra(ce) abo(ut) address`

See Also

"To trace including a specified address in the center of the trace list" under "Tracing Program Execution" in Chapter 3, "Debugging Programs"

Trace→Before () (ALT, T, B)

Traces the program so that the specified address or symbol is located at the end of the trace list.
Enter an address or symbol in the entry buffer and choose this command.

Note

The Debugger recognizes symbols defined for a command containing "()" in their name as an address range.
To specify a symbol that may be considered an address range as an explicit address value, add "+0" (zero) to the end of the symbol (for example, "main+0").

Command File Command

`tra(ce) bef(ore) address`

See Also

"To end tracing at a specified address" under "Tracing Program Execution" in Chapter 3, "Debugging Programs"

Trace→Only () (ALT, T, O)

Traces only the address range for the specified symbol.
Enter a symbol in the entry buffer and choose this command.

Command File Command

`tra(ce) onl(y) address`

See Also

"To trace accesses to a specified address" under "Tracing Program Execution" in Chapter 3, "Debugging Programs"

Trace→Until () (ALT, T, U)

Traces until the specified address or symbol is reached, then stops program execution.
Enter an address or symbol in the entry buffer and choose this command.

Command File Command

`tra(ce) unt(il) address`

See Also

"To trace accesses to a specified address and break" under "Tracing Program Execution" Chapter 3, "Debugging Programs"

Trace→Until Halt (ALT, T, H)

Traces program execution until the Trace→Halt (ALT, T, T) command is entered.
This command is useful in tracing execution that leads to a processor halt or a break to the monitor. Before executing the program, choose the Trace→Until Halt (ALT, T, H) command. Then run the program. After the processor has halted or broken to the monitor, choose the Trace→Halt (ALT, T, T) command to stop the tracing. The execution that led up to the break or halt will be displayed.

Command File Command

`tra(ce) unt(il) hal(t)`

See Also

"To trace until the command is halted" under "Tracing Program Execution" in Chapter 3, "Debugging Programs"

Trace→Variable... (ALT, T, V)

Traces the program using the specified variable and value as the trigger condition.

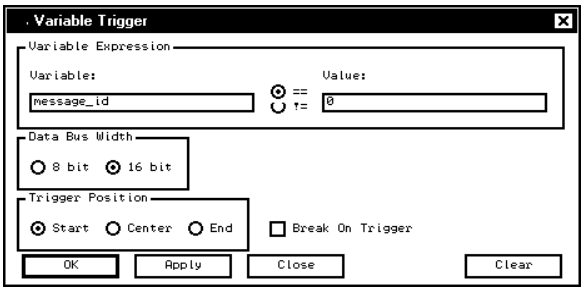
You can also specify that execution breaks to the monitor when the analyzer is triggered.

You can specify external variables and static variables, that is, variables with addresses that do not change during program execution. Variables are accessed in the address range corresponding to the type of variable.

The variable can be selected from another window before choosing the command; it will automatically appear in the edit box of the dialog box that is opened.

Variable Trigger Dialog Box

The Trace→Variable... (ALT, T, V) command displays the following dialog box:



Variable	Lets you specify the variable name you want to display.
Value	Lets you specify the value for the specified variable.
==	Triggers the analyzer when a write operation occurs to the variable with the value specified in "Value."
!=	Triggers the analyzer when a write operation occurs to the variable with values other than that specified in "Value."
Data Bus Width	Specifies the processor data bus width for the specified variable.

Trigger Position	Specifies the condition of states that will be stored in the trace buffer.
Start	Stores states in the trace buffer after the trigger conditions are met.
Center	Stores states in the trace buffer before and after the trigger conditions are met.
End	Stores states in the trace buffer before the trigger conditions are met.
Break on Trigger	Enables or disables a break to the monitor when the analyzer is triggered.
OK	Starts the specified trace and closes the dialog box.
Apply	Starts the specified trace. This command does not close the dialog box, so you can further change the trigger conditions and execute tracing.
Close	Closes the dialog box. Clicking this button before clicking the Apply button cancels the command.
Clear	Clears all specified trigger conditions.

Command File Commands

```
tra(ce) var(iable) aft(er) symbol {==|!=} value <8|16|32>
[bre(ak)]

tra(ce) var(iable) abo(ut) symbol {==|!=} value <8|16|32>
[bre(ak)]

tra(ce) var(iable) bef(ore) symbol {==|!=} value <8|16|32>
[bre(ak)]
```

See Also

"To trigger tracing on an access to a specified variable" under "Tracing Program Execution" in Chapter 3, "Debugging Programs"

Trace→Condition... (ALT, T, C)

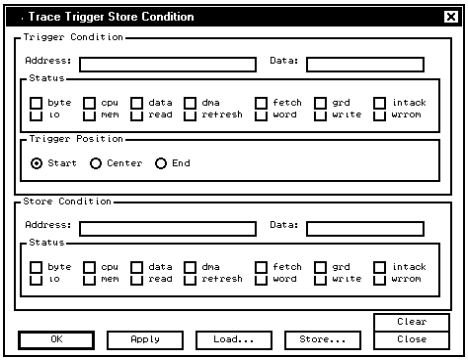
Traces program execution as specified in the Trace Trigger Store Condition dialog box.

You can specify addresses, data, and status in the dialog box. The values specified in the dialog box determine the trace conditions and trigger conditions for the analyzer.

Status represent the microprocessor's type of bus cycle. It can be select as option from the predefined list.

Trace Trigger Store Condition Dialog Box

Choosing the Trace→Condition... (ALT, T, C) command displays the following dialog box:



Trigger Condition — Provides a set of items for setting the trigger condition.

Address: — Lets you specify the address part of states.

Data: — Lets you specify the data part of states.

Status — Lets you specify the status part of states.

Trigger Position — Specifies the condition of states that will be stored in the trace buffer.

- Start — Stores states in the trace buffer after the trigger conditions are met.

Center	Stores states in the trace buffer before and after the trigger conditions are met.
End	Stores states in the trace buffer before the trigger conditions are met.

Store Condition	Provides a set of items for setting store conditions. Set the items in the same way as trigger conditions.
OK	Starts the specified trace and closes the dialog box.
Apply	Starts the specified trace. This command does not close the dialog box, so you can further change the trigger conditions and execute tracing.
Load...	Opens a file selection dialog box from which you can select previously saved trace specification file from any of the trace setting dialog boxes. Trace specification files have a ".trc" extension.
Store...	Opens a file selection dialog box from which you can store the trace specification file.
Close	Closes the dialog box. Clicking this button before clicking the Apply button cancels the command.
Clear	Clears all specified trigger conditions.

Command File Command

See Chapter 7, "Command File Command Summary"

See Also

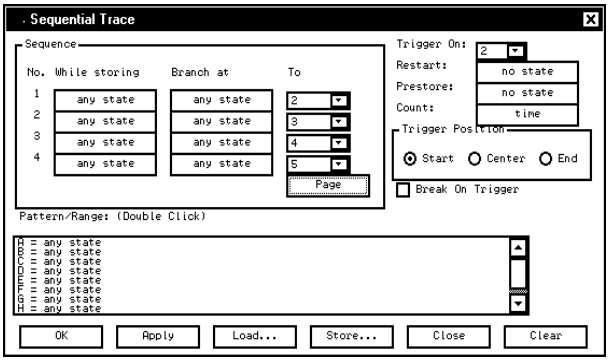
"To set up a condition trace specification" under "Setting Up Custom Trace Specifications" in Chapter 3, "Debugging Programs"

Trace→Sequence... (ALT, T, S)

Displays the Sequential Trace dialog box.

Sequential Trace Dialog Box

Choosing the Trace→Sequence...(ALT, T, S) command opens the following dialog box:



The Sequence group box specifies one type of branch condition for tracing to transfer from one sequence to another. It also specifies store conditions for each of sequence levels 1 through 8.

While storing	Lets you specify the states stored in the trace buffer at each sequence level.
Branch at	Lets you specify the condition for tracing to transfer to the sequence level specified in the To: text box.
To	Lets you specify the sequence level to which tracing is transferred when the conditions specified in the Branch at: text box are met.
Page	Toggles the display between sequence levels 1 through 4 and levels 5 through 8.
Trigger On	Lets you specify the sequence level whose entry triggers the analyzer.

Restart	Lets you specify the condition for restarting the sequence.
Prestore	Lets you qualify the states that may be stored before each stored state.
Count	Lets you specify whether time or the occurrences of a particular state are counted. You can also turn this option off.
Trigger Position	Specifies the condition of states that will be stored in the trace buffer.
Start	Stores states in the trace buffer after the trigger conditions are met.
Center	Stores states in the trace buffer before and after the trigger conditions are met.
End	Stores states in the trace buffer before the trigger conditions are met.
Break on Trigger	Enables or disables a break to the monitor when the analyzer is triggered.
Pattern/Range	Lets you specify the trace patterns for the state conditions. Double-clicking the desired pattern in the Pattern/Range list box opens the Trace Pattern dialog box or the Trace Range dialog box, where you specify the desired trace pattern or range. Clicking the Sequence, Restart, Prestore, or Count buttons causes the Condition dialog box to be opened. This dialog box lets you select or combine patterns or ranges to specify the condition.
OK	Executes the specified trace and closes the dialog box.
Apply	Executes the specified trace. This command does not close the dialog box, so you can further change the trigger conditions and execute tracing.

Load...	Opens a file selection dialog box from which you can select a previously saved trace specification file from any of the trace setting dialog boxes. Trace specification files have a ".trc" extension.
Store...	Opens a file selection dialog box from which you can store the trace specification file.
Close	Closes the dialog box. Clicking this button before clicking the Apply button cancels the command.
Clear	Restores the dialog box to its default state.

Command File Command

See Chapter 7, "Command File Command Summary"

See Also

"To set up a sequence trace specification" under "Setting Up Custom Trace Specifications" in Chapter 3, "Debugging Programs"

Trace→Again (ALT, T, G)

Traces program execution using the last trace specification stored in the emulator.

Command File Command

```
tra(ce)
```

See Also

"To repeat the last trace" under "Tracing Program Execution" in Chapter 3, "Debugging Programs"

Trace→Halt (ALT, T, T)

Stops a running trace.

This command stops a currently running trace whether the trace was started with the Trace→Until Halt (ALT, T, H) command or another trace command.

As soon as the analyzer stops the trace, stored states are displayed in the Trace window.

Command File Command

```
tra(ce) hal(t)
```

See Also

"To stop a running trace" under "Tracing Program Execution" in Chapter 3, "Debugging Programs"

Display→Trigger (ALT, D, T)

Displays the stored state with the trigger state as the heading in the Trace window.

Command File Command

No command.

See Also

"To search for a trigger/string in the trace list" under "Setting Display Mode for Tracing Results" in Chapter 3, "Debugging Programs"

Display→Find... (ALT, D, F)

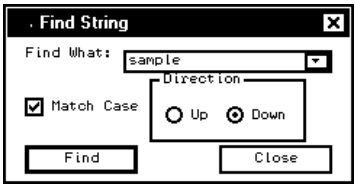
Searches for and displays a string in the Trace window.

The search starts from the current cursor position in the Trace window. You can set the search direction using the Direction option button in the Find String dialog box. Checking the Match Case check box enables a case-sensitive search.

The string can be selected from another window (that is, copied to the entry buffer) before choosing the command; it will automatically appear in the dialog box that is opened.

Find String Dialog Box

Choosing the Display→Find... (ALT, D, F) command opens the following dialog box:



Find What:	Lets you specify the string to be searched for in the Trace window.
Match Case	Enables or disables case matching.
Up	Specifies a search of the window contents from the current cursor position backward.
Down	Specifies a search of the window contents from the current cursor position forward.
Find	Searches for the string.
Close	Closes the dialog box.

Command File Command

No command.

See Also

"To search for a trigger/string in the trace list" under "Setting Display Mode for Tracing Results" in Chapter 3, "Debugging Programs"

Settings→Display Mode→Source Only (ALT, S, D, S)

Selects the source-only display mode.

Command File Command

```
tra(ce) set sou(rce) onl(y)
```

See Also

"To display bus cycles" under "Setting Display Mode for Tracing Results" in Chapter 3, "Debugging Programs"

Settings→Display Mode→Mixed (ALT, S, D, M)

Selects the source/bus cycle mixed mode.

Command File Command

```
tra(ce) set sou(rce) on
```

See Also

"To display bus cycles" under "Setting Display Mode for Tracing Results" in Chapter 3, "Debugging Programs"

Settings→Display Mode→Bus Cycles Only (ALT, S, D, B)

Selects the bus cycle-only display mode.

Command File Command

```
tra(ce) set sou(rce) off
```

See Also

"To display bus cycles" under "Setting Display Mode for Tracing Results" in Chapter 3, "Debugging Programs"

Settings→Display Mode→Symbols (ALT, S, D, Y)

Displays symbols.

Command File Commands

```
tra(ce) set sym(bols) on
```

```
tra(ce) set sym(bols) off
```

See Also

"To display symbol information in the trace list" under "Setting Display Mode for Tracing Results" in Chapter 3, "Debugging Programs"

Settings→Display Mode→Function Names (ALT, S, D, F)

Displays function names. This command causes line numbers to disappear.

Command File Command

```
tra(ce) set lin(enum) off
```

See Also

"To display the trace list with function names" under "Setting Display Mode for Tracing Results" in Chapter 3, "Debugging Programs"

Settings→Display Mode→Line Numbers (ALT, S, D, L)

Displays line numbers. This command causes function names to disappear.

Command File Command

```
tra(ce) set lin(enum) on
```

See Also

"To display the trace list with line numbers" under "Setting Display Mode for Tracing Results" in Chapter 3, "Debugging Programs"

Settings→Display Mode→Highlight Source Lines (ALT, S, D, H)

Highlights source lines.

The HP Debug User Interface allows you to change the color of highlighted source lines (WS version only). See "Customizing the Debugger Interface's Appearance" in Chapter 2, "Using the Debugger Interface" for details.

Command File Command

```
tra(ce) set hig(hlight) on
tra(ce) set hig(hlight) off
```

See Also

"To highlight source code in the trace list" under "Setting Display Mode for Tracing Results" in Chapter 3, "Debugging Programs"

Settings→Trace Count→Relative (ALT, S, C, R)

Selects the relative mode, which counts the time interval between the current and previous states.

Note

If you use the HP 64704/706A analyzer board and set the trace clock to Fast or Very Fast, states are counted instead of time. If you want to measure trace time with a 16-MHz or faster clock, you must use the HP 64794A/C/D analyzer board.

Note

If you use HP B3755A/56A Debug User Interface, maximum time count between states is 34 seconds.

Command File Command

No command.

See Also

"To specify display mode for the trace count" under "Setting Display Mode for Tracing Results" in Chapter 3, "Debugging Programs"

Settings→Trace Count→Absolute (ALT, S, C, A)

Selects the absolute mode, which counts the total time elapsed since the trigger of the trace.

Note

If you use the HP 64704/706A analyzer board and set the trace clock to Fast or Very Fast, states are counted instead of time. If you want to measure trace time with a 16-MHz or faster clock, you must use the HP 64794A/C/D analyzer board.

Note

If you use HP B3755A/56A Debug User Interface, maximum time count between states is 34 seconds.

Command File Command

No command.

See Also

"To specify display mode for the trace count" under "Setting Display Mode for Tracing Results" in Chapter 3, "Debugging Programs"

Settings→Trace Clock Speed→Very Fast (ALT, S, S, V)

Sets the trace clock to the fastest speed. Choose this command when the processor clock is less than 26 MHz.

This command can be selected only if your emulator is the HP 64704A/706A. If your emulator is the HP 64794A/C/D, this command is displayed in gray and cannot be selected.

Note

If you use the HP 64704/6A analyzer board and set the trace clock to Fast or Very Fast, states are counted instead of time. If you want to measure trace time with a 16-MHz or faster clock, you must use the HP 64794A/C/D analyzer board.

Command File Command

```
tra(ce) set clo(ck) ver(yfast)
```

See Also

"To specify the trace clock" under "Setting Up Custom Trace Specifications" in Chapter 3, "Debugging Programs"

Settings→Trace Clock Speed→Fast (ALT, S, S, F)

Sets the trace clock to the fast speed. Choose this command when the processor clock is less than 20 MHz.

This command can be selected only if your emulator is the HP 64704A or HP 64706A. If your emulator is the HP 64794A/C/D, this command is displayed in gray and cannot be selected.

Note

If you use the HP 64704/6A analyzer board and set the trace clock to Fast or Very Fast, states are counted instead of time. If you want to measure trace time with a 16-MHz or faster clock, you must use the HP 64794A/C/D analyzer board.

Command File Command

```
tra(ce) set clo(ck) fas(t)
```

See Also

"To specify the trace clock" under "Setting Up Custom Trace Specifications" in Chapter 3, "Debugging Programs"

Settings→Trace Clock Speed→Slow (ALT, S, S, S)

Sets the trace clock to the slow speed. Choose this command when the processor clock is less than 16 MHz.

This command can be selected only if your emulator is the HP 64704 or HP 64706. If your emulator is the HP 64794A/C/D, this command is displayed in gray and cannot be selected.

Command File Command

```
tra(ce) set clo(ck) slo(w)
```

See Also

"To specify the trace clock" under "Setting Up Custom Trace Specifications" in Chapter 3, "Debugging Programs"

Source Window Commands

This section describes the following commands:

- Display→Program At () (ALT, D, A)
- Display→Source Files... (ALT, D, S)
- Display→Find... (ALT, D, F)
- Settings→Display Mode→Source Only (ALT, S, D, S)
- Settings→Display Mode→Mixed (ALT, S, D, M)
- Settings→Display Mode→Mnemonics Only (ALT, S, D, N)
- Settings→Display Mode→Symbols (ALT, S, D, Y)
- Settings→Display Mode→Highlight Source Lines (ALT, S, D, H)

Display→Program At () (ALT, D, A)

Specifies the start address from which the source code is displayed.

Enter the address or symbol in the entry buffer on the tool bar and choose this command.

Command File Command

No command.

See Also

"To display source code specifying a heading line" under "Loading and Displaying Programs" in Chapter 3, "Debugging Programs"

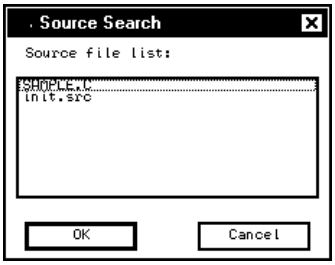
Display→Source Files... (ALT, D, S)

Displays the Source Search dialog box.

Source Search Dialog Box

Displays the contents of the specified source file in the Source window.

This command is disabled before the object file is loaded or when no source is available for the loaded object file.



Source file list:	Lists source files associated with the loaded object file. You can select the source file to be displayed from this list.
OK	Switches the Source window contents to the selected source file.
Cancel	Closes the dialog box.

Note

For some language tools, the contents of assembly source files cannot be displayed.

Command File Command

No command.

See Also

"To display source files by their names" under "Loading and Displaying Programs" in Chapter 3, "Debugging Programs"

Display→Find... (ALT, D, F)

Searches for and displays a specified string in the Source window.

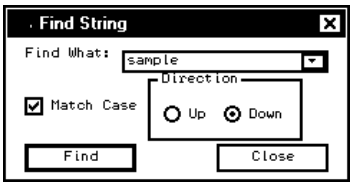
The search starts from the current cursor position in the Source window. You can set the search direction using the Direction option button in the Find String dialog box. Checking the Match Case check box enables a case-sensitive search.

Before searching for strings using this command, you must set the emulator to source-only display mode.

The string can be selected from another window (that is, copied to the entry buffer) before choosing the command; it will automatically appear in the dialog box that is opened.

Find String Dialog Box

Choosing the Display→Find... (ALT, D, F) command opens the following dialog box:



Find What:	Lets you specify the string to be searched for in the Source window.
Match Case	Enables or disables case matching.
Up	Specifies a search of the window contents from the current cursor position backward.
Down	Specifies a search of the window contents from the current cursor position forward.
Find	Searches for the string.
Close	Closes the dialog box.

Command File Command

No command.

Settings→Display Mode→Source Only (ALT, S, D, S)

Selects the source-only display mode.

Command File Command

No command.

See Also

"To display source code/mnemonics only" under "Loading and Displaying Programs" in Chapter 3, "Debugging Programs"

Settings→Display Mode→Mixed (ALT, S, D, M)

Selects the source/mnemonic mixed display mode.

Command File Command

No command.

See Also

"To display source code mixed with assembly instructions" under "Loading and Displaying Programs" in Chapter 3, "Debugging Programs"

Settings→Display Mode→Mnemonics Only (ALT, S, D, N)

Selects the mnemonic-only display mode.

Command File Command

No command.

See Also

"To display source code/mnemonics only" under "Loading and Displaying Programs" in Chapter 3, "Debugging Programs"

Settings→Display Mode→Symbols (ALT, S, D, Y)

Displays symbols.

Command File Command

No command.

Settings→Display Mode→Highlight Source Lines (ALT, S, D, H)

Highlights source lines.

The HP Debug User Interface allows you to change the color of highlighted source lines (WS version only). See "Customizing the Debugger Interface's Appearance" in Chapter 2 "Using the Debugger Interface" for details.

Command File Command

No command.

See Also

"To highlight source code" under "Loading and Displaying Programs" in Chapter 3, "Debugging Programs"

Register Window Commands

This section describes the following command:

- Settings→Auto Update (ALT, S, A)

Settings→Auto Update (ALT, S, A)

Automatically updates contents displayed in the Register window.
You can set the update interval in the initialization file (.netrap.ini for the WS version, or netrap.ini for the PC version).

Updating When an Event Occurs

The Register window is updated when an event occurs, regardless of the command setting.
"Events" refer to events that change the emulation status. The following list shows typical events:

- Breaking to the monitor.
- Changing the hardware configuration.
- Resetting the emulator.
- Changing the memory mapping.
- Loading a program file.

Command File Command

No command.

See Also

"To display registers" under "Displaying and Editing Registers" in Chapter 3, "Debugging Programs"

Memory Window Commands

This section describes the following commands:

- Display→From () (ALT, D, F)
- Display→Blocked→Byte (ALT, D, B, B)
- Display→Blocked→Word (ALT, D, B, W)
- Display→Blocked→Long (ALT, D, B, L)
- Display→Absolute→Byte (ALT, D, A, B)
- Display→Absolute→Word (ALT, D, A, W)
- Display→Absolute→Long (ALT, D, A, L)
- Display→Absolute→Float (ALT, D, A, F)
- Modify→Memory... (ALT, M, M)
- Settings→Auto Update (ALT, S, A)

Display→From () (ALT, D, F)

Specifies the start address from which memory contents are displayed in the Memory window.

Enter the address or symbol in the entry buffer on the tool bar and choose this command.

Command File Command

```
No command
```

See Also

"To display memory" under "Displaying and Editing Memory" in Chapter 3, "Debugging Programs"

Display→Blocked→Byte (ALT, D, B, B)

Displays memory contents in 8-bit block format.

Command File Command

No command.

See Also

"To display memory" under "Displaying and Editing Memory" in Chapter 3, "Debugging Programs"

Display→Blocked→Word (ALT, D, B, W)

Displays memory contents in 16-bit block format.

Command File Command

No command.

See Also

"To display memory" under "Displaying and Editing Memory" in Chapter 3, "Debugging Programs"

Display→Blocked→Long (ALT, D, B, L)

Displays memory contents in 32-bit block format.

Command File Command

No command.

See Also

"To display memory" under "Displaying and Editing Memory" in Chapter 3, "Debugging Programs"

Display→Absolute→Byte (ALT, D, A, B)

Displays memory contents in 8-bit absolute format.

Command File Command

No command.

See Also

"To display memory" under "Displaying and Editing Memory" in Chapter 3, "Debugging Programs"

Display→Absolute→Word (ALT, D, A, W)

Displays memory contents in 16-bit absolute format.

Command File Command

No command.

See Also

"To display memory" under "Displaying and Editing Memory" in Chapter 3, "Debugging Programs"

Display→Absolute→Long (ALT, D, A, L)

Displays memory contents in 32-bit absolute format.

Command File Command

No command.

See Also

"To display memory" under "Displaying and Editing Memory" in Chapter 3, "Debugging Programs"

Display→Absolute→Float (ALT, D, A, F)

Displays memory contents in floating-point absolute format.

Command File Command

No command.

See Also

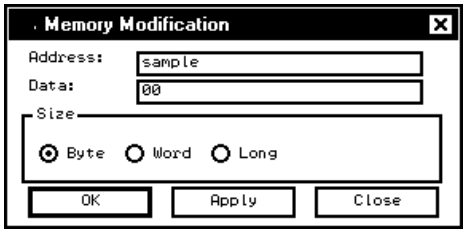
"To display memory" under "Displaying and Editing Memory" in Chapter 3, "Debugging Programs"

Modify→Memory... (ALT, M, M)

Displays the Memory Modification dialog box.

Memory Modification Dialog Box

You can edit the memory contents by entering address ranges and values in the following dialog box:



Address:	Lets you specify the address range of the memory you want to edit.
Data:	Lets you specify the new value.
Size	Selects the size of the value entered for Data: as Byte (8 bits), Word (16 bits), or Long (32 bits). If the size you specify is larger than the value's actual size, higher-order bits are neglected.
OK	Modifies memory contents with the specified value and closes the dialog box.
Apply	Modifies memory contents with the specified value. The Memory window is immediately updated to reflect the modification. This command does not close the dialog box, so you can further modify memory contents by specifying additional address ranges and values.
Close	Closes the dialog box. Clicking this button before clicking the Apply button cancels the command.

Command File Command

`mem(ory) fil(l) range byt(e) value`

Specifies the data in 8-bit format.

`mem(ory) fil(l) range wor(d) value`

Specifies the data in 16-bit format.

`mem(ory) fil(l) range lon(g) value`

Specifies the data in 32-bit format.

`mem(ory) mod(ify) range byt(e) value`

Modifies the data in 8-bit format.

`mem(ory) mod(ify) range wor(d) value`

Modifies the data in 16-bit format.

`mem(ory) mod(ify) range lon(g) value`

Modifies the data in 32-bit format.

See Also

"To edit memory" under "Displaying and Editing Memory" in Chapter 3, "Debugging Programs"

Settings→Auto Update (ALT, S, A)

Automatically updates displayed contents in the Memory window.
You can set the update interval in the initialization file (.netrap.ini for the WS version, or netrap.ini for the PC version).

Updating When an Event Occurs

The Memory window is updated when an event occurs, regardless of the command setting.
"Events" refer to events that change the emulation status. The following list shows typical events:

- Breaking to the monitor.
- Changing the hardware configuration.
- Resetting the emulator.
- Changing the memory mapping.
- Loading a program file.

Command File Command

No command.

See Also

"To display memory" under "Displaying and Editing Memory" in Chapter 3, "Debugging Programs"

Variable Window Commands

This section describes the following commands:

- Display→Variable () (ALT, D, V)
- Display→Address Of (ALT, D, A)
- Display→Contents Of (ALT, D, C)
- Modify→Variable... (ALT, M, V)
- Settings→Display Base→Default (ALT, S, D, D)
- Settings→Display Base→Decimal (ALT, S, D, C)
- Settings→Display Base→Hexadecimal (ALT, S, D, H)
- Settings→Display Base→Float (ALT, S, D, F)
- Settings→Display Base→String (ALT, S, D, S)
- Settings→Auto Update (ALT, S, A)

Display→Variable () (ALT, D, V)

Specifies the variable to be displayed.

Enter the variable name you want to display in the entry buffer on the tool bar and choose this command.

The following display formats are available:

- 10-base display
- 16-base display
- Floating-point display
- String display

For structure/union variables, all members are displayed except for structure/union/array members. For arrays, all components are displayed except for array components.

Command File Command

`dis(play) var(iable) variable`

See Also

"To display a variable" under "Displaying and Editing Variables" in Chapter 3, "Debugging Programs"

Display→Address Of (ALT, D, A)

Adds "&" to a variable name.
This command is useful when specifying pointer variables.

Command File Command

No command.

See Also

"To display a variable" under "Displaying and Editing Variables" in Chapter 3, "Debugging Programs"

Display→Contents Of (ALT, D, C)

Adds "*" to a variable name.
This command is useful when specifying pointer variables.

Command File Command

No command.

See Also

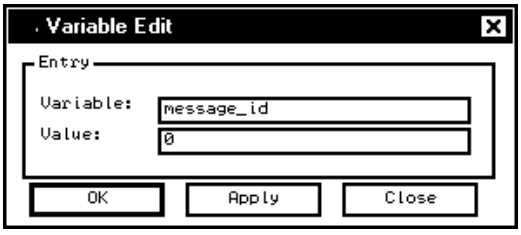
"To display a variable" under "Displaying and Editing Variables" in Chapter 3, "Debugging Programs"

Modify→Variable... (ALT, M, V)

Displays the Variable Edit dialog box, from which you can edit specified variables.

Variable Edit Dialog Box

Choosing the Modify→Variable... (ALT, M, V) command displays the following dialog box:



Variable:	Lets you specify the variable name to be edited. If a variable has been copied to the entry buffer (that is, selected from another window) before choosing this command, it will automatically appear in the entry box.
Value:	Lets you specify the value of the variable to be modified.
OK	Modifies the value of the specified variable and closes the dialog box.
Apply	Modifies the value of the specified variable. The Variable window is immediately updated to reflect the modification. This command does not close the dialog box, so you can further change variable names and their values.
Close	Closes the dialog box. Clicking this button before clicking the Apply button cancels the command.

Command File Command

`var(iable)` *`variable value`*

See Also

"To display a variable" under "Displaying and Editing Variables" in Chapter 3, "Debugging Programs"

"To edit a variable" under "Displaying and Editing Variables" in Chapter 3, "Debugging Programs"

Settings→Display Base→Default (ALT, S, D, D)

Sets the variable display format to the default.

Command File Command

No command.

See Also

"To display a variable" under "Displaying and Editing Variables" in Chapter 3, "Debugging Programs"

Settings→Display Base→Decimal (ALT, S, D, C)

Displays variable values in decimal format.

Command File Command

No command.

See Also

"To display a variable" under "Displaying and Editing Variables" in Chapter 3, "Debugging Programs"

Settings→Display Base→Hexadecimal (ALT, S, D, H)

Displays variable values in hexadecimal format.

Command File Command

No command.

See Also

"To display a variable" under "Displaying and Editing Variables" in Chapter 3, "Debugging Programs"

Settings→Display Base→Float (ALT, S, D, F)

Displays 32-bit variable values in floating-point format.

Command File Command

No command.

See Also

"To display a variable" under "Displaying and Editing Variables" in Chapter 3, "Debugging Programs"

Settings→Display Base→String (ALT, S, D, S)

Displays pointer or array variable values in string format.

Command File Command

No command.

See Also

"To display a variable" under "Displaying and Editing Variables" in Chapter 3, "Debugging Programs"

Settings→Auto Update (ALT, S, A)

Automatically updates displayed contents in the Variable window.

You can set the update interval in the initialization file (.netrap.ini for the WS version, or netrap.ini for the PC version).

Updating When an Event Occurs

The Variable window is updated when an event occurs, regardless of the command setting.

"Events" refer to events that change the emulation status. The following list shows typical events:

- Breaking to the monitor.
- Changing the hardware configuration.
- Resetting the emulator.
- Changing the memory mapping.
- Loading a program file.

Command File Command

No command.

See Also

"To display a variable" under "Displaying and Editing Variables" in Chapter 3, "Debugging Programs"

Peripheral Window Commands

This section describes the following commands:

- Display→ < register class name >
- Settings→Auto Update (ALT, S, A)

Display→ < register class name >

Displays the pull-down menu containing peripheral registers.
Select a register to display its value in the Peripheral window.

Registers that appear in the menu may change depending on the target processor you wish to evaluate.
See *HP Debug User Interface User's Guide* specific to your target system and emulator.

Command File Command

No command.

Settings→Auto Update (ALT, S, A)

Automatically updates displayed contents in the Peripheral window.

You can set the update interval in the initialization file (.netrap.ini for the WS version, or netrap.ini for the PC version).

Updating When an Event Occurs

The Peripheral window is updated when an event occurs, regardless of the command setting.

"Events" refer to events that change the emulation status. The following list shows typical events:

- Breaking to the monitor.
- Changing the hardware configuration.
- Resetting the emulator.
- Changing the memory mapping.
- Loading a program file.

Command File Command

No command.

See Also

"To display peripheral registers" under "Displaying and Editing Peripheral Registers" in Chapter 3, "Debugging Programs"

Backtrace Window Commands

This section describes the following command:

- [Source at Stack Level]

[Source at Stack Level]

For the cursor-selected function in the Backtrace window, this command displays the function call, the source code starting from the return address of the function, in the Debug window.

Command File Command

`bac(ktrace) level`

6

Window Pop-up Commands

Window Pop-up Commands

This chapter describes the commands that can be selected from the pop-up menus in HP Debug User Interface windows. Pop-up menus are accessed by clicking the right mouse button in the window.

- Debug Window Pop-up Commands
- Source Window Pop-up Commands
- Backtrace Window Pop-up Commands

Debug Window Pop-up Commands

- Set Breakpoint
- Clear Breakpoint
- Run to Here
- Evaluate ()

Set Breakpoint

Sets a breakpoint on the line containing the cursor.
See the Execution→Breakpoints... (ALT, E, P) command description.

Clear Breakpoint

Deletes the breakpoint on the line containing the cursor.
See the Execution→Breakpoints... (ALT, E, P) command description.

Run to Here

Executes the program up to the Debug window line containing the cursor.

Evaluate ()

Invokes a Variable window that displays the value of the variable selected in the entry buffer of the Debug window.

Source Window Pop-up Commands

- Set Breakpoint
- Clear Breakpoint
- Evaluate ()

Set Breakpoint

Sets a breakpoint on the line containing the cursor.
See the Execution→Breakpoints... (ALT, E, P) command description.

Clear Breakpoint

Deletes the breakpoint on the line containing the cursor.
See the Execution→Breakpoints... (ALT, E, P) command description.

Evaluate ()

Invokes a Variable window that displays the value of the variable selected in the entry buffer of the Source window.

Backtrace Window Pop-up Commands

- Source at Stack Level

Source at Stack Level

Displays the source code of the cursor-selected function in the Backtrace window, this command displays the function call in the Debug Window.

7

Command File Command Summary

Command File Command Summary

This chapter briefly describes the HP Debug User Interface command file commands and syntax. For details on each command, set the command descriptions.

This chapter classifies the commands as follows:

- Run control commands
- Variable and memory commands
- Breakpoint commands
- Window open commands
- Dialog Box Open commands
- Debug Window configuration commands
- Emulator configuration commands
- File commands
- Trace commands
- Miscellaneous commands
- Parameters

Characters in parentheses can be ignored for shortcut entry.

Run Control Commands

Command	Param_1	Param_2	Param_3	Param_4	Operation
BRE(AK)					Breaking execution
OVE(R)					Stepping over
OVE(R)	count				Repeated a number of times
OVE(R)	count	STA(RT)			From start address
OVE(R)	count	FRO(M)	address		From specified address
RES(ET)					Resetting processors
RET(URN)					Until return
RUN					From current address
RUN	RES(ET)				From reset
RUN	STA(RT)				From start address
RUN	FRO(M)	address			From specified address
RUN	UNT(IL)	address			Until specified address
STE(P)					Stepping
STE(P)	count				Repeated a number of times
STE(P)	count	FRO(M)	address		From specified address
STE(P)	count	RES(ET)			From reset
STE(P)	count	STA(RT)			From start address

Variable and Memory Commands

Command	Param_1	Param_2	Param_3	Param_4	Operation
MEM(ORY)	FIL(L)	range	BYT(E)	value	Filling memory contents (8-Bit format)
MEM(ORY)	FIL(L)	range	WOR(D)	value	Filling memory contents (16-Bit format)
MEM(ORY)	FIL(L)	range	LON(G)	value	Filling memory contents (32-Bit format)
MEM(ORY)	MOD(IFY)	address	BYT(E)	value	Editing memory contents (8-Bit format)
MEM(ORY)	MOD(IFY)	address	WOR(D)	value	Editing memory contents (16-Bit format)
MEM(ORY)	MOD(IFY)	address	LON(G)	value	Editing memory contents (32-Bit format)
VAR(IABLE)	variable	value			Editing variable

Breakpoint Commands

Command	Param_1	Param_2	Param_3	Param_4	Operation
BP	CLE(AR)	address			Deleting a breakpoint
BP	CLE(AR)	ALL			Deleting all breakpoints
BP	DIS(ABLE)	address			Disabling a breakpoint
BP	DIS(ABLE)	ALL			Disabling all breakpoints
BP	ENA(BLE)	address			Enabling a breakpoint
BP	ENA(BLE)	ALL			Enabling all breakpoints
BP	SET	address			Setting breakpoint

Window Open Commands

Command	Param_1	Param_2	Param_3	Param_4	Operation
DIS(PLAY)	BAC(KTRACE)				Opening Backtrace window
DIS(PLAY)	MEM(ORY)				Opening Memory window
DIS(PLAY)	MEM(ORY)	address			Opening Memory window with address specified
DIS(PLAY)	PER(IPHERAL)				Opening Peripheral window
DIS(PLAY)	REG(ISTER)				Opening Register window
DIS(PLAY)	SOU(RCE)				Opening Source window
DIS(PLAY)	VAR(IABLE)				Opening Variable window
DIS(PLAY)	VAR(IABLE)	variable			Opening Variable window with variable name specified

Dialog Box Open Commands

Command	Param_1	Param_2	Param_3	Param_4	Operation
$DLGCFGLOAD					Opening Configuration Load dialog box
$DLGCFGSAVE					Opening Configuration Save dialog box
$DLGSWBP					Opening Software Breakpoints dialog box
$DLGOBJ					Opening Object File Download dialog box
$DLGCFG					Opening Hardware Configuration dialog box
$DLGMAP					Opening Memory Map dialog box

Debug Window Configuration Commands

Command	Param_1	Param_2	Param_3	Param_4	Operation
MOD(E)	BP	DIS(ABLE)			Disabling a breakpoint
MOD(E)	BP	ENA(BLE)			Enabling a breakpoint
MOD(E)	HIG(HLIGHT)	OFF			Disabling highlighted display
MOD(E)	HIG(HLIGHT)	ON			Enabling highlighted display
MOD(E)	SOU(RCE)	OFF			Mnemonic only mode
MOD(E)	SOU(RCE)	ON			Source/Mnemonic mix mode
MOD(E)	SOU(RCE)	ONL(Y)			Source only mode
MOD(E)	SYM(BOLS)	OFF			Disabling symbol display
MOD(E)	SYM(BOLS)	ON			Enabling symbol display
MOD(E)	TAB	tabstops			Specifying tabstops
LOG	ON				Enabling log file output
LOG	OFF				Disabling log file output
LOG	SET	filename			Setting log file output file

File Commands

Command	Param_1	Param_2	Param_3	Param_4	Operation
FIL(E)	BIN(ARY)	filename			Loading data
FIL(E)	OBJ(ECT)	filename			Loading object
FIL(E)	OBJ(ECT)	filename	APP(END)		Appending object code
FIL(E)	SYM(BOL)	filename			Loading symbol
FIL(E)	SYM(BOL)	filename	APP(END)		Appending symbol

Trace Commands

Command	Param_1	Param_2	Param_3	Param_4	Operation
TRA(CE)					
TRA(CE)	ABO(UT)	address			Tracing about specified address
TRA(CE)	AFT(ER)	address			Tracing from specified address
TRA(CE)	BEF(ORE)	address			Tracing until specified address
TRA(CE)	EVE(RYTHING)				Tracing all statuses
TRA(CE)	HAL(T)				Stops tracing
TRA(CE)	ONL(Y)	address			Tracing specified address only
TRA(CE)	SET	CLO(CK)	VER(YFAST)		Using very fast trace clock
TRA(CE)	SET	CLO(CK)	FAS(T)		Using fast trace clock
TRA(CE)	SET	CLO(CK)	SLO(W)		Using slow trace clock
TRA(CE)	SET	LIN(ENUM)	OFF		Display linenumber
TRA(CE)	SET	LIN(ENUM)	ON		Display function name
TRA(CE)	SET	SYM(BOLS)	OFF		Disabling symbol display
TRA(CE)	SET	SYM(BOLS)	ON		Enabling symbol display
TRA(CE)	SET	SOU(RCE)	OFF		Enabling bus cycles display
TRA(CE)	SET	SOU(RCE)	ON		Enabling source/bus cycle display
TRA(CE)	SET	SOU(RCE)	ONL(Y)		Setting display to source only mode
TRA(CE)	UNT(IL)	address			Tracing until specified address and stopping program
TRA(CE)	UNT(IL)	HAL(T)			Tracing until halt

Miscellaneous Commands

Command	Param_1	Param_2	Param_3	Param_4	Operation
REG	regname	value			Editing register values
BAC(KTRACE)	level				Setting backtrace level

Parameters

Parameter	Description	Notation
address	Address	See Chapter 8 "Expressions in Commands".
count	Count	Decimal notation
filename	File name	
level	Backtrace level	
range	Address range	See Chpater 8 "Expressions in Commnads".
regname	Register name	
tabstops	Number of tabstops	
value	Value	See Chapter 8 "Expressions in Commands".
variable	Variable name	See Chapter 8 "Expressions in Commands".

8

Expressions in Commands

Expressions in Commands

When you enter values and addresses in commands, you can use the following expressions:

- Numeric constants (hexadecimal, decimal, octal, or binary values).
- Symbols (identifiers).
- Some C operators

This chapter describes the format for specifying values and addresses in commands, as well as how to specify address ranges.

Numeric Constants

All numeric constants without a suffix indicating the base are assumed to be hexadecimal, except when the number refers to a count; count values are assumed to be decimal.

Numeric values are regarded as unsigned values by default and usable values are between 0 and 0xFFFFFFFF. Adding "-" at the beginning of numeric values specifies signed values. In this case, usable values are between -0x80000000 to -0x00000001.

The Debugger expressions support the following numeric constants with or without radix:

Hexadecimal	Alphanumeric strings starting with "0x" or "0X" and consisting of any of "0" through "9", "A" through "F", or "a" through "f" (for example: 0x12345678, 0xFFFF0000). Alphanumeric strings starting with any of "0" through "9" and consisting of any of "0" through "9", "A" through "F", or "a" through "f" (for example: 12345678, 0FFFF0000).
Decimal	Numeric strings consisting of any of "0" through "9" and ending with "T" or "t" (for example: 128T, 1000t).
Octal	Numeric strings consisting of any of "0" through "7" and ending with "O" or "o" (not zero) (for example: 200o, 377O).
Binary	Numeric strings consisting of "0" or "1" and ending with "Y" or "y" (for example: 10000000y, 11001011Y).

Symbols

The Debugger expressions support the following symbols (identifiers):

- Symbols defined in C source code
- Symbols defined in assembly language source code
- Line number symbols

Symbol expressions may be in the following format:

<Module_name>:*<Variable_name>*	Directly specifies static variable.
<Module_name>:#*<Line_number>*	Specifies symbol by line number.

Module Name

The module names include C/Assembler module names as follows:

(*file_path*)*asm_file_name*	Assembler modulename
source_file_name (without extension)	C module name

Examples

Symbol expressions:

```
data[0].message
sample:#22
sample:data[1].status
&data[0]
```

C Operators

The Debugger expressions support the following C operators. The order of operator evaluation basically follows C conventions:

Pointers	* , &
Arrays	[,]
Structures or unions	. , ->
Unary minus	-
Dyadic operators	+ , -
Parentheses	(,)

The following operators are NOT available:

Unary operators	~, !, ++, --, sizeof()
Dyadic operators	*, /, %, <, >, >>, <<, &, |, ^, <=, >=, +=, ==, !=, &&, ||, ?:
Assignment operators	=, +=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=

Note

For a dyadic operator, the first operand should be a symbol.

Address Range

Address ranges should be specified in the following format:

<Start_address>..<End_address>

If the start address and the end address specified are the same value, the address is specified instead of an address range.

9

Concepts

Concepts

This chapter describes the following topics:

- Debugger Windows

Debugger Windows

This section describes the following debugger windows:

- The HP Debug User I/F window commands.
- The Debug window commands.
- The Trace window commands.
- The Source window commands.
- The Register window commands.
- The Memory window commands.
- The Variable window commands.
- The Peripheral window commands.
- The Backtrace window commands.

The HP Debug User I/F Window

The HP Debug User I/F window is used for connecting to the emulator or the logic analyzer and for exiting the Debugger.
The window for Windows 95 has the control menu.
Other windows are displayed in this window.
The window for workstations has the menu bar only and contains no windows.

The HP Debug User I/F window opens when the Debugger is started. It remains on the screen until you exit the HP Debug User Interface.

The following functions are available from the HP Debug User I/F window:

- Connecting to the emulator.
- Connecting to the logic analyzer.
- Displaying the version information.
- Cascading and tiling windows, and rearranging icons (for PC only).
- Displaying the online help.
- Exiting the HP Debug User Interface.

Note

Emulators having trace functions cannot be connected to logic analyzers.

See Also

"HP Debug User I/F Window Commands" in Chapter 5 "Window Control Menu Commands"

The Debug Window

The Debug window is a core window for debugging. You can display source files, set breakpoints, and control program execution in the window.

The Debug window opens when the HP Debug User Interface is started. It remains on the screen until you exit the HP Debug User Interface.

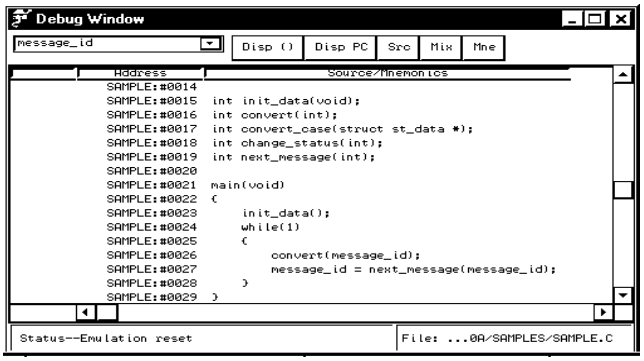
The following functions are available from the Debug window:

- Organizing files.
- Displaying source files and disassembled instructions.
- Controlling program execution.
- Setting/deleting breakpoints.
- Searching for strings.
- Entering miscellaneous settings.

The Debug window contains a cursor whose position is used to run the program to a certain line or to set and delete breakpoints.

The Debug window lets you copy strings, usually variable or function names to be used in commands, to the entry buffer by double-clicking words or by holding down the left mouse button and dragging the mouse pointer.

By clicking the right mouse button in the Debug window, you can access pop-up menu commands.



Control Menu Bar	You can access pull-down menus providing many functions from here.
Entry Buffer	This field is used to specify variable names and function names required for commands and action buttons. Clicking the button located to the right of the entry buffer displays previously specified strings for faster entry.
Source Lines	C source code is displayed when available. Source lines are preceded by their corresponding line numbers. When programs are written in assembly language or when no C source code is available, disassembled instruction mnemonics are displayed.
Disassembled Instructions	In Mnemonic Display mode, disassembled instruction mnemonics are intermixed with the source lines. Disassembled lines contain address, data, and mnemonic information. When symbolic information is available for the address, the corresponding symbol line precedes the disassembled instruction, displayed in *<module_name>:<symbol_name>* format.
Current PC	The line associated with the current program counter is highlighted.
"*" Marker	The breakpoint marker "*" appears at the beginning of breakpoint lines.
Emulation Status Area	The status of the connected emulator is indicated.
Filename	The name of the displayed source file is indicated.
Scroll Bars	For C source files, the display scrolls within the source files. For assembly language programs or programs for which no source code is available, the display scrolls for all the memory space.

Action Button	The action button assigned for a command enables you to execute the command without using the pull-down menu. Setting action buttons for frequently used commands enhances your debugging efficiency.

See Also

Chapter 4, "Debug Window Commands"
Chapter 10, "Customizing the HP Debug User Interface"

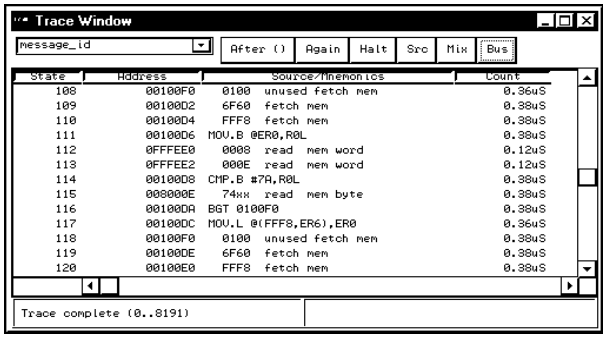
The Trace Window

The Trace window displays trace results and shows source code lines that correspond to the execution captured by the analyzer. Optionally, you can display bus cycle states along with the source code lines.

The Trace window opens when the HP Debug User Interface is started. It remains on the screen until you exit the HP Debug User Interface.

The following functions are available from the Trace window:

- Executing miscellaneous traces.
- Setting trigger conditions.
- Searching for strings.
- Setting the display mode.
- Setting the trace count display mode (absolute or relative).
- Setting the trace clock.



Each line in the Trace window includes the trace buffer state number captured in the trace buffer, the module name and line number or the function name and offset, the source line, and the time count information.

Bus cycle states show the address and data values that have been captured, as well as the disassembled instruction or status mnemonics.
The Trace window has the following display modes:

- Bus cycle only
- Source line only
- Source line/bus cycle mixed
- Symbols displayed/not displayed for addresses/operands
- Source line numbers/Function names for source lines

See Also

"Tracing Program Execution" and "Setting Up Custom Trace Specifications" in Chapter 3, "Debugging Programs"
"Trace Window Commands" in Chapter 5, "Window Control Menu Commands"
Chapter 10, "Customizing The HP Debug User Interface"

The Source Window

The Source window is used to display source files and set breakpoints.

You can open multiple instances of the Source window.

The following functions are available from the Source window:

- Displaying source files and disassembled instructions.
- Setting/deleting breakpoints.
- Searching for strings.
- Entering miscellaneous settings.

The Source window contains a cursor whose position is used to set and delete breakpoints.

The Source window lets you copy strings, usually variable or function names to be used in commands, to the entry buffer by double-clicking words or by holding down the left mouse button and dragging the mouse pointer.

By clicking the right mouse button in the Source window, you can access pop-up menu commands.

The Source window function set is a reduced set of the Debug window functions. See corresponding sections of the Debug window for a description of the window and details about functions.

See Also

Chapter 6, "Window Pop-up Commands"

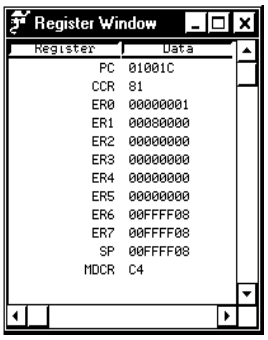
Chapter 10, "Customizing The HP Debug User Interface"

"Source Window Commands" in Chapter 5, "Window Control Menu Commands"

The Register Window

The Register window displays the name of registers and their value. Each line of the window contains the mnemonic name and the current value for each register.

You can open multiple instances of the Register window.



The following functions are available from the Register window:

- Editing register contents.
- Automatically updating the display.

You can modify register contents by double-clicking on the value, using the keyboard to type in the new value, and pressing Return or Tab. You can use symbols for specifying values.

Register window contents are updated automatically when the emulation status changes. The status is displayed at the lower left of the Debug window. The window can also be set to update the display contents at specified intervals. You can set the update interval in the initialization file (.netrap.ini for the workstation version, or netrap.ini for the PC version).

See Also

"Displaying and Editing Registers" in Chapter 3, "Debugging Programs"

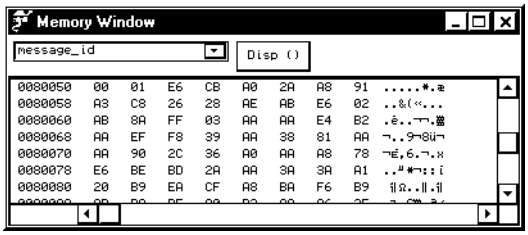
"Register Window Commands" in Chapter 5, "Window Control Menu Commands"

Chapter 10, "Customizing The HP Debug User Interface"

The Memory Window

The Memory window displays memory contents in the specified address range. You can specify an address from which the memory contents will be displayed when opening the Memory window by entering an address in the entry buffer on the Debug window tool bar.
The display start address can also be specified using the Disp () button in the Memory window. Specifying an address in the entry buffer in the Memory window and clicking this button displays the memory contents starting from the specified address.

You can open multiple instances of the Memory window.



The following functions are available from the Memory window:

- Changing the display format for memory contents.
- Editing memory contents.
- Automatically updating the display.

The following display formats are available in the Memory window:

- 8-bit, 16-bit, or 32-bit block format.
- 8-bit, 16-bit, 32-bit or float absolute format.

The default is the 8-bit block format. The Memory window contains the Display menu that lets you choose the format of the memory display from 8-bit, 16-bit, 32-bit, and float formats. When the absolute format is selected, symbols corresponding to addresses are displayed. When data is displayed in byte format, ASCII characters for the byte values are also displayed.

To edit memory contents, select the value you want to edit, enter the new value, and press Return or Tab. To fill memory, enter addresses and data using the Memory Modification dialog box. You can use symbols in either case.

Memory window contents are updated automatically when the emulation status changes. The status is displayed at the lower left of the Debug window. The window can also be set to update the display contents at specified intervals. You can set the update interval in the initialization file (.netrap.ini for the workstation version, or netrap.ini for the PC version).

See Also

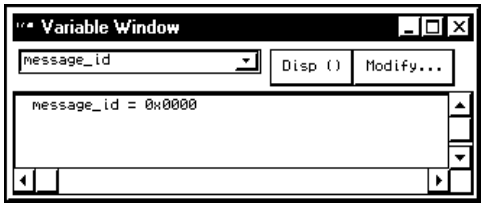
"Displaying and Editing Memory" in Chapter 3, "Debugging Programs"

"Memory Window Commands" in Chapter 5, "Window Control Menu Commands"

Chapter 10, "Customizing The HP Debug User Interface"

The Variable Window

The Variable window displays the value of variables. As you specify an address in the entry buffer on the Debug window tool bar, then bring up the pop-up menu, and select Evaluate (), the Variable window will open displaying a value of the variable you specified.
The address can also be specified with the Disp () button in the Variable window. Specify an address in the entry buffer in the Variable window and click this button to display the variable value at the specified address.
You can open multiple instances of the Variable window.



The following functions are available from the Variable window:

- Displaying variables.
- Changing the display format.
- Editing values.
- Automatically updating the display.

The following display formats for variables are available:

- Decimal format display.
- Hexadecimal format display.
- Floating-point format display.
- String format display.

For structure/union variables, all members are displayed except for structure/union/array members. For arrays, all components are displayed except for array components.
The Display→Address Of command, which adds "&" to a string, and the Display→Contents Of command, which adds "*" to a string, facilitate pointer variable reference.

To edit variable values, choose the Modify→Variable... command from the control menu to display the Variable Edit dialog box and specify the values.

Variable window contents are updated when the emulation status changes. The status is displayed at the lower left of the Debug window. The window can also be set to update the display contents at specified intervals. You can set the update interval in the initialization file (.netrap.ini for the WS version, or netrap.ini for the PC version).

See Also

"Displaying and Editing Variables" in Chapter 3, "Debugging Programs"

"Variable Window Commands" in Chapter 5, "Window Control Menu Commands"

Chapter 10, "Customizing The HP Debug User Interface"

The Peripheral Window

The Peripheral window displays values for on-chip peripheral registers.

Registers that can be displayed in the window may change depending on the target processor you wish to evaluate. Refer to the *HP Debug User Interface User's Guide* specific to your target system and emulator for further information.

The class name of the currently selected register is displayed in the status area in the window.

You can open multiple instances of the Peripheral window.

The following functions are available from the Peripheral window:

- Editing peripheral register contents.
- Automatically updating the display.

You can modify peripheral register contents by double-clicking on the value, using the keyboard to type in the new value, and pressing Return or Tab. You can use symbols for specifying values.

Peripheral window contents are updated automatically when the emulation status changes. The status is displayed at the lower left of the Debug window. The window can also be set to update the display contents at specified intervals. You can set the update interval in the initialization file (.netrap.ini for the workstation version, or netrap.ini for the PC version).

See Also

"Displaying and Editing Peripheral Registers" in Chapter 3, "Debugging Programs"

"Peripheral Window Commands" in Chapter 5, "Window Control Menu Commands"

Chapter 10, "Customizing The HP Debug User Interface"

HP Debug User Interface User's Guide specific to your target system and emulator.

The Backtrace Window

The Backtrace window displays the function associated with the current program counter value and this function's caller functions. The most recently called function is displayed at the top of the window. You can specify the stack level of the functions to be displayed. The current arguments of these functions are also displayed.

You can open multiple instances of the Backtrace window.

The window displays the stack level of return addresses at the left of the window. The highlighted stack level indicates the scope which is used for referencing stack variables. Moving the cursor to a stack level, clicking the right mouse button in the window, and choosing the Source at Stack Level command displays the source code starting from the return address in the Debug window. You will see the ">>" marker which indicates the current scope location. This shows you that the function currently referenced is in the scope. As the program counter changes, the marker moves to a new location where the program counter resides.

When the source code is not displayed, the corresponding mnemonic is displayed.

The Backtrace window is updated when program execution stops at a breakpoint, a break, or a Step command.

See Also

"Backtrace Window Commands" in Chapter 5, "Window Control Menu Commands"

Chapter 10, "Customizing The HP Debug User Interface"

Note

10

Customizing the HP Debug User Interface

Customizing the HP Debug User Interface

You can customize operation parameters and each window interface of the HP Debug User Interface.

This chapter describes the following topics:

- General information on customizing.
- Customizing global setting of Debug User Interface.
- Customizing each windows.
- Customizing the Action Buttons.

General information on customizing

You can customize the HP Debug User Interface by editing the initialization file. This file contains values for menus and parameters. The file is loaded when the HP Debug User Interface is started.

The following table lists the name of the initialization file and the directory where the file is saved.

Platform	**Initialization File Name**	**Directory**
Windows 95	netrap.ini	Start up directory
UNIX	.netrap.ini	Home directory

Settings in the initialization file are classified by window and function. A section name is given for each classification. Each setting should be written to the appropriate block in the following format:

<Parameter_name> = <Value>

Customizing global setting of Debug User Interface

This section describes parameter names and values used in the initial condition-setting file for global settings of the HP Debug User Interface.

Setting contents	Section name	Parameter name	Settable value
Object file extension	[Dlgobj]	suffix	Any strings
Any strings Number of status labels per line	[Dlgtgst]	numPerLine	Positive integer
Number of status labels per line	[Dlgtpat]	numPerLine	Positive integer
Step timeout [s]	[Debcore]	stepTimeout	1 - 20
Number of step retries	[Debcore]	stepRetryCount	10 - 500
String variable display length	[Debcore]	stringLength	Positive integer
Number of displayed array elements	[Debcore]	arrayElements	Positive integer
Indent for displaying structure members	[Debcore]	indentWidth	0 - 8
Polling interval to emulator status [ms]	[Tgt]	pollInterval	Positive integer
Polling interval to trace status [ms]	[Trccore]	pollInterval	Positive integer
Number of states for updating trace display	[Trccore]	captureStateNum	1 - 512
Maximum number of traceable states[K state]	[Trccore]	traceDepth	Positive integer
Address list of emulators	[WinappNetrap]	connectDlg.addressList	Any strings
Default address of the emulator	[WinappNetrap]	connectDlg.defaultAddress	Any strings
Connection timeout to emulator[s]	[WinappNetrap]	connectDlg.timeout	Positive integer

Customizing each windows

This section describes parameter names and values in the initial condition-setting file for the following windows:

- The Debug window
- The Source window
- The Memory window
- The Register window
- The Peripheral window
- The Backtrace window
- The Variable window
- The Trace window

Customizing the Debug Window

Setting contents	Section name	Parameter name	Settable value
Window display location (upper left y)	[Debug]	top	Positive integer
Window display location (upper left x)	[Debug]	left	Positive integer
Window display size (width)	[Debug]	width	Positive integer
Window display size (height)	[Debug]	height	Positive integer
Log execution mode	[Debug]	logMode	0:off, 1:on
Log file name	[Debug]	logFile	Any strings
Source display mode	[Debug]	sourceMode	0:Mnemonic only, 1:Mixed, 2:Source only
Symbol display mode	[Debug]	symbolMode	0:off, 1:on
Highlighted source	[Debug]	highLight	0:off, 1:on
Initial Entry buffer value	[Debug]	toolbarString	Any strings
Source search path	[Debug]	sourcePathList	Any strings
Number of tab columns	[Debug]	tabWidth	Positive integer
Number of address columns	[Debug]	addressWidth	Positive integer

Customizing the Source Window

Setting contents	Section name	Parameter name	Settable value
Window display location (upper left y)	[Source]	top	Positive integer
Window display location (upper left x)	[Source]	left	Positive integer
Window display size (width)	[Source]	width	Positive integer
Window display size (height)	[Source]	height	
Source display mode	[Source]	sourceMode	0:Mnemonic only, 1:Mixed, 2:Source only
Symbol display mode	[Source]	symbolMode	0:off, 1:on
Highlighted source	[Source]	highLight	0:off, 1:on
Number of address columns	[Source]	addressWidth	Positive integer

Customizing the Memory Window

Setting contents	Section name	Parameter name	Settable value
Window display location (upper left y)	[Memblk]	top	Positive integer
Window display location (upper left x)	[Memblk]	left	Positive integer
Window display size (width)	[Memblk]	width	Positive integer
Window display size (height)	[Memblk]	height	Positive integer
Display update interval [ms]	[Memblk]	pollInterval	Positive integer
Number of read bytes per access	[Memblk]	accessByteNum	1 - 32

Customizing the Register Window

Setting contents	Section name	Parameter name	Settable value
Window display location (upper left y)	[Reg]	top	Positive integer
Window display location (upper left x)	[Reg]	left	Positive integer
Window display size (width)	[Reg]	width	Positive integer
Window display size (height)	[Reg]	height	Positive integer
Display update interval [ms]	[Reg]	pollInterval	Positive integer

Customizing the Peripheral Window

Setting contents	Section name	Parameter name	Settable value
Window display location (upper left y)	[Peripheral]	top	Positive integer
Window display location (upper left x)	[Peripheral]	left	Positive integer
Window display size (width)	[Peripheral]	width	Positive integer
Window display size (height)	[Peripheral]	height	Positive integer
Display update interval [ms]	[Peripheral]	pollInterval	Positive integer

Customizing the Backtrace Window

Setting contents	Section name	Parameter name	Settable value
Window display location (upper left y)	[Btrace]	top	Positive integer
Window display location (upper left x)	[Btrace]	left	Positive integer
Window display size (width)	[Btrace]	width	Positive integer
Window display size (height)	[Btrace]	height	Positive integer
Maximum nesting level	[Btrace]	maxLevel	Positive integer

Customizing the Variable Window

Setting contents	Section name	Parameter name	Settable value
Window display location (upper left y)	[Var]	top	Positive integer
Window display location (upper left x)	[Var]	left	Positive integer
Window display size (width)	[Var]	width	Positive integer
Window display size (height)	[Var]	height	Positive integer
Display update interval [ms]	[Var]	pollInterval	Positive integer

Customizing the Trace Window

Setting contents	Section name	Parameter name	Settable value
Window display location (upper left y)	[Trc64]	top	Positive integer
Window display location (upper left x)	[Trc64]	left	Positive integer
Window display size (width)	[Trc64]	width	Positive integer
Window display size (height)	[Trc64]	height	Positive integer
Trace list generation interval [ms]	[Trc64]	updateInterval	Positive integer (over 1000)
Number of states processed per one list generation	[Trc64]	stateNumOneTime	Positive integer (over 128)
Source display mode	[Trc64]	sourceMode	0:Mnemonic only, 1:Mixed, 2:Source only
Symbol display mode	[Trc64]	symbolMode	0:off, 1:on
Highlighted source	[Trc64]	highLight	0:off, 1:on
Line number/function set, symbol selection	[Trc64]	srclineMode	0:Function name+off 1:Line number
State column width	[Trc64]	stateWidth	Positive integer
Address column width	[Trc64]	addressWidth	Positive integer
Count column width	[Trc64]	countWidth	Positive integer

Example

Initialization file settings for the Debug window:

```
[Debug]
; log setting mode (0: off 1: on)
logMode=0
; log file name
logFile=log.cmd
; source reference mode (0: mnemonics only 1: mixed 2:
source only)
sourceMode=1
; symbol reference mode (0: off 1: on)
symbolMode=1
; source line highlight mode (0: off 1: on)
highLight=1
; toolbar initial string
toobarString=main
; source path list
sourcePathList=
; tab width
tabwidth=4
; address column width
addressWidth=20
; window size
top=150 left=100 width=600 height=300
```

Customizing the Action Buttons

Some of the Debugger's windows have action buttons, located to the right of the entry buffer.

Using action buttons lets you execute a command with a click of the mouse instead of choosing the command from the menu.

For example, in the Debug window, you can display a source code (or a mnemonic code), starting from the address specified in the entry buffer, by clicking the Disp () action button. Without the action button, you must choose the Display command from the control menu and then choose the Program At () command.

By default, action buttons are defined for common debugging operations. In some windows, you can define action buttons to customize operations.

Action buttons can be defined in the initialization file (.netrap.ini for WS version, netrap.ini for PC version).

The syntax is as follows:

```
actionButton= "Button name" "Command file command
executed" \\                    . . .
. . .
            "Button name" "Command file command
executed"
```

Lines starting with ";" are comment lines. When specifying several values for one parameter, place \ between two values.

Example

Action button setting of the initialization file for the Debug window:

```
[Debug]
; action button
actionButton=  "Disp ()"    "display from ()"   \\
               "Disp PC"    "display from PC"     \\
               "Var ()"     "display variable ()" \\
               "Run"        "run"   \\
               "Break"      "break" \\
               "Step"       "step"  \\
               "Over"       "over"
```

Customizing the Debugger Interface's Appearance (for workstations)

The HP Debug User Interface for workstations enables you to specify the font and colors using the X resources.

Resources should be specified in the .Xdefaults file in the home directory.

You can specify the following resource names:

pkgui*foreground: *color*
Specifies the foreground color.

pkgui*background: *color*
Specifies the background color.

pkgui*highlight: *color*
Specifies the color for highlighted source code.

pkgui*fontList: *font list*
Specifies the font list.

pkgui*font: *font*
Specifies the font type.

Glossary

Terms used in the Debugger help information:

analyzer An instrument that captures data on signals of interest at discreet periods. The emulation bus analyzer captures emulator bus cycle information synchronously with the processor's clock signal.

breakpoint A point you identify in the user program where program execution is to stop. Breakpoints let you look at the state of the target system at particular points in the program.

control menu The menu that is accessed by clicking the control menu box in the upper left corner of a window. You can also access control menus by pressing the Alt and "-" keys.

count condition Specifies whether time or the occurrences of a particular state are counted for each state in the trace buffer.

embedded microprocessor system The microprocessor system that the emulator plugs into.

emulation memory Memory provided by the emulator that can be used in place of memory in the target system.

emulation monitor A program, executed by the emulation microprocessor (as directed by the emulation system controller), that gives the emulator access to target system memory, microprocessor registers, and other target system resources.

emulator An instrument that performs just like the microprocessor it replaces, while providing information about the processor's operation. An emulator gives you control over target system execution and allows you to view or modify the contents of processor registers, target system memory, and I/O resources.

guarded memory Memory locations that should not be accessed by user programs. These locations are specified when mapping memory. If the user program accesses a location mapped as guarded memory, emulator execution breaks to the monitor.

monitor program A program, executed by the emulation microprocessor (as directed by the emulation system controller), that gives the emulator access to target system memory, microprocessor registers, and other target system resources.

object file A file that can be loaded into emulation or target system memory and executed by the debugger.

pop-up menu A menu that is accessed by clicking the right mouse button in a window.

prestore condition Specifies the states that may be stored before each normally stored state.

restart condition Specifies the condition that restarts the two-step sequential trigger. That is, if the restart condition occurs while the analyzer is searching for the trigger condition, the analyzer starts looking for the enable condition again.

sequence levels Levels in the analyzer that let you specify a complex sequential trigger condition. For each level, the analyzer searches for branch conditions. You can specify a different store condition for each level. The Page button toggles the display between sequence levels 1 through 4 and sequence levels 5 through 8.

state qualifier A combination of address, data, and status values that identifies particular states captured by the analyzer.

start address The program's start address defined by the software development tools and included with the symbolic information in the object file.

store condition Specifies which states get stored in the trace buffer. In the "Condition" trace setup, the store condition specifies the states that get stored after the trigger. In the "Sequence" trace setup, each sequence level

has a store condition that specifies the states that get stored while looking for the branch conditions.

target system The microprocessor system that the emulator plugs into.

trace state The information captured by the analyzer on a particular microprocessor bus cycle.

trigger The captured analyzer state about which other captured states are stored. The trigger state specifies when the trace measurement is taken.

trigger condition Specifies the condition that causes states to be stored in the trace buffer.

trigger position Specifies whether the state that triggered the analyzer appear at the start, center, or end of the trace buffer. That is, the trigger position specifies whether states are stored after, about, or before the trigger.

trigger store condition Specifies which states get stored in the trace buffer while the analyzer searches for the trigger condition.

Note

Index

A

absolute time-count mode, 98, 182
address range, 232
analyzer, 261-263
  communication, 156
  halting, 93, 173
  repeating last trace, 93, 173
  setting by "Condition", 101
  setting by Trace Trigger Store Condition dialog box, 168
  setting up with "Sequence", 103, 170
  setting up with "Trigger Store", 101
  tracing until halt, 93, 165
Analyzer Connection dialog box, 156
arguments
  function, 249
arm condition, 103
arrays (C operators), 231
ASCII values in Memory window, 75, 244
assembly language instructions
  stepping multiple, 127
  stepping single, 60-61, 127

B

Backtrace window, 249
BP marker, 25
break to monitor, 64, 130
breakpoint, 261-263
breakpoint marker, 23, 237
breakpoints
  deleting, 25, 70
  disabling, 69
  listing, 132-134
  setting, 23, 67
  Software Breakpoints dialog box, 132-134
bus cycles
  displaying, 97, 178

C
C operators, 231
Clear Breakpoint command, 217, 219
command files
  command summary, 222
  creating, 43, 121
  executing, 44, 119-120
  parameters, 119-120
  turning logging on or off, 122
command line options for connection, 37
command summary, 222
communications (analyzer)
  setting up, 156
communications (emulator)
  setting up, 155
"Condition" trace, 101
conditions
  state, 168
  store, 101
  trigger, 101
configurations
  emulator, 147
  saving and loading, 108-110
connection
  command line option, 37
control menu, 261-263
count condition, 261-263

D
Debug window, 237
  displaying Backtrace window, 141
  displaying from current program counter, 135
  displaying from specified address, 135
  displaying Memory window, 139
  displaying mnemonics only, 143
  displaying Peripheral window, 140
  displaying Register window, 139
  displaying source and mnemonics mixed, 142
  displaying source only, 142
  displaying Source window, 139
  displaying Variable window, 140
  setting tabstops, 40
  To display symbol information, 143
debugger

arranging icons in window, 158
cascaded windows, 158
exiting, 34, 38, 154
overview, 4
starting, 17-18, 37
tiled windows, 158
debugger windows
opening, 39
demo programs, 16
loading, 21
mapping memory, 20
running, 24
Display command, 175
display contents
outputting, 118
display mode
mixed, 53
source only, 52
toggling, 142
Display-Absolute-Byte (ALT, D, A, B) command, 198
Display-Absolute-Float (ALT, D, A, F) command, 199
Display-Absolute-Long (ALT, D, A, L) command, 199
Display-Absolute-Word (ALT, D, A, W) command, 198
Display-Address Of (ALT, D, A) command, 205
Display-Blocked-Byte (ALT, D, B, B) command, 196
Display-Blocked-Long (ALT, D, B, L) command, 197
Display-Blocked-Word (ALT, D, B, W) command, 197
Display-Contents Of (ALT, D, C) command, 205
Display-Find... (ALT, D, F) command, 94, 137-138, 189
Display-From () (ALT, D, F) command, 196
Display-Program at () (ALT, D, A) command, 135, 187
Display-Program at PC (ALT, D, P) command, 135
Display-Source Files... (ALT, D, S) command, 136, 188
Display-Trigger (ALT, D, T) command, 94, 174
Display-Variable () (ALT, D, V) command, 204
displaying symbols, 178-179

E

embedded microprocessor system, 261-263
emulation memory, 261-263
emulation microprocessor
resetting, 65, 131
emulation monitor, 261-263

emulator, 261-263
emulator configuration, 147
  loading, 109, 114
  saving, 108, 117
Emulator Connection dialog box, 155
Evaluate command, 218-219
Execution-Break (ALT, E, B) command, 130
Execution-Breakpoints... (ALT, E, P) command, 132-134
Execution-Reset (ALT, E, T) command, 131
Execution-Run-From () (ALT, E, R, F) command, 123
Execution-Run-From PC (ALT, E, R, P) command, 123
Execution-Run-From Reset (ALT, E, R, R) command, 124
Execution-Run-From Start Address (ALT, E, R, S) command, 124
Execution-Run-Return to Caller (ALT, E, R, C) command, 126
Execution-Run-Until () (ALT, E, R, U) command, 125
Execution-Step-From () (ALT, E, S, F) command, 127
Execution-Step-From PC (ALT, E, S, P) command, 127
Execution-Step-From Start Address (ALT, E, S, S) command, 128
Execution-Step-Over Procedure Call (ALT, E, S, O) command, 129
expressions, 228

F
file
  copying window contents to, 40
File-Close (ALT, F, C) command, 152
File-Copy-Display... (ALT, F, P, D) command, 118, 152
File-Exit (ALT, F, X) command, 154
File-Load-Configuration... (ALT, F, L, C) command, 114
File-Load-Program... (ALT, F, L, P) command, 115-116
File-Log-Playback... (ALT, F, O, P) command, 119-120
File-Log-Record... (ALT, F, O, R) command, 121
File-Log-Stop (ALT, F, O, S) command, 122
File-Store-Configuration... (ALT, F, S, C) command, 117
Find command, 137-138, 189
function arguments, 249
functions
  running until return, 29, 59, 126
  searching, 179
  stepping over, 26, 62
  to step over a function, 129

G
glossary, 261-263
guarded memory, 147-148, 261-263

H help
contents, 159
displaying, 159
how to use, 159
tutorial, 160
Help-Contents (ALT, H, C) command, 159
Help-How to Use Help (ALT, H, H) command, 159
Help-Search for Help on... (ALT, H, S) command, 159
Help-Tutorial (ALT, H, T) command, 160
Help-Version... (ALT, H, V) command, 160
hostname, 155-156
HP Debug User I/F window, 236

I icons (debugger window)
arranging, 158
.ini file, 58, 125, 194, 202, 211, 213, 243, 245, 247-248, 253, 259
IP address, 155-156

L labels, 230
levels
trace sequence, 103, 170
line (source file)
running until, 28, 125
line number
displaying, 179
log (command) files, 43, 119-122
logical operators, 103

M memory
displaying, 75
displaying from specified address, 196
editing, 77, 200
filling, 78
mapping, 147-148
memory type, 147-148
Memory window, 244
disabling auto display update, 202
displaying in 16-bit absolute format, 198
displaying in 16-bit block format, 197
displaying in 32-bit absolute format, 199
displaying in 32-bit block format, 197
displaying in 8-bit absolute format, 198

displaying in 8-bit block format, 196
enabling auto display update, 202
in floating point absolute format, 199
microprocessor
resetting, 65, 131
mixed display mode, 53, 142
Modify-Memory... (ALT, M, M) command, 200
Modify-Variable... (ALT, M, V) command, 206
monitor program, 261-263

N numeric constants, 229

O object file, 261-263
object files
loading, 50, 115-116
operators
C, 231
logical, 103
overview, 4

P parameters
command file, 119-120
path for source file search, 55
patterns
trace, 103, 170
Peripheral window, 248
pointers (C operators), 231
pop-up menu, 261-263
pop-up menus
accessing, 216
prestore condition, 103, 170, 240, 261-263
processor
resetting, 65, 131
program counter, 57, 60, 123, 127, 237
programs
demo, 16
executing, 124
loading, 50, 115-116
running, 123
stopping execution, 64

Q qualifier
state, 101

R Register window, 243
disabling auto display update, 194
enabling auto display update, 194
registers
displaying, 32, 79, 82
editing, 83
relative time-count mode, 98, 181
reset
emulator, 65, 131
executing from target reset, 124
running from target system, 123
restart condition, 103, 261-263
return (function)
running until, 29, 59, 126
run
from a specified address, 57
from the current program counter, 57
from the start address, 58
fromthe target reset, 58
until a specified address, 58
Run to Here command, 217

S search for trigger, 94
search path for source files, 55
sequence levels, 261-263
"Sequence" trace, 103
sequential trigger, 170
Set Breakpoint command, 217, 219
Settings-Auto Update (ALT, S, A) command, 194, 202, 211, 213
Settings-Configuration-Hardware... (ALT, S, C, H) command, 147
Settings-Configuration-Memory Map... (ALT, S, C, M) command, 147-148
Settings-Connect-Emulator... (ALT, C, E) command, 155
Settings-Connect-Logic Analyzer... (ALT, C, L) command, 156
Settings-Display Base-Decimal (ALT, S, D, C) command, 208
Settings-Display Base-Default (ALT, S, D, D) command, 208
Settings-Display Base-Float (ALT, S, D, F) command, 209
Settings-Display Base-Hexadecimal (ALT, S, D, H) command, 209
Settings-Display Base-String (ALT, S, D, S) command, 210
Settings-Display Mode-Bus Cycles Only (ALT, S, D, B) command, 178
Settings-Display Mode-Function Names (ALT, S, D, F) command, 99, 179
Settings-Display Mode-Highlight Source Lines (ALT, S, D, H) command, 99, 144, 180, 193

Settings-Display Mode-Line Numbers (ALT, S, D, L) command, 99, 179
Settings-Display Mode-Mixed (ALT, S, D, M) command, 142, 177, 191
Settings-Display Mode-Mnemonics Only (ALT, S, D, N) command, 143, 192
Settings-Display Mode-Source Only (ALT, S, D, S) command, 142, 177, 191
Settings-Display Mode-Symbols (ALT, S, D, Y) command, 98, 143, 178, 192
Settings-Source View... (ALT, S, S) command, 145-146
Settings-Trace Clock Speed-Fast command, 95
Settings-Trace Clock Speed-Slow command, 95
Settings-Trace Clock Speed-Very Fast command, 95
Settings-Trace Count-Absolute(ALT, S, C, A) command, 98
Settings-Trace Count-Relative(ALT, S, C, R) command, 98
single-step one line, 27
source display mode
    toggling, 142
source file line
    running until, 125
source files
    displaying, 22, 51, 136
    displaying from a specified heading line, 51
    displaying from the current program counter, 52
    highlighting, 54
    searching for function names, 179
    searching for strings, 137-138, 175, 189
    specifying search directories, 55
source lines
    highlighting, 99, 180
    running until, 28
    stepping multiple, 127
    stepping single, 60-61, 127
source only
    displaying, 52, 142
Source Search dialog box, 136, 188
Source View Settings dialog box, 145-146
Source window, 242
    displaying from specified address, 187
    displaying mnemonic only, 192
    displaying source and mnemonic mixed, 191
    displaying source code only, 191
    displaying symbol information, 144
    displaying symbols, 192-193
    toggling the display mode, 142

stack level, 220
start address, 24, 58, 61, 123-124, 127
state conditions
  specifying, 168
state qualifier, 101, 261-263
status values, 261-263
step one line, 27
store condition, 101, 261-263
strings
  searching in trace result, 94, 175
  searching source files, 137-138, 175, 189
structures (C operators), 231
subroutine
  to step over a function, 129
symbol information
  displaying, 98
symbols, 230

T tabstops in the Debug window
  setting, 40
target reset, 58
target system, 261-263
to display function names, 99
to display line numbers, 99
trace
  about specified address, 163
  all states, 162
  before specified address, 164
  "Condition", 101
  from specified address, 162
  "Sequence", 103
  specified address only, 164
  until specified address, 165
trace clock, 95, 183-185
  specifying, 95
Trace Clock Speed-Fast (ALT, S, P, F) command, 184
Trace Clock Speed-Slow (ALT, S, P, S) command, 185
Trace Clock Speed-Very Fast (ALT, S, P, V) command, 183
Trace Condition dialog box, 170
trace count, 98, 181-182
Trace Count-Absolute (ALT, S, C, A) command, 182
Trace Count-Relative (ALT, S, C, R) command, 181

trace patterns, 103, 170
trace result
  searching for strings, 94, 175
  searching for trigger, 94
trace specification
  loading, 107
  storing, 107
trace state, 261-263
Trace window, 240
  displaying source and mnemonic mixed, 177
  displaying source code only, 177
Trace-About () (ALT, T, A) command, 163
Trace-After () (ALT, T, A) command, 162
Trace-Again (ALT, T, G) command, 173
Trace-Before () (ALT, T, B) command, 164
Trace-Condition... (ALT, T, C) command, 168
Trace-Everything (ALT, T, E) command, 162
Trace-Halt (ALT, T, T) command, 173
Trace-Only () (ALT, T, O) command, 164
Trace-Sequence... (ALT, T, S) command, 170
Trace-Until () (ALT, T, U) command, 165
Trace-Until Halt (ALT, T, H) command, 165
Trace-Variable... (ALT, T, V) command, 166
trigger, 261-263
  searching, 174
trigger condition, 101, 261-263
trigger position, 261-263
trigger store condition, 261-263
tutorial, 16
type of memory, 147-148

U
unary (C operators), 231
unions (C operators), 231
User Debug Interface and files
  turning logging on or off, 121
user programs
  loading, 50

V
Variable window, 246
  enabling auto display update, 211, 213
variables
  displaying, 30, 71

displaying in decimal format, 208
displaying in floating-point format, 209
displaying in hexadecimal format, 209
displaying in string format, 210
editing, 31, 73, 204, 206, 208
tracing accesses, 33
tracing specified variable and stopping program execution, 166
tracing variable access, 166
version information, 160

W window contents
copying to the file, 40
Window-Arrange Icons (ALT, W, A) command, 158
Window-Backtrace (ALT, W, B) command, 141
Window-Cascade (ALT, W, C) command, 158
Window-Memory (ALT, W, M) command, 139
Window-Peripheral (ALT, W, P) command, 140
Window-Register (ALT, W, R) command, 139
Window-Source (ALT, W, S) command, 139
Window-Tile (ALT, W, T) command, 158
Window-Variable (ALT, W, V) command, 140

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.