
User's Guide

Publication number B3759-97022
July 2000

© Copyright Agilent Technologies 1994-2000
All Rights Reserved

Emulation Interface

Notice

Agilent Technologies 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. Agilent Technologies 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.

Agilent Technologies assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Agilent Technologies.

© Copyright 1994-2000 Agilent Technologies.

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

Windows 95 are registered trademarks of Microsoft Corporation.

UNIX is a registered trademark licensed by the X/Open Company Ltd. in the U.S.A. and other countries.

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

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

Printing History

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

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

Edition 1 B3759-97000, Sep 1998

Edition2 B3759-97004, Dec 1998

Edition3 B3759-97022, July 2000

Certification and Warranty

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

Contents

Overview

The Emulation Solution User Interface — Overview	19
In This Book	25

Concepts

Emulation Solution User I/F Window	29
------------------------------------	----

Debug Window	30
The Debug Window	31
The Source Window	34
The Register Window	35
The Memory Window	36
The Variable Window	38
The Peripheral Window	40
The Backtrace Window	41
The I/O Window	42

The PC Trace Window	43
---------------------	----

The Bus Trace Window	44
----------------------	----

1 Getting Started

Installation overview	47
-----------------------	----

Platform Requirements	48
-----------------------	----

Requirements for Windows95	48
Requirements for Windows NT	48
Requirements for HP-UX	48
Requirements for SunOS	49
Requirements for Solaris	49

Before Installing the Debugger	50
--------------------------------	----

For Windows95	50
For Windows NT	50
For HP-UX	51
Setting Up a LAN Connection	52
To obtain an IP address	53
To configure LAN parameters using the built-in terminal interface	54
To configure LAN parameters using BOOTP	57
To set the 10BASE-T configuration switches	60
To verify LAN communications	61
Setting Up a Serial Connection	62
To set the serial configuration switches	63
To connect a serial cable	64
To verify serial communications	65
Problems with the LAN Interface	66
If you cannot verify LAN communication	66
If you have LAN connection problems	67
If it takes a long time to connect to the network	68
Problems with the Serial Interface	69
If you cannot verify RS-232 communication	69
Installing the debugger software	70
Installation on PC (Windows 95/NT)	71
Installation on HP9000/700	74
Installation on Sun	75
Installation for SunOS	75
Installation for Solaris	77
Operation flow using the Demo program	80
Step 1. Start the debugger	80
Step 2. Set the hardware options	83
Step 3. Load the demo program	84
Step 4. Display the source file	85

Step 5. Set a breakpoint	86
Step 6. Run the demo program	87
Step 7. Delete the breakpoint	88
Step 8. Step over a function	89
Step 9. Single-step one line	90
Step 10. Run the program to a specified line	91
Step 11. Run the program until the current function returns	92
Step 12. Display a variable	93
Step 13. Edit a variable	94
Step 14. Display register contents	95
Step 15. Exit the debugger	96

2 Using the Debugger Interface

Starting and Exiting the Debugger 99

To start the Debugger	99
To exit the Debugger	102

Working with Emulation Solution User Interface Windows 103

To display windows from the Debug window	103
To display windows and dialog boxes from the PC Trace window	104
To display windows and dialog boxes from the Bus Trace window.	104
To copy window contents to a file	105
To set tabwidth in the Debug window	105

Using the Entry Buffer 106

History Function	106
------------------	-----

Using Action Buttons 107

Using Command Files 108

To create a command file	108
To execute a command file	109

3 Debugging Programs

Loading and Displaying Programs 113

To load user programs	114
To display source files by their names	115
To display source code specifying a heading line	115

To display source code from the current program counter	116
To display source code or mnemonics only	116
To display source code mixed with assembly instructions	117
To highlight source code	118
To specify source file directories	119
Running, Stepping, and Stopping the Program	120
To run the program from the current program counter	121
To run the program from a specified address	121
To run the program from the start address	122
To run the program from target reset	122
To run the program from the current program counter to a specifies address	122
To run the program until the current function returns	123
To step a single line or instruction from the current program counter	124
To step a single line or instruction from a specified address	125
To step a single line or instruction from the start address	125
To step over a function	126
To stop program execution	128
To reset the processor	129
Using Breakpoints	130
To set a breakpoint	131
To list the breakpoints	133
To disable a breakpoint	133
To delete breakpoints	134
Displaying and Editing Variables	135
To display a variable	135
To edit a variable	137
Displaying and Editing Memory	138
To display memory	139
To edit memory	141
To fill memory	142
Displaying and Editing Peripheral Registers	143
To display peripheral registers	143
To edit peripheral registers	144
Displaying and Editing Registers	145

To display registers	146
To edit registers	147
Saving and Loading Configurations	148
To save the current emulator configuration	148
To load the emulator configuration	149

4 Debug Window Commands and Window Control Menu Commands

Emulation Solution User I/F Main Window Commands	153
File -> Exit (ALT, F, X)	154
Open -> Debug Window (ALT, O, D)	154
Open->PC Trace Window... (ALT, O, P)	155
Open->Bus Trace Window... (ALT, O, B)	155
Settings->font (ALT, S, F) (For PC only)	156
Window->Tile (ALT, W, T) (For PC only)	156
Window->Cascade (ALT, W, C) (For PC only)	156
Window-> Arrange Icons (ALT, W, A) (For PC only)	156
Help->Contents (ALT, H, C)	157
Help->Search for Help on... (ALT, H, S)	157
Help->How to Use Help (ALT, H, H)	157
Help->Tutorial (ALT, H, T)	158
Help->Version... (ALT, H, V)	158
Debug Window Commands	159
File→Load→Configuration... (ALT, F, L, C)	161
File→Load→Program... (ALT, F, L, P)	162
File→Load→Environment... (ALT, F, L, E)	163
File->Load->Flash Operation... (ALT, F, L, F)	164
File→Store→Configuration... (ALT, F, S, C)	166
File→Store→Environment... (ALT, F, S, E)	166
File→Copy→Display... (ALT, F, P, D)	167
File→Log→Playback... (ALT, F, O, P)	168
File→Log→Record... (ALT, F, O, R)	170
Execution→Run→From PC (ALT, E, R, P)	171
Execution→Run→From () (ALT, E, R, F)	171
Execution→Run→From Start Address (ALT, E, R, S)	172
Execution→Run→From Reset (ALT, E, R, R)	172
Execution→Run→Until () (ALT, E, R, U)	173

Execution→Run→Return to Caller (ALT, E, R, C)	174
Execution→Step→From PC (ALT, E, S, P)	175
Execution→Step→From () (ALT, E, S, F)	175
Execution→Step→From Start Address (ALT, E, S, S)	176
Execution→Step→Over Procedure Call (ALT, E, S, O)	177
Execution→Break (ALT, E, B)	178
Execution→Reset (ALT, E, T)	178
Execution→Breakpoints... (ALT, E, P)	179
Display→Program At PC (ALT, D, P)	182
Display→Program At () (ALT, D, A)	182
Display→Source Files... (ALT, D, S)	183
Window→Source (ALT, W, S)	184
Window→Register (ALT, W, R)	184
Window→Memory (ALT, W, M)	184
Window→Variable (ALT, W, V)	185
Window→Peripheral (ALT, W, P)	185
Window→I/O (ALT, W, I)	185
Display→Find... (ALT, D, F)	186
Window→Backtrace (ALT, W, B)	187
Settings→Display Mode→Source Only (ALT, S, D, S)	188
Settings→Display Mode→Mixed (ALT, S, D, M)	188
Settings→Display Mode→Mnemonics Only (ALT, S, D, N)	189
Settings→Display Mode→Symbols (ALT, S, D, Y)	189
Settings→Display Mode→Highlight Source Lines (ALT, S, D, H)	190
Settings→Source View... (ALT, S, S)	191
Settings→Configuration→Hardware... (ALT, S, C, H)	193
Settings→Configuration→Access Size... (ALT, S, C, M)	193
Settings→Data Read From→Memory... (ALT, S, R, M)	194
Settings->Data Read From->Object File...(ALT, S, R, O)	194
Source Window Commands	195
File→Copy→Display(ALT,D,A)	196
File→Open(ALT,F,O)	196
File→Close(ALT,F,C)	196
Display→Program At () (ALT, D, A)	196
Display→Source Files... (ALT, D, S)	197
Display→Find... (ALT, D, F)	198
Settings→Display Mode→Source Only (ALT, S, D, S)	199
Settings→Display Mode→Mixed (ALT, S, D, M)	199
Settings→Display Mode→Mnemonics Only (ALT, S, D, N)	200
Settings→Display Mode→Symbols (ALT, S, D, Y)	200

Settings→Display Mode→Highlight Source Lines (ALT, S, D, H)	201
Settings→Data Read From→Memory(ALT,S,R,M)	201
Settings→Data Read From→Object File(ALT,S,R,O)	201
Register Window Commands	202
File→Copy→Display(ALT,D,A)	202
File→Open(ALT,F,O)	202
File→Close(ALT,F,C)	202
Settings→Auto Update (ALT, S, A)	203
Settings→Display Base→Binary(ALT,S,D,B)	203
Settings→Display Base→Octal(ALT,S,D,O)	203
Settings→Display Base→Decimal(ALT,S,D,D)	204
Settings→Display Base→Hexadecimal(ALT,S,D,H)	204
Memory Window Commands	205
File→Load(ALT,F,L)	206
File→Store(ALT,F,S)	206
File→Copy→Display(ALT,D,A)	207
File→Open(ALT,F,O)	207
File→Close(ALT,F,C)	207
Display→From () (ALT, D, F)	208
Display→Blocked→Byte (ALT, D, B, B)	208
Display→Blocked→Word (ALT, D, B, W)	208
Display→Blocked→Long (ALT, D, B, L)	208
Display→Absolute→Byte (ALT, D, A, B)	209
Display→Absolute→Word (ALT, D, A, W)	209
Display→Absolute→Long (ALT, D, A, L)	210
Display→Absolute→Float (ALT, D, A, F)	210
Modify→Memory... (ALT, M, M)	211
Settings→Auto Update (ALT, S, A)	213
Variable Window Commands	214
File→Copy→Display(ALT,D,A)	215
File→Open(ALT,F,O)	215
File→Close(ALT,F,C)	215
Display→Variable () (ALT, D, V)	215
Display→Address Of (ALT, D, A)	217
Display→Contents Of (ALT, D, C)	217
Modify→Variable... (ALT, M, V)	218
Settings→Display Base→Default (ALT, S, D, D)	219

Settings→Display Base→Decimal (ALT, S, D, C)	219
Settings→Display Base→Hexadecimal (ALT, S, D, H)	220
Settings→Display Base→Float (ALT, S, D, F)	220
Settings→Display Base→String (ALT, S, D, S)	221
Settings→Auto Update (ALT, S, A)	222

Peripheral Window Commands 223

I/O Window Commands 224

File→Copy→Display(ALT,D,A)	225
File→Open(ALT,F,O)	225
File→Close(ALT,F,C)	225
Settings→Set(ALT, S, S)	226
Settings→Display Base→Binary (ALT, S, D, B)	227
Settings→Display Base→Octal (ALT, S, D, O)	227
Settings→Display Base→Decimal (ALT, S, D, D)	227
Settings→Display Base→Hexadecimal (ALT, S, D, H)	227
Settings→Auto Update(ALT,S,A)	227

Backtrace Window Commands 228

File→Copy→Display(ALT,D,A)	228
File→Open(ALT,F,O)	228
File→Close(ALT,F,C)	228
Source at Stack Level	229

Debug Window Pop-up Commands 230

Set Software Breakpoint	230
Clear Software Breakpoint	230
Set Hardware Breakpoint	230
Clear Hardware Breakpoint	230
Run to Here	231
Evaluate ()	231

Source Window Pop-up Commands 232

Set Breakpoint	232
Clear Breakpoint	232
Set Hardware Breakpoint	232
Clear Hardware Breakpoint	232
Evaluate ()	233

Backtrace Window Pop-up Commands	234
Source at Stack Level	234

5 PC Trace Window Commands

File→Copy→Display...(ALT,F,P,D)	237
File→Close...(ALT,F,C)	237
Trace→Always (ALT, T, L)	238
Trace→After () (ALT, T, F)	238
Trace→About () (ALT, T, A)	239
Trace→Before () (ALT, T, B)	239
Trace→Until () (ALT, T, U)	240
Trace→Until Halt (ALT, T, H)	240
Trace→Condition... (ALT, T, C)	241
Trace→Again (ALT, T, G)	243
Trace→Halt (ALT, T, T)	243
Display→Trigger (ALT, D, T)	243
Display→Find... (ALT, D, F)	244
Trace→Trace Mode→Realtime	245
Trace→Trace Mode→Non-Realtime	245
Settings→Clock Speed (ALT, S, S)	245
Settings→Display Mode→Source Only (ALT, S, D, S)	246
Settings→Display Mode→Mixed (ALT, S, D, M)	246
Settings→Display Mode→Mnemonic Only (ALT, S, D, N)	246
Settings→Display Mode→Symbols (ALT, S, D, Y)	247
Settings→Display Mode→Function Names (ALT, S, D, F)	247
Settings→Display Mode→Line Numbers (ALT, S, D, L)	247
Settings→Display Mode→Highlight Source Lines (ALT, S, D, H)	249
Settings→Trace Count→Relative (ALT, S, C, R)	249
Settings→Trace Count→Absolute (ALT, S, C, A)	250
Settings→Data Read From→Memory (ALT, S, R, M)	250
Settings→Data Read From→Object File(ALT, S, R, O)	250

6 Bus Trace Interface

Bus Trigger Window Commands	253
File→Load→Program...(ALT,F,L,P)	254
File→Copy→Display...(ALT, F, P, D)	255
Trigger→Always (ALT, T, L)	256
Trigger→After () (ALT, T, F)	256

Trigger→About () (ALT, T, A)	256
Trigger→Before () (ALT, T, B)	256
Trigger→Until Halt (ALT, T, U)	257
Trigger→Condition... (ALT, T, C)	257
Trigger→Again (ALT, T, G)	257
Trigger→Halt (ALT, T, H)	257
Display→Program At() (ALT, D, A)	258
Display→Source Files (ALT, D, S)	258
Display→Find... (ALT, D, F)	259
Setting→Display Mode→Source Only (ALT, S, D, S)	260
Settings→Display Mode→Mixed (ALT, S, D, M)	260
Settings→Display Mode→Mnemonics Only (ALT, S, D, N)	260
Settings→Display Mode→Symbols (ALT, S, D, Y)	260
Settings→Display Mode→Highlight Source Lines (ALT, S, D, H)	260
Settings→Source View	261

Bus Trace Window 262

File→Copy→Display...(ALT, F, P, D)	263
Display→Trigger (ALT, D, T)	263
Display→Find String (ALT, D, F)	263
Display→Find G1 (ALT, D, I)	264
Display→Find G2 (ALT, D, N)	264
Setting→Display Mode→Source Only (ALT, S, D, S)	265
Settings→Display Mode→Mixed (ALT, S, D, M)	265
Settings→Display Mode→Mnemonics Only (ALT, S, D, N)	265
Settings→Display Mode→Symbols (ALT, S, D, Y)	265
Settings→Display Mode→Highlight Source Lines (ALT, S, D, H)	265
Settings→Trace Count→Relative (ALT, S, C, R)	266
Settings→Trace Count→Absolute (ALT, S, C, A)	266

7 Command File Command Summary

Run Control Commands	269
Variable and Memory Commands	269
Breakpoint Commands	270
Window Open Commands	270
Dialog Box Commands	270
Debug Window Configuration Commands	271
File Commands	271
PC Trace Commands	272
Miscellaneous Commands	272

Parameters 273

8 Expressions in Commands

Numeric Constants 277

Symbols 278

C Operators 279

9 Customizing the Emulation Solution User Interface

General information on customizing 283

Customizing global setting of Emulation Solution User Interface
284

Customizing each windows 285

Customizing the Debug Window 286

Customizing the Source Window 287

Customizing the Memory Window 287

Customizing the Register Window 288

Customizing the Peripheral Window 288

Customizing the Backtrace Window 289

Customizing the Variable Window 289

Customizing the PC Trace Window 290

Customizing the I/O Window 291

Customizing the Action Buttons 292

Customizing the Emulation Solution User Interface's Appearance (for
workstations) 293

Glossary

Index

Certification and Warranty 309

Certification 309

Warranty 309



Overview

Overview

This chapter describes the overview of the Emulation Solution User Interface Software. It also describes how each hardware of the Emulation Solution works with the functionality of the Emulation Solution User Interface Software.

- Overview
- Role of Emulation Probe
- Role of Emulation Module
- Role of Analysis Probe
- Role of Target Interface Module
- Role of Trace Port Analyzer

The Emulation Solution User Interface — Overview

The Emulation Solution User Interface is a multi-platform application that lets you debug the C language program for embedded microprocessor systems.

The Main window of the Emulation Solution User Interface provides three windows: Debug window, PC Trace window, and Bus Trace window.

The Debug window is used to control execution of the user application and to display the subwindows.

The PC Trace window provides the user interface to control trace of data captured from several external signal output pins supplied with the microprocessor.

The Bus Trace window displays the signals passing through the bus on the host computer screen to show how the user program runs on the target processor by using the Logic Analyzer connected to a LAN. It can also be used to setup the trigger settings under the specified conditions.

Note

To open the Debug window or PC Trace window, the Emulation Probe or the Emulation Module is required.

To open the Bus Trace window, the Analysis Probe is required.

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.

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 at the C function level.
- You can trace all the C statements which generated a write cycle to 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.
- You can trace a program calling a function.
- You can trace control flow within a function at the C statement level.

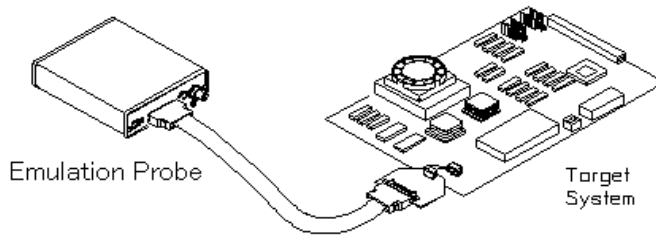
Debug your program while it runs continuously at full speed.

- 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.

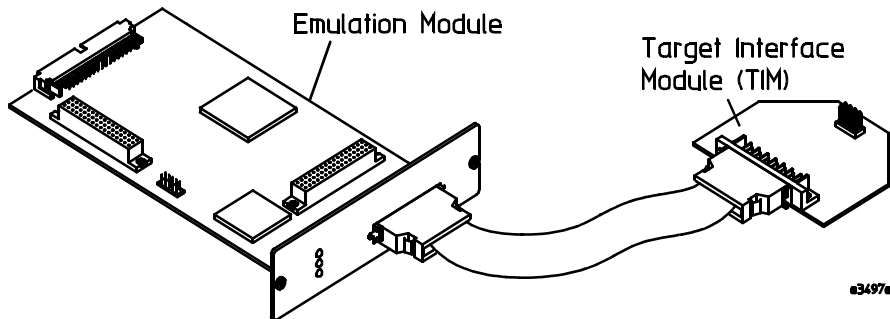
Role of Emulation Probe

The Emulation Probe enables you to download the program codes onto the flash memory, control execution of the program, and display or modify the memory and registers by using the on-chip debug functions built in the processor. It ensures that the program executes in real time on the actual target system however high the processor clock speed is



Role of Emulation Module

The Emulation Module is a hardware which can be incorporated into the Agilent Technologies 16600A/700A Series Logic Analyzer System. It has the same functions as the Emulation Probe described above.

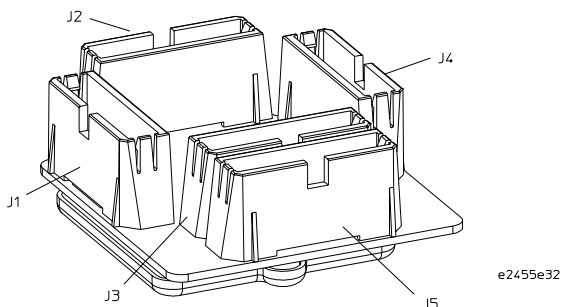


Role of Analysis Probe

The Analysis Probe is a module which captures the external bus signals from the processor and sends the data to the Logic Analyzer. The information captured from the Analysis Probe is passed through the Logic Analyzer and gateway to the Bus Trace Interface Software, a part of Emulation Solution User Interface Software, for processing. The information can be displayed and controlled on the user host screen.

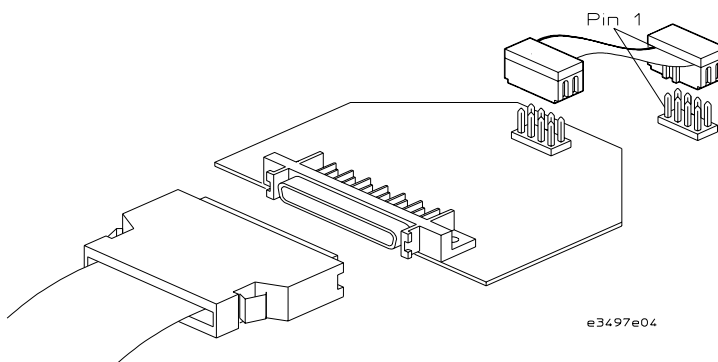
Bus Trace Interface Software

The Bus Trace Interface Software controls the external bus data sent to the Debugger through the Analysis Probe and the Logic Analyzer on the Debugger screen



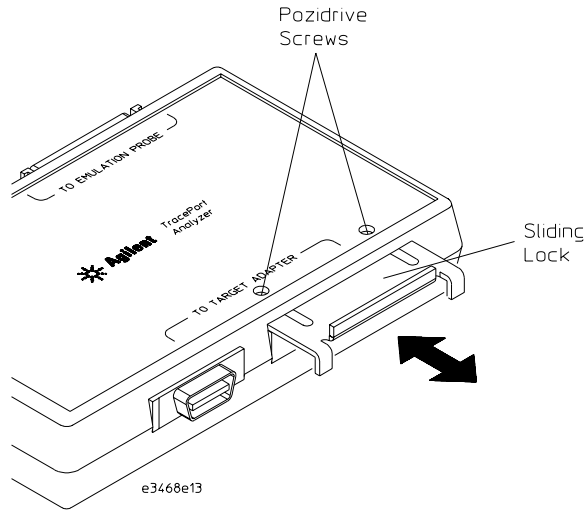
Role of Target Interface Module

The Target Interface Module is a circuit board which captures the signals from the connector mounted on the target system via the cable and transmits the captured signals to the Emulation Probe/Emulation Module via the flat cable



Role of Trace Port Analyzer

The Trace Port Analyzer works with the Emulation Probe or the Emulation Module and allows you to setup trace trigger, start and stop of acquisition of the trace data and so on under different conditions from the Emulation Probe or the Emulation Module



In This Book

This book contains the following eleven chapters. The contents of each chapter are described below.

Chapter 1 quickly shows you how to install the Emulation Solution User Interface Software and how to use the Debugger.

Chapter 2 shows you how to use Emulation Solution User Interface.

Chapter 3 shows you how to debug programs.

Chapter 4 describes commands that can be invoked from the Debug window as well as the windows that can be selected for display from the Debug window control menu.

Chapter 5 describes commands that appear in the PC Trace window.

Chapter 6 describes commands that appear in the Bus Trace window.

Chapter 7 describes commands that appear in pop-up menus.

Chapter 8 summarizes the debugger commands as they are used in command files.

Chapter 9 describes the format for expressions used in commands.

Chapter 10 contains conceptual (and more detailed) information on various topics.

Chapter 11 describes how to customize the runtime parameters of the C Debugger and the interface of each window.

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

Note

Concepts

Concepts

This chapter describes the following topics:

- Emulation Solution User I/F Window
- Debugger Windows
- PC Trace Windows
- Bus Trace Windows

Emulation Solution User I/F Window

The Emulation Solution User I/F window is used to connect to the Emulation Probe/Emulation Module or the Logic Analyzer and exit the Debugger. For the PC version, this window contains the control menus and other windows are displayed in this window.

For the WS version, this window is displayed as a single window containing the menu bar.

The following functions are available in the Emulation Solution User I/F window:

- Connecting to the Emulation Probe/Emulator Module.
- Connecting to the Logic Analyzer.
- Displaying the version
- Cascading and tiling windows, and rearranging icons (for PC version only).
- Displaying the online help.
- Exiting the Emulation Solution User Interface.

See Also

"Debug User I/F Window Commands" in Chapter 4, "Debug Window Commands and Window Control Menu Commands"

Debug Window

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

This section describes the following windows:

- The Debug 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 I/O window 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 <code><module_name>:<symbol_name></code> format.
Current PC	The line associated with the current program counter is highlighted.
" Marker	The breakpoint marker "" appears at the beginning of breakpoint lines.
EmulationStatus 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 and Window Control Menu Commands"

Chapter 11, "Customizing the Emulation Solution 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 7, "Window Pop-up Commands"

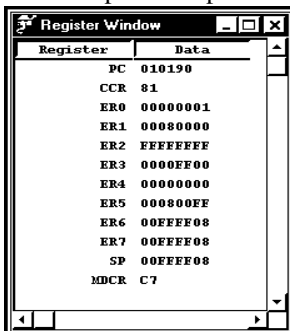
Chapter 11, "Customizing The Emulation Solution User Interface"

"Source Window Commands" in Chapter 4, "Debug Window Commands and 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.

Updating the register contents causes the user program to temporarily interrupt the monitor. If you want to run the program without such interruption, you can override interruption to the monitor.

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 (.munin.ini for the workstation version, or munin.ini for the PC version).

See Also

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

"Register Window Commands" in Chapter 4, "Debug Window Commands and Window Control Menu Commands"

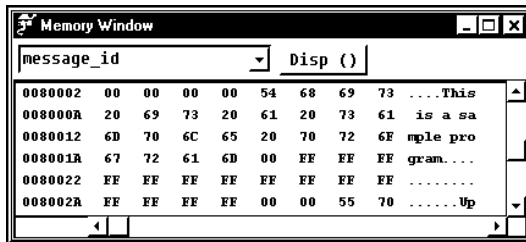
Chapter 11, "Customizing The Emulation Solution User Interface"

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 (.munin.ini for the workstation version, or munin.ini for the PC version).

See Also

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

"Memory Window Commands" in Chapter 4, "Debug Window Commands and Window Control Menu Commands"

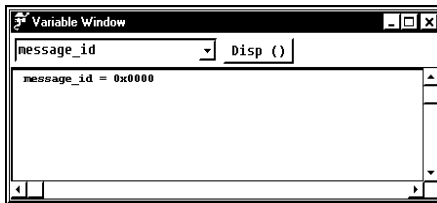
Chapter 11, "Customizing The Emulation Solution User Interface"

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 (.munin.ini for the WS version, or munin.ini for the PC version).

See Also

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

"Variable Window Commands" in Chapter 4, "Debug Window Commands and Window Control Menu Commands"

Chapter 11, "Customizing The Emulation Solution 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 *Emulation Solution 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 (.munin.ini for the workstation version, or munin.ini for the PC version).

See Also

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

"Peripheral Window Commands" in Chapter 4, "Debug Window Commands and Window Control Menu Commands"

Chapter 11, "Customizing The Emulation Solution User Interface"

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 4, "Debug Window Commands and Window Control Menu Commands"

Chapter 11, "Customizing The Emulation Solution User Interface"

The I/O Window

The I/O Window

The I/O window is used to display I/O value in the specified address.

You can open multiple instances of the I/O window.

To specify an address for which you want to display the I/O value, choose Settings → Set command to display the dialog box for specifying address. Enter the desired address in the dialog box. For display size of the I/O value, you can choose from three options: Byte, Word, and Long.

The PC Trace Window

The PC Trace window provides the user interface to the N-Trace functions which can be activated when the Emulation Probe or the Emulation Module is connected. Real time program flow trace is acquired through a trace port on the target CPU. The trace port analyzer captures program flow branch information. The interface software reconstructs the actual program flow and displays it on the PC Trace window.

The following functions are available from the PC Trace window:

- Setting various trace trigger.
- Displaying source files and disassembled instructions.
- Searching for strings.
- Entering miscellaneous settings.
- Setting trace mode.

The PC Trace 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.

The Bus Trace Window

The Bus Trace window enables you to direct the inverse assembler to easily capture the memory cycle and bus state with ADDRESS, DATA, and STATUS schemes. It can provide a source level external bus trace and triggering capability. You can see a complete source level trace display and can specify any trigger point by pointing to a source line in a source window.

The following functions are available from the Bus Trace window:

- Setting various bus trigger.
- Displaying source files and disassembled instructions.
- Searching for strings.
- Entering miscellaneous settings.

Getting Started

Getting Started

This chapter describes the installation of the Emulation Solution User Interface Software and the basic operation using the demo program.

Installation overview

- Installation overview
- Platform Requirements
- Before Installing the debugger
- Installation on PC
- Installation on HP9000/700
- Installation Sun

Operation flow using the Demo program

- | | |
|----------|--|
| 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. |

Installation overview

The installation procedure contains the following steps.

1 Before installation

Check that your computer meets the requirements for using the Emulation Solution User Interface. See the "Platform Requirements" section for details.

Then, set up the environment for using the Emulation Solution User Interface. See the "Before Installing the Debugger" section for details.

2 Setting up the Emulation Probe

If you're using the Emulation Probe for the first time, you must set parameters for connecting to the LAN. This step is done by connecting the Emulation Probe and the PC via RS-232 serial port. If the Emulation Probe is already on the LAN, you can skip this step. For details, see the documentation that comes with your Emulation Probe.

Logic Analyzer Setup

If you are using the Logic Analyzer for the first time, you must set parameters for connecting to the LAN. If the Logic Analyzer is already connected to the LAN, you can skip this step. For details, see the documentation that comes with your Logic Analyzer.

3 Connecting the Emulation Probe to the computer

Connecting the Logic Analyzer to the computer

4 Installing the Emulation Solution User Interface software

5 Verifying installation

Platform Requirements

Requirements for Windows95

- IBM PC compatible with a pentium processor.
- 32 Mbytes of memory.
- 10 Mbytes available disk space
- Windows95
- LAN card supported by Windows95 (supporting NDIS 3.0)
- Display with VGA or higher resolution

Requirements for Windows NT

Caution

For Windows NT 3.51, the administrative account is required to register an icon. (Without the above account, files can be downloaded, but the icon will not be created.) Requirements for Windows NT 4.0 are the same as for Windows 95.

Requirements for HP-UX

- HP-UX version 9.0 or greater.
- 32 Mbytes of memory (Over 64 Mbytes recommended)
- 20 Mbytes available disk space
- X11R5 X window system

Requirements for SunOS

- SunOs version 4.1.4 or greater
 - OpenWindows version 3.0 or greater
 - 20 Mbytes available disk space
 - 32 Mbytes of memory (Over 64 Mbytes recommended)
-

Requirements for Solaris

- Solaris version 2.3 or greater
- OpenWindows version 3.0 or greater
- 20 Mbytes available disk space
- 32 Mbytes of memory (Over 64 Mbytes recommended)

Before Installing the Debugger

For Windows95

- Install Windows95 on your PC according to its installation documentation.
- Follow the next steps to connect the Emulation Probe to the PC via LAN.

- 1 Install the LAN card supporting NDIS 3.0 into the PC.
- 2 Choose Control Panel in the Windows95 My Computer group and double-click the Network icon to open the Network dialog box. Configure the network using Configuration in the Network dialog box.

See the documentation that comes with Windows 95 documentation for details.

For Windows NT

- Install Windows NT on your PC according to its installation documentation.
- Follow the next steps to connect the Emulation Probe to the PC via LAN.

- 1 Install the LAN card supporting NDIS 3.0 into the PC.
- 2 Choose Control Panel in the Windows NT My Computer group and double-click the Network icon to open the Network dialog box. Configure the network using Configuration in the Network dialog box.

See the documentation that comes with Windows NT documentation for details.

For HP-UX

- Check the version of your HP-UX.
Installation procedure differs depending on your HP-UX version. You can check the version by executing the following command.

```
$ uname -a
```

Setting Up a LAN Connection

Introduction

You must connect the Emulation Probe used for the Emulation Solution User Interface via LAN.

The Emulation Probe has two LAN connectors:

- A BNC connector that can be directly connected to a IEEE 802.3 Type 10BASE2 cable (ThinLAN). When using this connector, the Emulation Probe provides the functional equivalent of a Medium Attachment Unit (MAU) for ThinLAN.
- An IEEE 802.3 Type 10BASE-T connector.

Use either the 10BASE2 or the 10BASE-T connector. Do *not* use both. The Emulation Probe will not work with both connected at the same time.

You must assign an IP address (Internet address) to the Emulation Probe before it can operate on the LAN. The IP address and other network parameters are stored in nonvolatile memory within the Emulation Probe.

The Emulation Probe automatically sets a subnet mask based on the subnet mask used by other devices on the network.

You can configure LAN parameters in any of the following ways:

- Using the built-in terminal interface.
- Using BOOTP. BOOTP is part of the HP-UX operating systems.

To obtain an IP address

- 1 Obtain the following information from your local network administrator or system administrator:

- An IP address for the Emulation Probe.
- The gateway address.

The gateway address is an IP address and is entered in integer dot notation. The default gateway address is 0.0.0.0, which allows all connections on the local network or subnet. If connections are to be made to workstations on other networks or subnets, this address must be set to the address of the gateway machine.

- 2 Find out whether port numbers 6470 and 6471 are already in use on your network.

The host computer interfaces communicate with the Emulation Probe through two TCP service ports. The default base port number is 6470. The second port has the next higher number (default 6471).

The default numbers (6470, 6471) can be changed if they conflict with some other products on your network. TCP service port numbers must be greater than 1024. If you change the base port, the new value must also be entered in the `/etc/services` file on the host computer. For example, you could modify the line:

```
hp64700 6470/tcp
```

To change the port numbers, see page 55. If you have already set the IP address, you can use a **telnet** connection instead of a serial connection to connect to the Emulation Probe.

- 3 Write down the link-level address of the Emulation Probe.

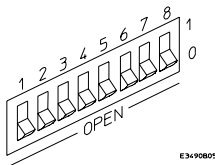
You will need this address if you use BOOTP to set the IP address.

The link-level address (LLA) is printed on a label above the LAN connectors on the Emulation Probe. This address is configured in each Emulation Probe shipped from the factory and cannot be changed.

To configure LAN parameters using the built-in terminal interface

- 1 Set configuration switches S1 through S4 to CLOSED, and set the other switches as appropriate for your serial interface.

Switch settings are printed on the bottom of the Emulation Probe. If you will use a baud rate of 9600 baud, set the switches like this:



- 2 Connect an ASCII terminal (or terminal emulator) to the Emulation Probe's RS-232 port with a 9-pin RS-232 cable.

Complete instructions for setting up a serial connection are described at "Setting Up a Serial Connection" in this chapter.

- 3 Plug in the Emulation Probe's power cord. Press the terminal's <RETURN> key a couple times. You should see a "R>", "p>" or "c>" prompt.

At this point, you are communicating with the Emulation Probe's built-in terminal interface.

- 4 Display the current LAN configuration values by entering the lan command:

```
R>lan
lan is disabled
lan -i 0.0.0.0
lan -g 0.0.0.0
lan -p 6470
Ethernet Address : 08000903212f
```

The "lan -i" line shows the current IP address (IP address) of the Emulation Probe.

The "Ethernet Address", also known as the link-level address, is preassigned at the factory, and is printed on a label above the LAN connectors.

5 Enter the following command:

```
lan -i <internet> [-g <gateway>] [-p <port>]
```

The lan command parameters are:

- i <internet> The IP address which you obtained from your network administrator.
- g <gateway> The gateway address.
This is optional. The default is 0.0.0.0.
- p <port> This changes the base TCP service port number.
The host computer communicates with the Emulation Probe via two TCP service ports. The default base port number is 6470. The second port number is the next higher value (default is 6471).

If you change the base port, the new value must also be entered in the /etc/services file on the host computer. For example, you could modify the line:

```
hp64700 6470/tcp
```

The default numbers (6470, 6471) can be changed if they conflict with some other products on your network. TCP service port numbers must be greater than 1024.

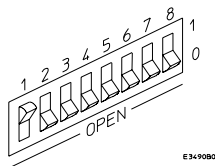
6 Disconnect the power cord from the Emulation Probe, and connect the the Emulation Probe to your network.

This connection can be made by using either the 10BASE-T connector or the 10BASE2 (BNC) connector on the Emulation Probe. Do not use both connectors at the same time.

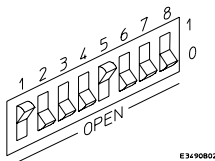
7 Set the configuration switches to indicate the type of connection that is to be made.

Switch S1 must be set to OPEN, indicating that a LAN connection is being made.

Switch S5 should be 1 if you are connecting to the BNC connector :



Switch S5 should be 0 if you are connecting to the 10BASE-T connector:



Set all other switches to CLOSED.

- 8 Connect the power cord to the Emulation Probe.
- 9 Verify your Emulation Probe is now active and on the network. See "To verify LAN communications" in this chapter.

Once you have set a valid IP address, you can use the **telnet** utility to connect to the Emulation Probe, and use the **lan** command to change LAN parameters.

Example

For example, to assign an IP address of 192.6.94.2 to the Emulation Probe, enter the following command:

```
R>lan -i 192.6.94.2
```

The IP address and any other LAN parameters you change are stored in nonvolatile memory and will take effect the next time the Emulation Probe is powered off and back on again.

See Also

"Problems with the LAN Interface" in this chapter, if you have problems verifying LAN communication.

To configure LAN parameters using BOOTP

This method is applicable only if your HP-UX workstation is already running **bootpd**, the BOOTP daemon. The **ipconfig700** command does the same thing as BOOTP and is easier to use.

The BOOTP software is shipped with HP-UX version 8.0 or later. For the earlier version of HP-UX, contact Hewlett-Packard.

1 Make sure that your host computer supports BOOTP.

If the following commands yield the results shown below, your machine supports the BOOTP protocol.

```
$ grep bootp /etc/services
bootps      67/udp
bootpc      68/udp
$ grep bootp /etc/inetd.conf
bootps dgram udp wait    root  /etc/bootpd  bootpd
```

If the commands did not yield the results shown, you must either add BOOTP support to your workstation or use a different method to configure the Emulation Probe LAN parameters.

2 Add an entry to the host BOOTP database file, /etc/bootptab. For example:

```
# Global template for options common to all Agilent
Technologies 64700 emulators.
# Gateway addresses can be specified differently if
# necessary.

hp64700.global:\
    :gw=0.0.0.0:\
    :vm=auto:\
    :hn:\
    :bs=auto:\
    :ht=ether

# Specific emulator entry specifying hardware address
# (link-level address) and ip address.

hprobe.div.hp.com:\
    :tc=hp64700.global:\
    :ha=080009090B0E:\
    :ip=192.6.29.31
```

In the example above, the "ha=080009090B0E" identifies the link-level address of the Emulation Probe.

The "ip=192.6.29.31" specifies the IP address that is assigned to the Emulation Probe. The node name is "hprobe.div.hp.com".

For additional information about using bootpd, refer to the HP-UX man pages.

3 Connect the Emulation Probe to your network.

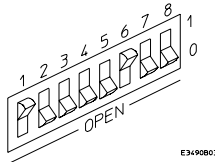
This connection can be made by using either LAN connector on the Emulation Probe.

4 Set the configuration switches to indicate the type of connection that is to be made.

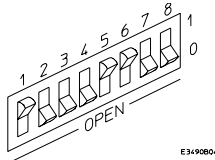
Switch S1 must be set to OPEN, indicating that a LAN connection is being made.

Switch S6 must be set to OPEN to enable BOOTP mode.

Switch S5 should be set to CLOSED if you are connecting to the BNC connector



Switch S5 should be set to OPEN if you are connecting to the 10BASE-T connector.



Set all other switches to CLOSED.

5 Connect the power cord to the Emulation Probe.

Verify that the power light stays on after 10 seconds.

The IP address will be stored in EEPROM.

6 Set switch S6 back to CLOSED and connect the power cable again.

Do this so that the Emulation Probe does not request its IP address each time power is cycled. The IP address is stored in EEPROM, so BOOTP does not need to be run again. Leaving this switch on will result in slower performance, increased LAN traffic (if the BOOTP server becomes inactive).

7 Verify your Emulation Probe is now active and on the network. See "To verify LAN communications" in this chapter.

To set the 10BASE-T configuration switches

Set switches S7 and S8 to CLOSED unless one of the following conditions is true:

- If the LAN cable exceeds the standard length, set switch S7 to OPEN.

The Emulation Probe has a switch-selectable, twisted-pair receiver threshold. With switch S7 set to OPEN, the twisted-pair receiver threshold is lowered by 4.5 dB. This should allow you to use cable lengths of up to about 200 meters. If you use a long cable, you should consult with your LAN cabling installer to ensure that:

- The device at the other end of the cable has long cable capability, and
 - The cable is high-grade, low-crosstalk cable with crosstalk attenuation of greater than 27.5 dB.
- If your network doesn't support LINK BEAT integrity checking or if the Emulation Probe is connected to a non 10BASE-T network set this switch to LINK BEAT OFF (0 or OPEN).

In normal mode (switch S8 set to CLOSED), a link integrity pulse is transmitted every 15 milliseconds in the absence of transmitted data. It expects to receive a similar pulse from the remote MAU. This is the standard link integrity test for 10BASE-T networks. If your network doesn't support the LINK BEAT integrity checking or if the Emulator is used on a non 10BASE-T network set this switch to LINK BEAT OFF (OPEN).

Note

Setting switch S8 to OPEN when Link Beat integrity checking is required by your network will cause the remote MAU to disable communications.

To verify LAN communications

- 1 Verify your Emulation Probe is now active and on the network by issuing a telnet to the IP address.

This connection will give you access to the Emulation Probe's built-in terminal interface.

- 2 To view the LAN parameters, enter the `lan` command at the terminal interface prompt.
- 3 To exit from this telnet session, type `<CTRL>D` at the prompt.

The best way to change the Emulation Probe's IP address, once it has already been set, is to telnet to the Emulation Probe and use the terminal interface `lan` command to make the change. Remember, after making your changes, you must cycle power or enter a terminal interface `init -p` command before the changes take effect. .

ExampleSetting switch S8 to OPEN when Link Beat integrity checking is required by your network will cause the remote MAU to disable communications.

Note

If you encounter problems, refer to the "Problems with the LAN Interface" in this chapter.

Example

```
$ telnet 192.35.12.6
R>lan
lan is enabled using TP
  lan -i 192.35.12.6
  lan -g 0.0.0.0
  lan -p 6470
Subnet Mask: 255.255.255.0
Ethernet Address: 0800F090B30
```

Setting Up a Serial Connection

To set up a serial connection, you will need to:

- Set the serial configuration switches
- Connect the Agilent Technologies Emulation Probe to the RS-232 interface
- Connect a serial cable between the host computer and the Emulation Probe
- Verify communications

Serial connections on a workstation

You should not use a serial connection on a workstation, except to set LAN parameters.

Serial connections on a PC

You should not use a serial connection on a PC, except to set LAN parameters or to update the Agilent Technologies E3472A/72D/73A firmware.

To set the serial configuration switches

- 1 Set switch S1 to 1 (RS-232).
- 2 Set switches S2-S4 to 1.
- 3 Set switch S5 to 1 (HW HANDSHAKE ON) if your serial interface uses the DSR/DTR lines for flow control. Set S5 to 0 (HW HANDSHAKE OFF) if your serial interface uses software flow control (XON/XOFF).

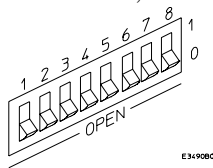
If your serial interface supports hardware handshaking, you should use it (set switch S5 to CLOSED). Hardware handshaking will make the serial connection much more reliable.

- 4 Set switches S6-S8 for the baud rate you will use. These switch settings are listed on the bottom of the Emulation Probe.

The higher baud rates may not work reliably with all hosts and user interfaces. Make sure the baud rate you choose is supported by your host and user interface.

Example

To use a baud rate of 9600 baud, set the switches as follows:



To connect a serial cable

Use the grounded and shielded cable.

If the cable is not shielded, or if the cable is not grounded at the serial controller, the Emulation Probe may be damaged by electrostatic discharge.

Connect an RS-232C modem cable from the host computer to the Emulation Probe. The recommended cable is Hewlett-Packard part number C2932A. This is a 9-pin cable with one-to-one pin connections.

Note Use the recommended cable. If the cable is not shielded, or if the cable is not grounded at the serial controller, the Emulation Probe may be damaged by electrostatic discharge.

To verify serial communications

- 1 Start a terminal emulator program on the host computer.

If you are using a PC, the Terminal application (HyperTerminal) in Microsoft Windows will work fine.

If you are using a UNIX workstation, you can use a terminal emulator such as kermit.

- 2 Plug the power cord into the Emulation Probe.

When the Emulation Probe powers up, it sends a message (similar to the one that follows) to the serial port and then displays a prompt:

```
Copyright (c) Agilent Technologies 1987
All Rights Reserved.  Reproduction, adaptation, or translation without prior
written permission is prohibited, except as allowed under copyright laws.
```

```
E3499B Series Emulation System
Version:  A.07.00 17Aug96
Location:  Generics
```

```
E3468A Toshiba TX19/39 Series Emulator
Version:  A.01.00 17Aug96
```

```
R>
```

The version numbers may be different for your Emulation Probe.

- 3 Press the Return or Enter key a few times.

You should see a prompt such as "R>", "p>" or "c>".

See Also

"Problems with the Serial Interface" in this chapter.

Problems with the LAN Interface

If you cannot verify LAN communication

Use the "telnet" command on the host computer to verify LAN communication. After powering up the Emulation Probe, it takes up to a minute before the Emulation Probe can be recognized on the network. After a minute, try the "telnet <internet address>" command.

If "telnet" does not make the connection:

- Make sure that you have connected the Emulation Probe to the proper power source and that the power light is lit.
- Make sure that the LAN cable is connected. Refer to your LAN documentation for testing connectivity.
- Make sure that only one of the LAN ports is connected.
- Make sure the Emulation Probe communication configuration switches are set correctly. Unplug the Emulation Probe power cord, then plug it in again to make sure the switch settings are read correctly by the Emulation Probe.
- Make sure that the Emulation Probe's IP address is set up correctly. Use the RS-232 port to verify this that the IP address is set up correctly. When you are connected to the RS-232 port, run performance verification on the Emulation Probe's LAN interface with the "pv" command.

If "telnet" makes the connection, but no prompt (for example, R>, M>, U>, etc.) is supplied:

- It's possible that the host software is in the process of running a command. You can use <CTRL>c to interrupt and get the Terminal Interface prompt.
- It's also possible for there to be a problem with the Emulation Probe firmware while the LAN interface is still up and running. In this case, you must reboot the Emulation Probe by disconnecting power to the Emulation Probe and reconnecting it again.

If you have LAN connection problems

- ❑ Try to "ping" the Emulation Probe.

```
ping <hostname or IP address>
```

If it does not respond:

1. Check that switch S1 is "0" (attached to LAN, not RS-232).
2. Check that switch S5 is in the correct position for your LAN interface (either 10BASE2 or 10BASE-T).

(Remember: if you change any switch settings, the changes do not take effect until you cycle power.)

- ❑ If the Emulation Probe still does not respond to a "ping", you need to verify the IP address and gateway mask of the Emulation Probe. To do this, connect the Emulation Probe to a terminal or terminal emulator, change the switch settings so it is connected to RS-232, and enter the "lan" command. The output looks something like this:

```
lan -i 15.5.24.116  
lan -g 15.5.23.1  
lan -p 6470  
Ethernet Address : 08000909BAC1
```

"lan -i" shows the internet address is 15.5.24.116 in this case. If the Internet address (IP) is not what you expect, you can change it with the 'lan -i <new IP>' command.

"lan -g" shows the gateway address. Make sure it is the address of your gateway if you are connecting from another subnet, 0.0.0.0 if you are connecting from the local subnet.

"lan -p" shows the port is 6470. If the port is not 6470, you must change it with the "lan -p 6470" command (unless you have deliberately set the port number to a different value because of a conflict).

If it takes a long time to connect to the network

- ❑ Check the subnet masks on the other LAN devices connected to your network. All of the devices should be configured to use the same subnet mask.

Subnet mask error messages do not indicate a major problem. You can continue using the Emulation Probe.

The Emulation Probe automatically sets its subnet mask based on the first subnet mask it detects on the network. If it then detects other subnet masks, it will generate error messages.

If there are many subnet masks in use on the local subnet, the Emulation Probe may take a very long time to connect to the network after it is turned on.

Problems with the Serial Interface

If you cannot verify RS-232 communication

If the Emulation Probe prompt does not appear in the terminal emulator window:

- Make sure that you have connected the Emulation Probe to the proper power source and that the power light is lit.

- Make sure that you have properly configured the data communications switches on the Emulation Probe and the data communications parameters on the host computer. You should also verify that you are using the correct cable.

The most common type of data communications configuration problem involves the configuration of the Emulation Probe as a DTE device instead of as a DCE device. If you are using the wrong type of cable, no prompt will be displayed.

Caution

Use the recommended cable. If the cable is not shielded, or if the cable is not grounded at the serial controller, the Emulation Probe may be damaged by electrostatic discharge (Recommended cable part number is Hewlett-Packard C2932A).

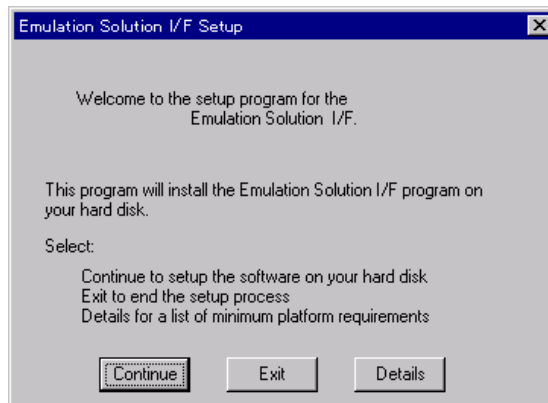
Installing the debugger software

This chapter shows you how to install the Debugger Software.

- Installation on PC(Windows 95/NT)
- Installation on HP9000/700 Workstation
 - Installation for HP-UX version 9.x
 - Installation for HP-UX version 10.x
- Installation on Sun Workstation
 - Installation for SunOS
 - Installation for Solaris

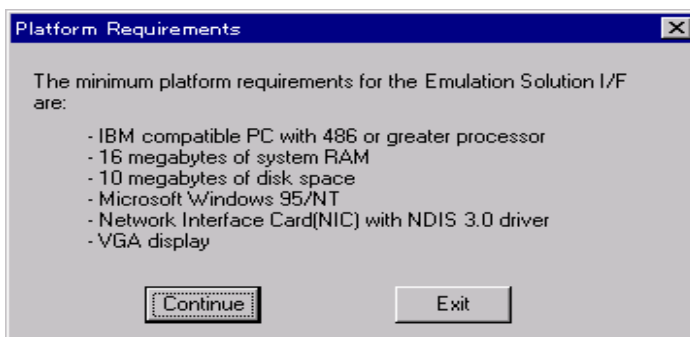
Installation on PC (Windows 95/NT)

- 1 Start Windows 95/NT.
- 2 Put the Emulation Solution User Interface CD-ROM into the CD-ROM drive.
- 3 From the Windows 95/NT Start menu, choose the Windows Explorer -> CD-ROM drive. The dialog box appears and double-click the "setup" application. Then the installation program starts.



- 4 Press the Continue button to continue the installation. Follow the instructions on your screen. If you decided to cancel the installation, press the Exit button.

Pressing the Details button shows the minimum platform requirements for the Emulation Solution User Interface.

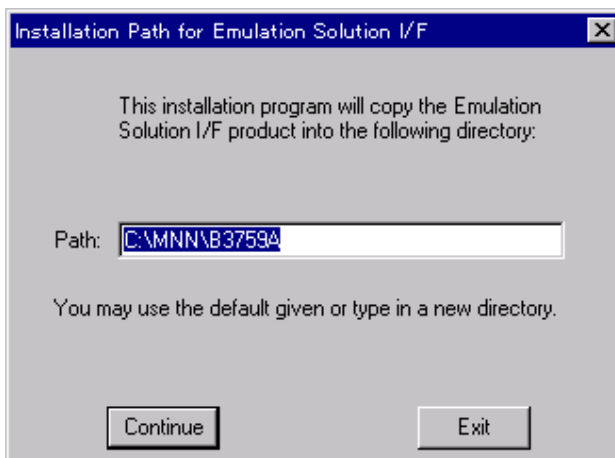


Press the Continue to return to the previous screen for continuing the installation. If you decided to cancel the installation, press the Exit button.

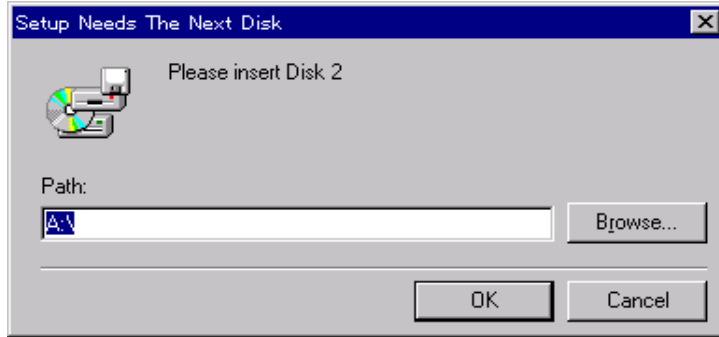
The dialog box, prompting you to enter the installation path, appears. The default installation path is C:\hpmnn\B3759a.

Enter the installation path and press the Continue button. Files will be copied to your hard disk.

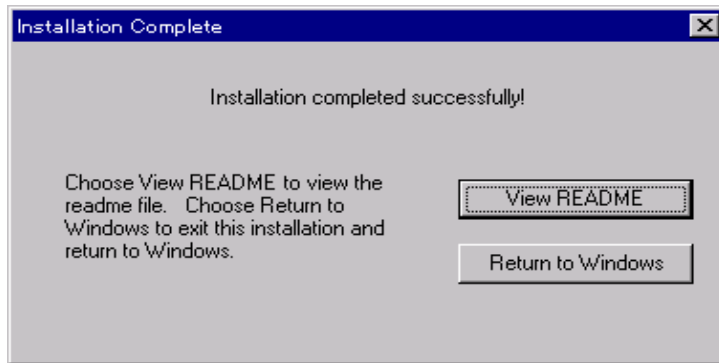
During the installation process, the following dialog box appears. This software installs the program from the CD-ROM, therefore press OK button to continue.



Enter the LAN address for the emulator you use with this software. This enables you to connect the debugger to the emulator immediately after you start the software.



When the installation is successfully completed, the following dialog box appears. Press the View README button to view the README file.



- 5 Choose B3759A from the Emulation Solution User Interface group to verify the installation.

Installation on HP9000/700

This section describes how to install the Debugger Software on the HP 9000 Series 700 computer running under the HP-UX operating system version 9.x and 10.x.

See the information on updating in your HP-UX documentation for how to install the software. Generally the steps you take are those shown below.

- 1 Log in to the system on which you want to install the software as root.
- 2 Put the CD-ROM into the CD-ROM drive that will be the *source device* for the installation.
- 3 Enter the following command to set the mask.
umask 0
- 4 Move to the directory where the software is installed.
Example: `cd /usr`
- 5 Start installing the software by entering
`<cdrom_path> /install`
- 6 When the installation is successfully completed, add
"/usr/hpmnn/hp700_9/bin" (If the install base directory was /usr) to your PATH environment variable.
- 7 Enter the following command to check that the software starts normally.

```
$ munin &
```

Installation on Sun

Installation for SunOS

- 1 Log in the system on which you want to install as root.
- 2 Put the CD-ROM into the CD-ROM drive that will be the *source device* for the installation.
- 3 Enter the following command to set the mask.
umask 0
- 4 Move to the directory where the software is installed.
Example: `cd /usr`
- 5 Start installing the software by entering
`<cdrom_path> /install`
- 6 Run the configure script. See "Running the configure script" section below for details.
- 7 Set the PATH environment variable to include the /usr/hpmnn/sunos_4/bin (If the install base directory was /usr) directory.
- 8 Enter the following command to check that the software starts normally.

```
$ munin &
```

Running the configure script

To run the **configure** script, you must change to the `/usr/hpmnn/sunos_4` directory:

```
cd /usr/hpmnn/sunos_4
./configure
```

The stand-alone configure script should be run on the workstation where you originally installed the software and again on each workstation which NFS mounts the `/usr/hpmnn` install tree (and has a local file system). The configure script may be run as root at any time to verify that all files required by the software are linked to the correct locations on the local file system.

The configure script does the following:

- Installs the XKeysymDB file if the required OSF keymappings do not exist in `$OPENWINHOME/lib/XKeysymDB`.
`$OPENWINHOME` is `/usr/openwin` by default.

Example

Assume the software product was installed on the file server `snow_white` in the directory `/home/snow_white`.

There are seven workstations who want to share `snow_white`'s software install directory. To share this directory, you must do the following for each dwarf workstation:

- 1 Log in to the dwarf's system as root.
- 2 Mount `snow_white`'s software install directory:

```
mount snow_white:/home/snow_white/usr/hpmnn /usr/hpmnn
```

- 3 Change directories to `/usr/hpmnn/sunos_4`, and run the configure script to link the necessary software files onto your local file system. The configure script is verbose and will tell you which files failed to link. Follow the instructions given by the configure script, and keep re-running it until it succeeds.

```
cd /usr/hpmnn/sunos_4
./configure
```

Installation for Solaris

- 1 Log in to the system on which you want to install the software as root.
- 2 Put the CD-ROM into the CD-ROM drive that will be the *source device* for the installation.
- 3 Enter the following command to set the mask.

```
umask 0
```

- 4 Move to the directory where the software is installed.
Example: `cd /opt`

- 5 Start installing the software by entering

```
<cdrom_path> /install
```

- 6 Run the configure script. See the "Running the configure script" section below for details.
- 7 Set the PATH environment variable to include the /opt/hpmnn/solar_2/bin (If the install base directory was /opt) directory.
- 8 Set the MOTIFHOME environment variable. See the "Running the configure script" section below for details.
- 9 Enter the following command to check that the software starts normally.

```
$ munin &
```

Running the configure script

To run the **configure** script, you must change to the `/opt/hpmmn/solar_2` directory:

```
cd /opt/hpmmn/solar_2
./configure
```

For the environmental variable `MOTIFHOME`, specify the directory where your Motif package software resides.

Example

Set the environmental variable `MOTIFHOME` as follow:

```
setenv MOTIFHOME /opt/SUNWmotif
          (SunSoft OSF/Motif 1.2.2)
setenv MOTIFHOME /usr/dt
          (SunSoft OSF/Motif 1.2.3)
setenv MOTIFHOME /Motif1.2.2/usr
          (IXI Motif 1.2.2)
```

The stand-alone configure script should be run on the workstation where you originally installed the software and again on each workstation which NFS mounts the `/opt/hpmmn` install tree (and has a local file system). The configure script may be run as root at any time to verify that all files required by the software are linked to the correct locations on the local file system.

The configure script does the following:

- 1 Checks for the existence of the SunSoft OSF/Motif 1.2.2 or greater or the IXI Motif 1.2.2 or greater product under `$MOTIFHOME`. If `$MOTIFHOME` has not been defined, the script looks under `/opt/SUNWmotif` (for Motif 1.2.2) or under `/usr/dt` (for Motif 1.2.3).
`$OPENWINHOME` is `/usr/openwin` by default.
- 2 For SunSoft OSF/Motif 1.2.3 only, creates symbolic links between versions 2 and 3 of the Motif shared libraries.

Example

Assume the software product was installed on the file server *snow_white* in the directory */home/snow_white*.

There are seven workstations who want to share *snow_white*'s software install directory. To share this directory, you must do the following for each dwarf workstation:

- 1 Log in to the dwarf's system as root.
- 2 Mount *snow_white*'s software install directory:

```
mount snow_white:/home/snow_white/usr/hpmnn/usr/hpmnn
```

- 3 Change directories to */opt/hpmnn/solar_2*, and run the configure script to link the necessary files onto your local file system. The configure script is verbose and will tell you which files failed to link. Follow the instructions given by the configure script, and keep re-running it until it succeeds.

```
cd /opt/hpmnn/solar_2  
./configure
```

Operation flow using the Demo program

Step 1. Start the debugger

For PC:

- 1 Choose Agilent Technologies B3759A #XXX in the Emulation Solution Interface group from the Windows 95/NT Start menu.

Or:

- a. Click the Windows 95/NT Start button and choose the Run (ALT, S, R) command.
- b. Enter the debugger startup file name
C:\hpmnn\B3759a\munin.exe (if C:\hpmnn\B3759a\ was the installation path chosen during installation).



- 2 Choose a window to connect from the Connect menu.

You may choose "Emulator...", "PC Trace..." or "Logic Analyzer..."

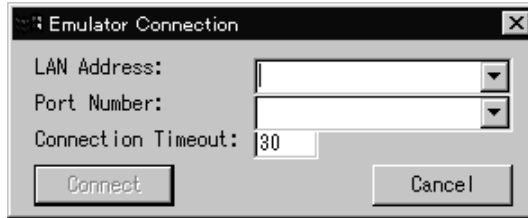
A Debug window opens upon establishing a connection with a emulation probe/module when you choose to connect to Emulator or PC Trace.

Bus Trace window opens upon a connection with your logic analyzer.

Note

Some windows may not open depending on your system configuration.

3 Choose open the Debug window or PC Trace window, choose Debug Window... or PC Trace Window... from the Open menu. Then the Connection dialog box appears. In the Connection dialog box, enter the information of the hardware you use (IP address/port of the Emulation Probe or Emulaton Module).

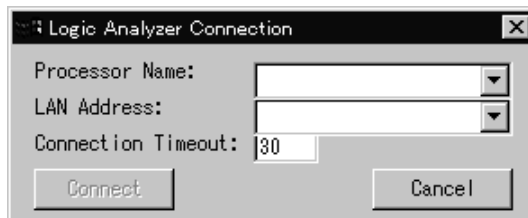


4 When you complete, click the Connect button.

If connection is made successfully, the Debug window or the PC Trace window opens.

5 If you choose Bus Trace Window... from the Open menu of the Main window, the Connection dialog box for the Logic Analyzer appears.

In the Connection dialog box, enter the name of the processor you use and the IP address/ port information of the Logic Analyzer.



6 When you complete, click the Connect button.

If connection is made successfully, the Bus Trigger window and the Bus Trace window open.

For workstations:

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

```
$ munin &
```

2 Unlike the PC version of the software, the main window in the WS version only equips with pull-down menus.

3 Choose the window to be connected from the Open menu of the Main window. You can choose from the three options;

Debug Window..., PC Trace Window, or Bus Trace Window.

According to your choice, the Debug window, PC Trace window, or Bus Trace window opens.

Note

Note that some windows can not open depending on your hardware environment.

4 When you choose Debug Window... or PC Trace Window, the Connection dialog box appears.

In the Connection dialog box, enter the IP address/port information of user's hardware (Emulation Probe/Module).

5 When you complete, click the Connect button.

If connection is made successfully, the Debug window or the PC Trace window opens.

6 If you choose Bus Trace Window... from the Open menu of the Main window, the Connection dialog box for the Logic Analyzer appears.

In the Connection dialog box, enter the name of the processor you use and the IP address/port information of the Logic Analyzer.

7 When you complete, click the Connect button. If connection is made successfully, the Bus Trigger window and the Bus Trace window opens.

Step 2. Set the hardware options

The Emulation Solution 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 file resides in the following directory (if `C:\hpmnn\B3759a` (for PC) or `/usr/hpmnn/hp700_9` (for workstations) was the installation path chosen during installaton):

For PC: `C:\hpmnn\B3759a\samples\800`

For WS: `/usr/hpmnn/hp700_9/samples/800`

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. Load the demo program

- 1 Choose the File→Load→Program... (ALT, F, L, P) command from the Debug window menu.

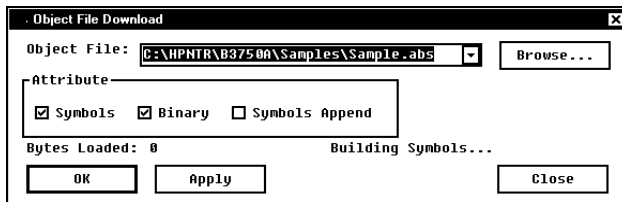
The Object File Download dialog box appears.

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

The file resides in the following directory (if C:\hpmnn\b3759A (for PC) or /usr/hpmnn/hp700_9 (for workstations) was the installation path chosen during installation):

For PC: C:\hpmnn\B3759A\samples\800

For workstations: /usr/hpmnn/hp700_9/samples/800



- 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 4. Display the source file

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

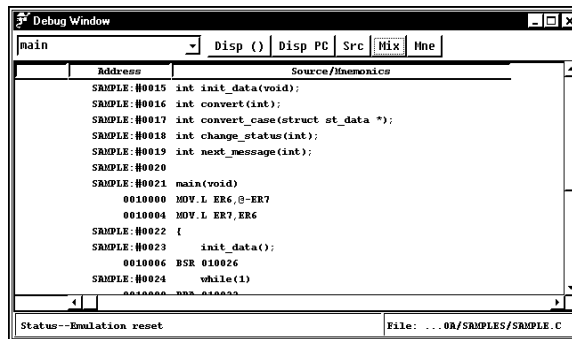
- 1 Confirm that "main" is displayed in the entry buffer of the Debug window.

You can define the initial value of the entry buffer in the initialization file. See Chapter 10, "Customizing the 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.

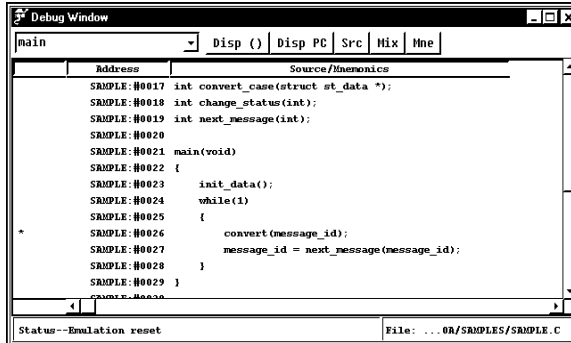
Step 5. Set a breakpoint

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.



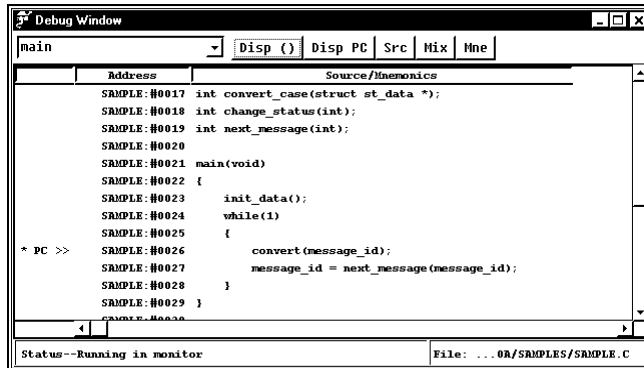
Note that the line containing "convert" is marked with an "*", indicating that a breakpoint has been set on the line.

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

Step 6. 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 7. Delete the breakpoint

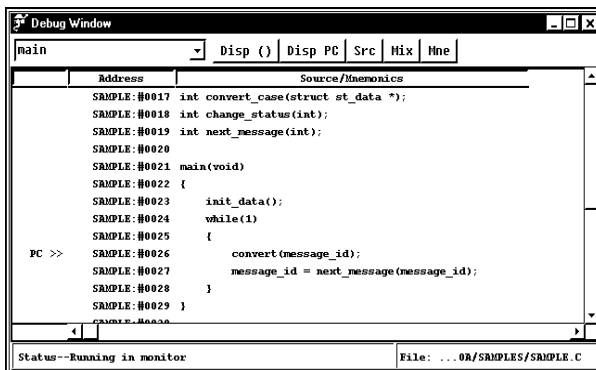
To delete the breakpoint you set in step 5:

- 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 8. 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.

Address	Source/Mnemonic
SAMPLE:#0017	int convert_case(struct st_data *);
SAMPLE:#0018	int change_status(int);
SAMPLE:#0019	int next_message(int);
SAMPLE:#0020	
SAMPLE:#0021	main(void)
SAMPLE:#0022	{
SAMPLE:#0023	init_data();
SAMPLE:#0024	while(1)
SAMPLE:#0025	{
SAMPLE:#0026	convert(message_id);
PC >>	message_id = next_message(message_id);
SAMPLE:#0028	}
SAMPLE:#0029	}

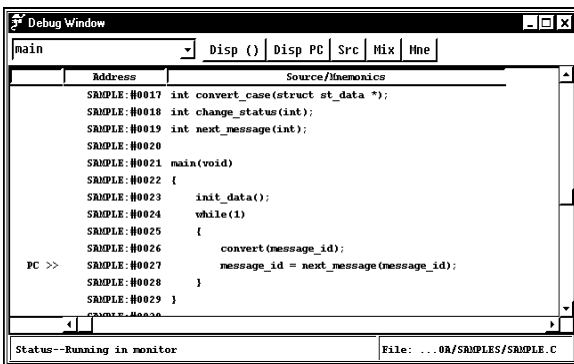
Status--Running in monitor File: ...OR/SAMPLES/SAMPLE.C

Step 9. 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.



The screenshot shows a 'Debug Window' with a menu bar containing 'main', 'Disp ()', 'Disp PC', 'Src', 'Mix', and 'Mne'. Below the menu is a table with two columns: 'Address' and 'Source/Mnemonic'. The source code is as follows:

```
SRMPLE:#0017 int convert_case(struct st_data ");
SRMPLE:#0018 int change_status(int);
SRMPLE:#0019 int next_message(int);
SRMPLE:#0020
SRMPLE:#0021 main(void)
SRMPLE:#0022 {
SRMPLE:#0023     init_data();
SRMPLE:#0024     while(1)
SRMPLE:#0025     {
SRMPLE:#0026         convert(message_id);
PC >> SRMPLE:#0027         message_id = next_message(message_id);
SRMPLE:#0028     }
SRMPLE:#0029 }
```

At the bottom of the window, the status bar reads 'Status--Running in monitor' and the file path is 'File: ...OR/SAMPLES/SRMPLE.C'.

Step 10. 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 screenshot shows a 'Debug Window' with a menu bar containing 'main', 'Disp ()', 'Disp PC', 'Src', 'Mix', and 'Mne'. The main area displays a table with two columns: 'Address' and 'Source/lineonics'. The code is as follows:

```

SRMPLE:#0062
SRMPLE:#0063 next_message(int id) /* Change message to be conve
SRMPLE:#0064 {
PC >> SRMPLE:#0065     if(id == MESSAGE1)
SRMPLE:#0066         return(MESSAGE2);
SRMPLE:#0067     else
SRMPLE:#0068         return(MESSAGE1);
SRMPLE:#0069 }
  
```

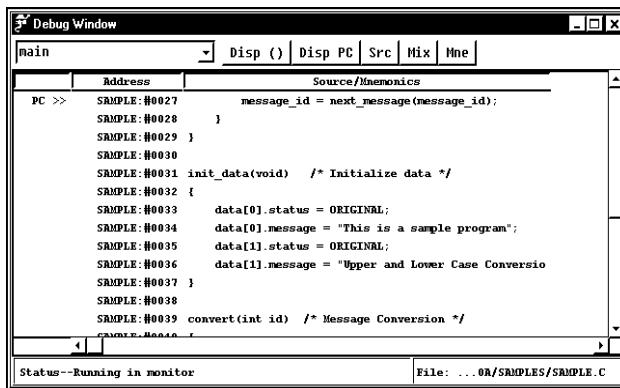
The status bar at the bottom indicates 'Status--Running in monitor' and 'File: ...OR/SRMPLES/SRMPLE.C'.

Step 11. 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".



```
Debug Window
main
Disp () | Disp PC | Src | Mix | Mne
+-----+-----+
| Address | Source/Mnemonics |
+-----+-----+
PC >> | SAMPLE:#0027 | message_id = next_message(message_id); |
| SAMPLE:#0028 | } |
| SAMPLE:#0029 | } |
| SAMPLE:#0030 | |
| SAMPLE:#0031 | init_data(void) /* Initialize data */ |
| SAMPLE:#0032 | { |
| SAMPLE:#0033 | data[0].status = ORIGINAL; |
| SAMPLE:#0034 | data[0].message = "This is a sample program"; |
| SAMPLE:#0035 | data[1].status = ORIGINAL; |
| SAMPLE:#0036 | data[1].message = "Upper and Lower Case Conversio |
| SAMPLE:#0037 | } |
| SAMPLE:#0038 | |
| SAMPLE:#0039 | convert(int id) /* Message Conversion */ |
+-----+-----+
Status--Running in monitor | File: ...OR/SAMPLES/SAMPLE.C
```

Step 12. Display a variable

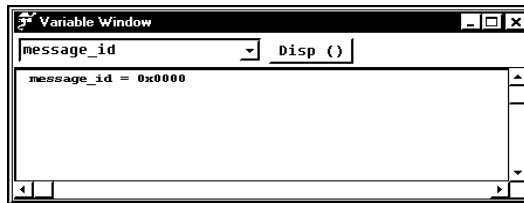
To display the contents of the "message_id" variable:

- 1 Double-click to highlight "message_id" 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".



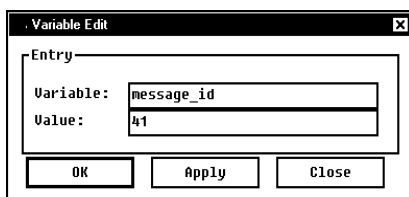
Step 13. Edit a variable

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.

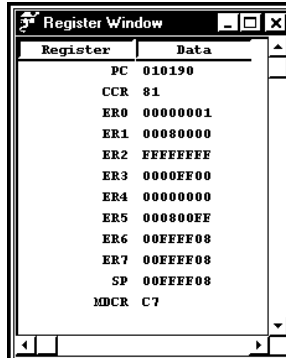
- 2 Enter "41" in the Value: text box.
- 3 Click the Apply button.



Step 14. 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 15. Exit the debugger

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

This will end your Emulation Solution User Interface session.

Note Never exit the debugger by sending “kill signal” (In Unix). Doing so may cause the logic analyzer’s network connection to be unstable. Always exit the interface software by choosing File→Exit.

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 Emulation Solution User Interface 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

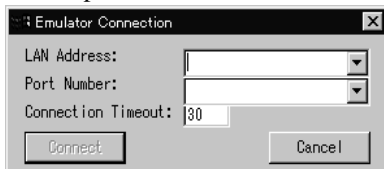
For PC:

- 1 Choose Agilent Technologies B3759A in the Emulation Solution Interface group from the Windows 95/NT Start menu.
- 2 The Main window opens which is used to invoke the Debug window, PC Trace window, and Bus Trace window.



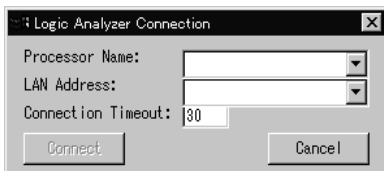
- 3 Choose the window to be connected from the Open menu of the Main window.

To open the Debug window or PC Trace window, choose Debug Window... or PC Trace Window... from the Open menu. Then the Connection dialog box appears. In the Connection dialog box, enter the information of the hardware you use (IP address/port of the Emulation Probe or Emulation Module).



- 4 When you complete, click the Connect button.
If connection is made successfully, the Debug window or the PC Trace window opens.
- 5 If you choose Bus Trace Window... from the Open menu of the Main window, the Connection dialog box for the Logic Analyzer appears.

In the Connection dialog box enter the name of the processor you use and the IP address/port information of the Logic Analyzer.



- 6 When you complete, click the Connect button.
If connection is made successfully, the Bus Trigger window and the Bus Trace window open.

For WS:

- 1 Enter the following command from the command line and press the Return key.

```
$ munin&
```

- 2 Unlike the PC version of the software, the main window in the WS version only equips with pull-down menus.
- 3 Choose the window to be connected from the Open menu of the Main window. You can choose from the three options; Debug Window..., PC Trace Window, or Bus Trace Window.

According to your choice, the Debug window, PC Trace window, or Bus Trace window opens.

Note

Note that some windows cannot open depending on your hardware environment.

- 4 When you choose Debug Window... or PC Trace Window, the Connection dialog box as shown below appears.

In the Connection dialog box, enter the IP address/port information of user's hardware (Emulation Probe/Emulation Module).

- 5 When you complete, click the Connect button.
If connection is made successfully, the Debug window or the PC Trace window opens.

- 6 If you choose Bus Trace Window... from the Open menu of the Main window, the Connection dialog box for the Logic Analyzer appears.

In the Connection dialog box enter the name of the processor you use and the IP address/port information of the Logic Analyzer

- 7 When you complete, click the Connect button.
If connection is made successfully, the Bus Trigger window and the Bus Trace window open.

To exit the Debugger

- From the Emulation Solution User I/F window, choose the File→Exit command.

This will end your Emulation Solution User Interface session.

Working with Emulation Solution User Interface Windows

This section shows you how:

- To use the Debug window.
- To use the PC Trace window.
- To use the Bus Trace window..
- To copy window contents to the file.
- To set tabstops in the Debug window.

To display windows from the Debug window

You can open the following windows from the Debug window:

- Source window
- Register window
- Memory window
- Variable window
- Peripheral window
(only when you are connected to the Emulation Probe/Emulation Module)
- I/O 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 once, or debug a program with source codes displayed for each module.

To display windows and dialog boxes from the PC Trace window

You can open the following windows and dialog boxes from the PC Trace window.

- PC Trace window
- File Selection dialog box
- Clock Speed Setting dialog box
- Trigger Condition dialog box
- Find String dialog box

To display windows and dialog boxes from the Bus Trace window.

You can open the following windows and dialog boxes from the PC Trace window.

- Bus Trigger window
- Bus Trace window
- Logic Analyzer Setting dialog box
- File Selection dialog box
- Object File Download dialog box
- Trigger Condition dialog box
- Memory Map dialog box

To copy window contents to a 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 ascii file. Selecting this command invokes the File Selection dialog box, which you use to specify the name of the file.

To set tabwidth in the Debug window

- 1 From the control menu, choose the Settings→Source View... (ALT, S S) command.
- 2 Enter the number of columns for the tab settings in the Tab Width: box.
- 3 Click the OK button.

Using the Entry Buffer

Some of the Emulation Solution User Interface's windows have an entry buffer, located at 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 a value or symbols in the entry buffer select it by double-clicking in an Emulation Solution User Interface's window. It is easier to double-click the symbols 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 changing the memory contents of a variable in a program, double-click a variable name in one of the Emulation Solution User Interface's windows, choose the Window→Memory... (ALT, W, N) command in the Debug window, specify a value, and press enter. In this case, you do not need to enter a variable name in the edit box in the Memory Window.

History Function

In the C 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 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 9, "Customizing the Emulation Solution 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 8.

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 Execute 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.

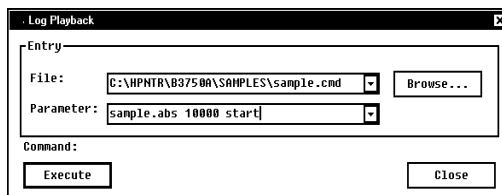
The Log Playback dialog box appears.

- 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 field, replacing the parameter field with %1, %2, %3, %4, or %5. Note that this replacement can be applied to the command field.

Example

Create a command file as shown below.

```
file binary %1
```

```
bp set %2
```

```
run %3
```

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 Emulation Solution User Interface loads the file sample.abs, sets a breakpoint at address 10000, and runs the program from the start address.

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.
- 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 Emulation Solution 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.

For this command, the entry buffer accepts symbol 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

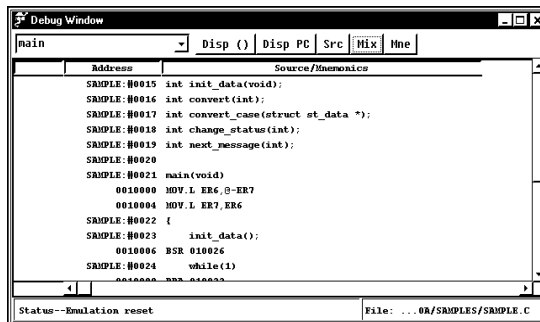
- From the Debug window's control menu, choose the Settings→Display Mode→Mixed (ALT, S, D, M) command.

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:



The screenshot shows a 'Debug Window' with a menu bar containing 'main', 'Disp ()', 'Disp PC', 'Src', 'Mix', and 'Mne'. The main area displays a table with two columns: 'Address' and 'Source/Mnemonics'. The content is as follows:

Address	Source/Mnemonics
SAMPLE:#0015	int init_data(void);
SAMPLE:#0016	int convert(int);
SAMPLE:#0017	int convert_case(struct st_data *);
SAMPLE:#0018	int change_status(int);
SAMPLE:#0019	int next_message(int);
SAMPLE:#0020	
SAMPLE:#0021	main(void)
0010000	MOV.L ER6,0-ER7
0010004	MOV.L ER7,ER6
SAMPLE:#0022	{
SAMPLE:#0023	init_data();
0010006	BSR 010026
SAMPLE:#0024	while(1)
0000000	... 010026

At the bottom of the window, the status bar shows 'Status--Emulation reset' and 'File: ...00/SAMPLES/SAMPLE.C'.

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 (for workstations)" in Chapter 8 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 specify the directories in which the source files can be found so that you may search them from the Emulation Solution User Interface. You can specify multiple directories at a time, so if the source files exist in different directories, you can have the Emulation Solution User Interface search these source files for different directories.

- 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 from the current program counter to a specifies 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, S, S) command from the Debug window's control menu.

For PC:

- Click the Continue action button in the Emulation Solution User I/F window.

Or:

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

This command executes the user program from the current program counter.

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 Start Address (ALT, E, R, S) 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 from the current program counter to a specifies 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 (.munin.ini for the WS version, or munin.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 returns

- 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 from the address where the program counter resides 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 Emulation Solution 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 Emulation Solution User I/F window.

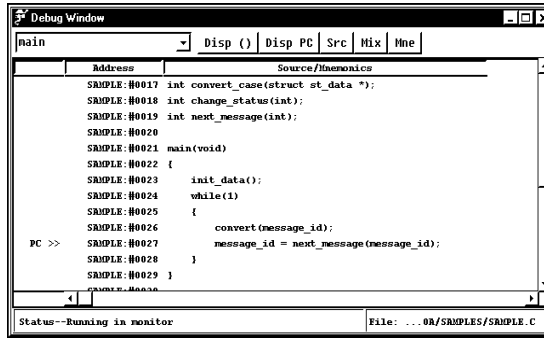
Or:

- 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. 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 Emulation Solution 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 Emulation Solution 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.
- 2 In the Debug window, press the right mouse button to display the pop-up menu.
- 3 Choose the Set Breakpoint command from the pop-up menu.

Or:

- 1 Select the symbol where you wish to set a breakpoint.
- 2 Choose the Execution→Breakpoint...(ALT, E, P) command.

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.

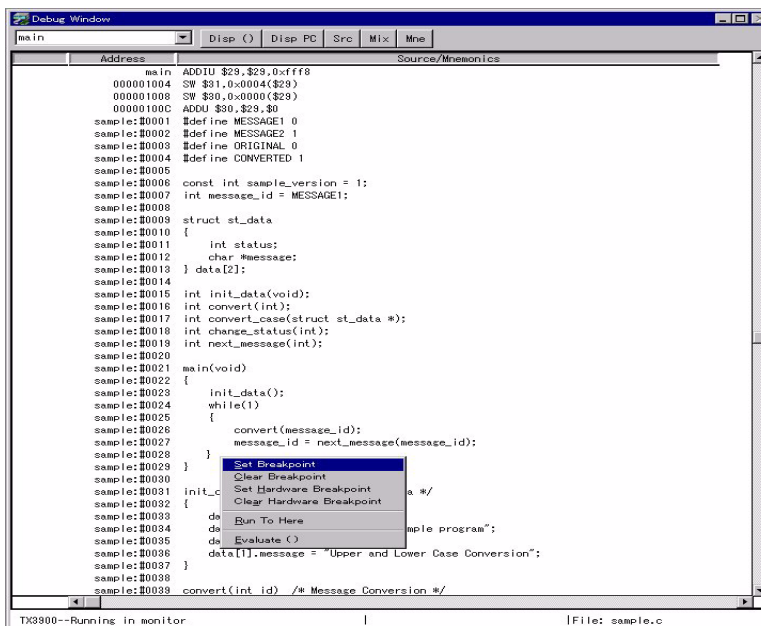
- 4 Click the Close button.

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

The location of the current program counter is highlighted.

Example

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



To list the breakpoints

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

The 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 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 to close the dialog box.

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" 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 delete 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 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 Choose the Evaluate () command from the pop-up menu.

Or:

- 1 Position the mouse pointer over the variable in the window and double-click the left mouse button.
- 2 From the Debug window's control menu, 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:

- Double-click the variable you want to display.

The selected variable name appears in the entry buffer in the Variable window.

3 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) from the Variable window's control menu. See "Settings→Auto Update (ALT, S, A)" in "Variable Window Commands" in Chapter 4 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 window and double-click the left mouse button.
- 2 Choose the Window→Variable (ALT, W, V) command from the Debug window's control menu.

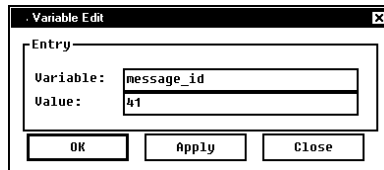
The Variable window appears. The selected variable and its value are displayed in the window.

- 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.

Note that the text box accepts only a value that matches the type of the variable.



- 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 In the Debug window, 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)

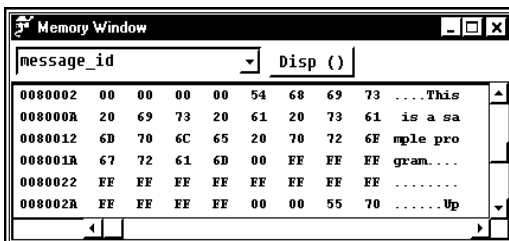
- Absolute Format

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). From the Memory window's control menu See "Settings→Auto Update (ALT, S, A)" in "Memory Window Commands" in Chapter 4, for details on updating options.

Example

Memory display in byte format:



To edit memory

- 1 Choose 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 starting from the selected address or 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.

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.

<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 4 for details.

For details on the classes that can be selected in the window, see *Emulation Solution User Interface User's Guide* specific to your processor.

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 From the Peripheral window, choose the Display command from the Peripheral window 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.

To apply the value you have entered, you need to press the Return key each time you enter the value.

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

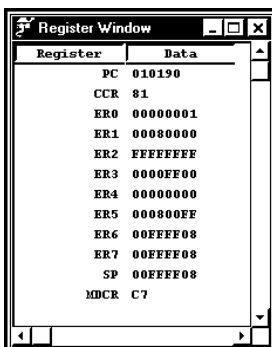
- Choose the Window→Register (ALT, W, R) command from the Debug window's control menu.

The Register window appears.

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 4, "Register Window Commands," for details.

Example

Register contents displayed in the Register window:



To edit registers

- 1 From the Debug window control menu, choose Window → Register (ALT, W, R) command to display the register value.
- 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.

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 C debugger 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 C Debugger 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

**Debug Window Commands and Window
Control Menu Commands**

Debug Window Commands and Window Control Menu Commands

This chapter describes the Debug window commands and the windows which can be selected for display from the Debug window control menu.

- Emulation Solution I/F Main window commands.
- Debug window commands.
- Source window commands.
- Register window commands.
- Memory window commands.
- Variable window commands.
- Peripheral window commands.
- I/O window commands
- Backtrace window commands.
- Debug Window Pop-up Commands
- Source Window Pop-up Commands
- Backtrace Window Pop-up Commands

Emulation Solution User I/F Main Window Commands

The main window of the Emulation Solution User interface includes the commands below.

- File -> Exit (ALT, F, X)
- Open -> Debug Window... (ALT, O, D)
- Open -> PC Trace Window... (ALT, O, P)
- Open -> Bus Trace Window... (ALT, O, B)
- Settings -> Font (ALT, S, F) *
- Window -> Tile (ALT, W, T) *
- Window -> Cascade (ALT, W, C) *
- Window -> Arrange Icons (ALT, W, A) *
- Help -> Contents (ALT, H, C)
- Help -> How to Use Help (ALT, H, H)
- Help -> Search for help on... (ALT, H, S)
- Help -> Version (ALT, H, V)

* indicates that these commands are only available for PC version only.

File -> Exit (ALT, F, X)

Exit (Quit) the Emulation Solution User I/F.

Command File Command

No command

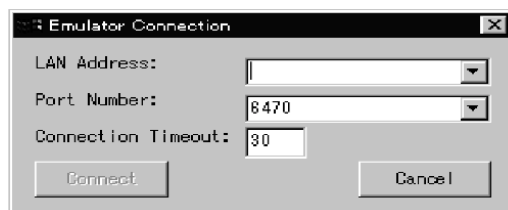
See Also

“Starting and Exiting the Debugger” in chapter 2, “Using the Debugger Interface”

Open -> Debug Window (ALT, O, D)

Opens a Emulator Connection dialog box. You may connect to your emulation probe/module using this dialog box. A debug window appears upon providing the IP address and the port number in the dialog box.

Emulator Connection dialog box



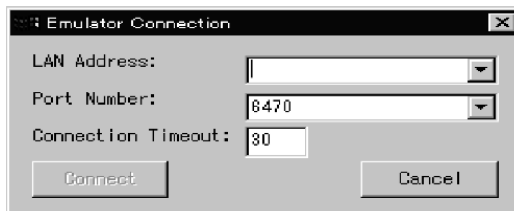
LAN Address	Specify the IP address of the emulation probe/module.
Port Number	Specify the port number of the emulation probe/module.
Connection Timeout	Specify the time (in seconds) for the interface software to wait until the connection is established. If the connection were not made within the time specified, it outputs an error message.
Connect	Attempt to connect to an emulation probe/module.
Cancel	Closes the dialog box without making a connection.

Open->PC Trace Window... (ALT, O, P)

If you have already made a connection with your emulation probe/module using Open->Debug Window command, Open->PC Trace Window brings up a PC Trace window. If the connection has not been established, Emulator Connection dialog box appears which you must provide a correct IP address for the emulation probe/module.

Open->Bus Trace Window... (ALT, O, B)

Opens the Logic 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.



Processor Name	Specify the name of the target processor.
LAN Address	Specify the IP address/hostname for the logic analyzer.
Connection Timeout	Specifies, in seconds, the timeout period for connecting to the analyzer. If the connection is not made within the specified time, it displays a connection error message and aborts the connection.
Connect	Attempts to establish the connection.
Cancel	Closes the dialog box without making a connection.

Settings->font (ALT, S, F) (For PC only)

Changes the font settings of windows. You may change font, font style and font size.

Window->Tile (ALT, W, T) (For PC only)

Arranges and sizes windows so that none are overlapped.

Windows are sized evenly.

Command File Command

No command

Window->Cascade (ALT, W, C) (For PC only)

Arranges, sizes, and overlaps windows.

Windows are sized evenly as large as possible.

Command File Command

No command

Window-> Arrange Icons (ALT, W, A) (For PC only)

Rearranges icons in the Debug User I/F window.

Icons are distributed evenly along the lower edge of the Emulation Solution 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 Emulation Solution 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 Emulation Solution User Interface. The Emulation Solution User Interface install directory and the start-up directory are also displayed.

Command File Command

No command

Debug Window Commands

The Debug window is used to control execution of the user application and display the subwindows.

This chapter describes the following commands:

- File→Load→Configuration...(ALT,F,L,C)
- File→Load→Program...(ALT,F,L,P)
- File→Load→Environment...(ALT,F,L,E)
- File→Store→Configuration...(ALT,F,S,C)
- File→Store→Environment...(ALT,F,S,E)
- File→Copy→Display...(ALT,F,P,D)
- File→Log→Playback...(ALT,F,O,P)
- File→Log→Recode...(ALT,F,O,R)
- File→Log→Stop(ALT,F,O,S)
- Execution→Run→FromPC(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,N)
- Window→Variable(ALT,W,V)
- Window→Peripheral(ALT,W,P)
- Window→I/O(ALT,W,I)
- Window→Backtrace(ALT,W,B)

Debug Window Commands

- 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→Access Size..(ALT,S,C,A)
- Settings→Data Read Form→Memory(ALT,S,R,M)
- Settings→Data Read Form→Object File(ALT,S,R,O)

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

Loads an emulation probe configuration file.

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

Emulation probe configuration files contain:

- Hardware configuration settings.
- Memory map configuration settings.

Command File Command

No command.

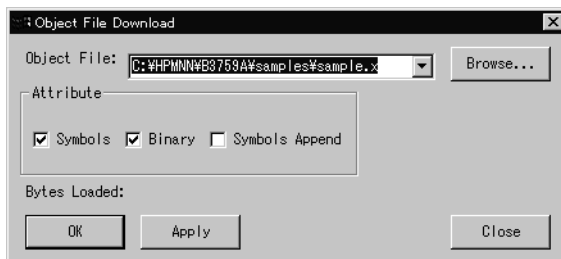
See Also

"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, see *Emulation Solution User Interface User's Guide* specific to your system.

Object File Download 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, or when code resides in the target system's ROM).
Binary	Loads program code but not 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.

`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"

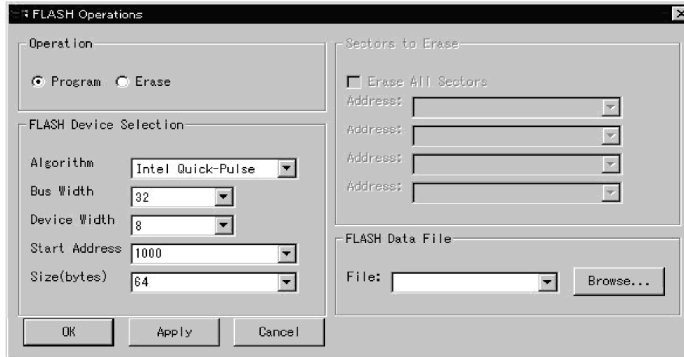
File→Load→Environment... (ALT, F, L, E)

Loads the information of the debugging environment.

File->Load->Flash Operation... (ALT, F, L, F)

This command let you download or erase the target flash memory.

Flash Operations Dialog Box



Operation

- Program** Choose this option when downloading code to the target flash memory.
- Erase** Choose this option when erasing the data from the target flash memory.

Flash Device Selection

- Algorithm** Choose the supported flash algorithm from below.

AMD 5V Embedded

Let you execute the sector erase. You may specify the sector address to erase or erase all the sectors.

AMD 12V Embedded

You may only execute to erase all the sectors.

Intel Auto

You may only execute to erase specified sector address.

Intel Quick-Pulse

You may only execute to erase all the sectors.

Bus Width	Specify the bus width (8, 16 or 32bit)
Device width	Specify the device width (8 or 16bit)

Sectors to Erase

Erase All Sectors	Erases all the flash sectors.
Address	Specify the sector address to be erased.

Flash Data File

File	Specify the file name for the data which is downloaded into the flash memory.
OK	Start downloading the specified data file into the flash memory. The dialog box closes.
Apply	Start downloading the specified data file into the flash memory. The dialog box remains open, so you may continuously download more data files.
Cancel	Cancel the flash operation.

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

Saves the current hardware configuration to a 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.

Command File Command

No command.

See Also

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

File→Store→Environment... (ALT, F, S, E)

Saves the information of the debugging environment.

File→Copy→Display... (ALT, F, P, D)

Outputs the current contents of the 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.

The output file is created in ASCII format, so that you can view the file contents with an appropriate text editor.

Command File Command

No command.

See Also

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

File→Log→Playback... (ALT, F, O, P)

Executes the specified command file.

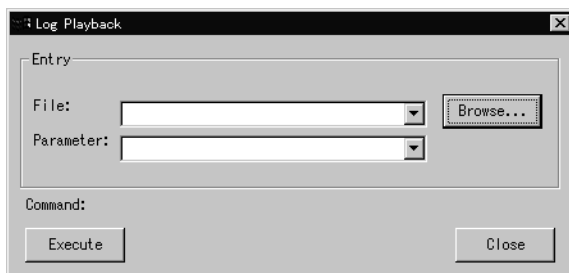
You can specify the following command files:

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. Note that this replacement can be applied to the command field.

Example

Create a command file as shown below.

```
file binary %1  
  
bp set %2  
  
run %3
```

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 Emulation Solution User 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 Emulation Solution 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 Emulation Solution User 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 emulation probe/module 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 (`.munin.ini` for the WS version, or `munin.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, 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 from the specified 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 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 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.

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 target 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 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.

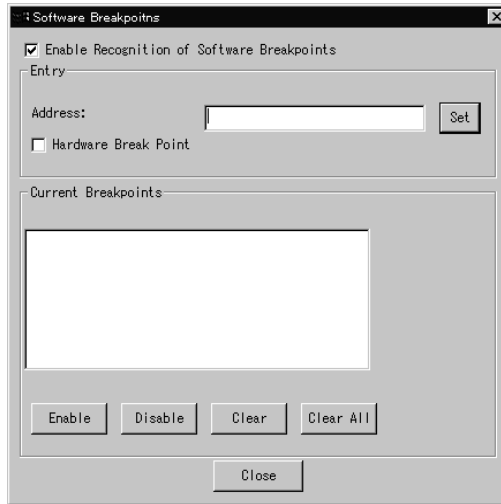
Breakpoints cannot be set in programs stored in target system ROM.

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.

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.
Set	Sets a breakpoint.
Hardware BreakPoint	Check the box if you want to set a hardware break point on the specified address.
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 software breakpoint.

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

Enables/Disables the specified software breakpoint.

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

Enables/Disables all software breakpoints.

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

Deletes the specified software breakpoint.

```
bp cle (ar) all
```

Enables a hardware breakpoint.

```
bp set address hard
```

Disables a hardware breakpoint.

```
bp clear address hard
```

Deletes all software 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.

Specify the address to be displayed in the entry buffer on the tool bar, then choose the above command.

You can use symbols to specify addresses.

Command File Command

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

See Also

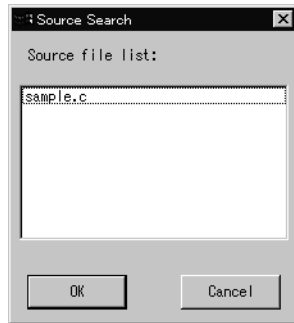
"To display source code specifying a heading line" under "Loading and Displaying Programs" in Chapter 3, "Debugging Programs"

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. |
| Source file list | 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"

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)
```

Note

The Peripheral window is displayed only when the Emulation Probe or the Emulation Module is used in the debugging environment.

See Also

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

Window→I/O (ALT, W, I)

Opens the Peripheral window.

Command File Command

```
dis(play) I(O)
```

Display→Find... (ALT, D, F)

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

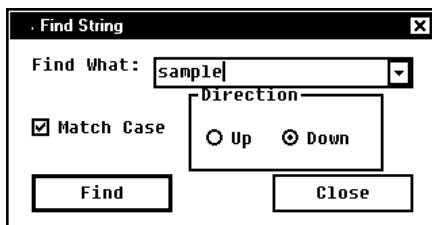
The search starts from the current cursor position in the Debug window. You can set the search direction using the Direction option button in the Find String dialog box. Checking the Match Case check box enables a case-sensitive search.

Before searching for strings using this command, you must set the Source window 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.

Window→Backtrace (ALT, W, B)

Opens the Backtrace window.

Command File Command

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

See Also

"Backtrace Window Commands" in Chapter 4, "Debug Window Commands and Window Control Menu Commands"

"The Backtrace Window" under "Debugger Windows" 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 Emulation Solution User Interface allows you to change the color of highlighted source lines (workstation version only). See "Customizing the Debugger Interface's Appearance (for workstations)" in Chapter 9, "Customizing the Emulation Solution User 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"

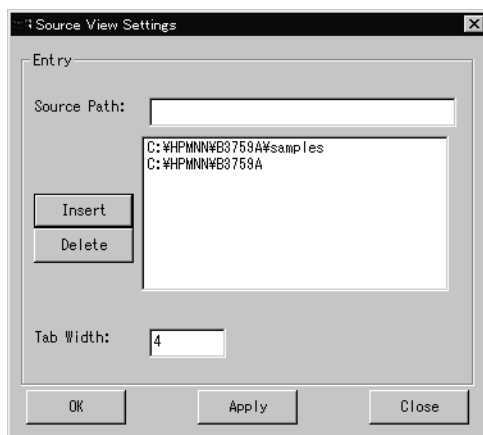
Settings→Source View... (ALT, S, S)

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 *Emulation Solution User Interface User's Guide* specific to your processor.

Command File Commands

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

See *Emulation Solution User Interface User's Guide* specific to your processor.

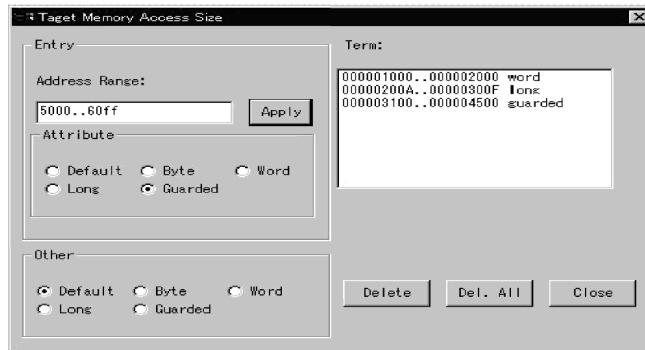
Settings→Configuration→Access Size... (ALT, S, C, M)

Displays a Target Memory Access Size dialog window.

You may specify the following settings in the Target Memory Access Size window.

Specify the address range and the access size.

Specify the default access size.



Address Range: Specify the address range.

Apply: Stores the address range data you entered.

Attribute: Specify the access size for the address range.

Other: Specify the access size for the address range other than what are entered in above.

Term: Displays the current settings in the area.

Delete:	Deletes a specified settings in the Term area.
Delete.All:	Deletes all the access size settings.
Close:	Closes the dialog box.

Settings→Data Read From→Memory... (ALT, S, R, M)

Read the information needed for the interface software to inverse assemble the user program from the target memory.

Settings->Data Read From->Object File...(ALT, S, R, O)

Read the information needed for the interface software to inverse assemble the user program from the object file resides in the interface software.

Source Window Commands

The Source window is used to view source files.

This command allows you to display the source file comprising the object file and set breakpoints.

The data to be deassembled is read from the object file, not from the memory.

This section describes the following commands:

- File→Copy→Display(ALT,F,C,D)
- File→Open (ALT,F,O)
- File→Close(ALT,F,C)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)
- Settings→Data Read From→Memory(ALT,S,R,M)
- Settings→Data Read From→Object File(ALT,S,R,O)

File→Copy→Display(ALT,D,A)

Outputs the contents of the Source window to a file.

When you choose the above command, the file selection dialog box appears from which you can specify the name of the output file.

File→Open(ALT,F,O)

Opens a new Source window.

File→Close(ALT,F,C)

Closes the current Source window.

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"

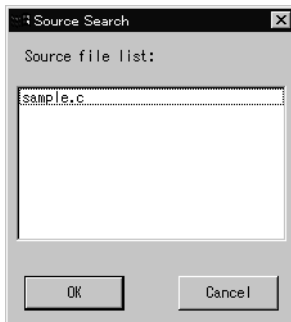
Display→Source Files... (ALT, D, S)

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.

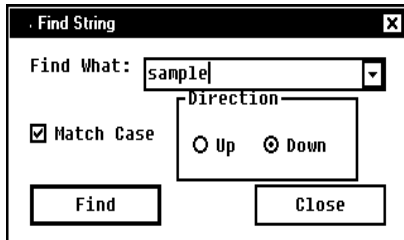
The search starts from the current cursor position in the Source window. You can set the search direction using the Direction option button in the Find String dialog box. Checking the Match Case check box enables a case-sensitive search.

Before searching for strings using this command, you must set the Source window 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 Emulation Solution User Interface allows you to change the color of highlighted source lines (WS version only). See "Customizing the Debugger Interface's Appearance (for workstations)" in Chapter 9 "Customizing the Emulation Solution User Interface" for details.

Command File Command

No command.

See Also

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

Settings→Data Read From→Memory(ALT,S,R,M)

Reads the data required for deassembling from the memory.

Settings→Data Read From→Object File(ALT,S,R,O)

Reads the data required for deassembling from the object file.

Register Window Commands

The Register window is used to display names and values of the basic registers.

This section describes the following command:

- File→Copy→Display
- File→Open
- File→Close
- Settings→Display Base→Binary(ALT,S,D,B)
- Settings→Display Base→Octal(ALT,S,D,O)
- Settings→Display Base→Decimal(ALT,S,D,D)
- Settings→Display Base→Hexadecimal(ALT,S,D,H)
- Settings→Auto Update(ALT,S,A)
- Settings→Auto Update (ALT, S, A)

File→Copy→Display(ALT,D,A)

Outputs the contents of the register window to a file.

When you choose the above command, the file selection dialog box appears from which you can specify the name of the output file.

File→Open(ALT,F,O)

Opens a new Register window.

File→Close(ALT,F,C)

Closes the current Register window.

Settings→Auto Update (ALT, S, A)

Automatically updates contents displayed in the Register window.

You can set the update interval in the initialization file (.munin.ini for the WS version, or munin.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"

Settings→Display Base→Binary(ALT,S,D,B)

Displays register values in binary format.

Settings→Display Base→Octal(ALT,S,D,O)

Displays register values in octal format.

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

Displays register values in decimal format.

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

Displays register values in hexadecimal format.

Memory Window Commands

The Memory window is used to display and change the memory contents.

This section describes the following commands:

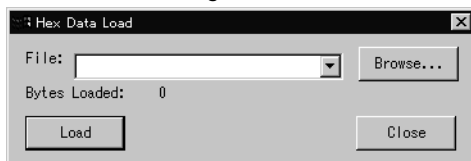
- File→Load(ALT,F,L)
- File→Store(ALT,F,S)
- File→Copy→Display
- File→Open
- File→Close
- Display→From () (ALT, D, F)
- Display→Blocked→Byte (ALT, D, B, B)
- Display→Blocked→Word (ALT, D, B, W)
- Display→Blocked→Long (ALT, D, B, L)
- Display→Absolute→Byte (ALT, D, A, B)
- Display→Absolute→Word (ALT, 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)

File→Load(ALT,F,L)

Loads the data into the memory.

The Hex Data Load dialog box appears.

Hex Data Load dialog box



- | | |
|------------|--|
| File | Lets you specify the file to be downloaded. |
| Browse... | Opens a file selection dialog box from which you can select the file to be downloaded. |
| ByteLoaded | Displays the amount of the data being downloaded into the memory. |
| Load | Downloads the file you have specified. |
| Close | Closes the dialog box. |

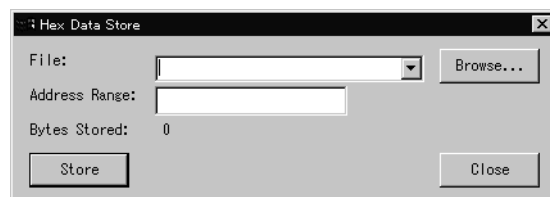
Command File Command

mem(ory) loa(d) file_name

File→Store(ALT,F,S)

Saves the loaded data from the memory into a file. The Hex Data dialog box appears.

Hex Data dialog box



File	Lets you specify the file to be uploaded.
Browse...	Opens a file selection dialog box from which you can select the file to be uploaded.
AddressRange	Lets you specify the address range for the data to be stored.
ByteStores	Displays the amount of the data being uploaded in ??kilobytes??.
Store	Uploads the file you have specified.
Close	Closes the dialog box.

File→Copy→Display(ALT,D,A)

When you choose the above command, the file selection dialog box appears from which you can specify the name of the output file

File→Open(ALT,F,O)

Opens a new memory window.

File→Close(ALT,F,C)

Closes the current memory window.

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

Display→Blocked→Byte (ALT, D, B, B)

Displays memory contents in 8-bit block format.

Command File Command

No command.

Display→Blocked→Word (ALT, D, B, W)

Displays memory contents in 16-bit block format.

Command File Command

No command.

Display→Blocked→Long (ALT, D, B, L)

Displays memory contents in 32-bit block format.

Command File Command

No command.

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

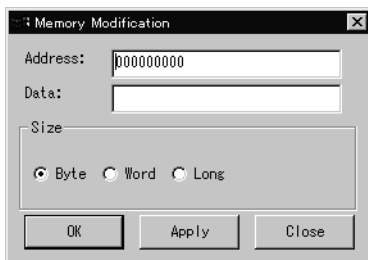
"To display memory" under "Displaying and Editing Memory" in Chapter 3, "Debugging Programs"

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(1) range byt(e) value`

Specifies the data in 8-bit format.

`mem(ory) fil(1) range wor(d) value`

Specifies the data in 16-bit format.

`mem(ory) fil(1) 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 fill 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 (.munin.ini for the WS version, or munin.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 target processor.
- 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

The Variable window is used to display the contents of a specified variable.

One window is used for one variable and you can open multiple instances of the variable window.

This section describes the following commands:

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

File→Copy→Display(ALT,D,A)

Outputs the contents of the Variable window to a file.

When you choose the above command, the file selection dialog box appears from which you can specify the name of the output file

File→Open(ALT,F,O)

Opens a new Variable window.

File→Close(ALT,F,C)

Closes the current Variable window.

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.

Variable Window Commands

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

"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

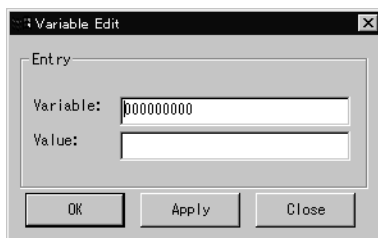
"To display a variable" under "Displaying and Editing Variables" in Chapter 3, "Debugging Programs"

Modify→Variable... (ALT, M, V)

Modifies the contents of a specified variable.

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"

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 (.munin.ini for the WS version, or munin.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

- File->Copy->Display (ALT, P, D)
- File->Open (ALT, F, O)
- File->Close (ALT, F, C)
- Displays-><register class name>
- Settings->Auto Update (ALT, S, A)
- Settings->Display Base->Binary(ALT,S,D,B)
- Settings->Display Base->Octal(ALT,S,D,O)
- Settings->Display Base->Decimal(ALT,S,D,D)
- Settings->Display Base->Hexadecimal(ALT,S,D,H)

Above commands may be executed in the Peripheral window. Please note that this window menu changes depend on your target processor. Some of the functions may not present for your processor. For detailed information for this window, please refer to the Emulation Solution User Interface user's guide specific to your processor.

I/O Window Commands

I/O window is used to display I/O values of a specified address.

This section describes the following commands:

- Files→Copys→Display(ALT,P,D)
- Files→Open(ALT,F,O)
- Files→Close(ALT,F,C)
- Settings→Set(ALT,S,S)
- Settings→Display Base→Binary(ALT,S,D,B)
- Settings→Display Base→Octal(ALT,S,D,O)
- Settings→Display Base→Decimal(ALT,S,D,D)
- Settings→Display Base→Hexadecimal(ALT,S,D,H)
- Settings→Auto Update (ALT, S, A)

File→Copy→Display(ALT,D,A)

Outputs the contents of the I/O window to a file.

When you choose the above command, the file selection dialog box appears from which you can specify the name of the output file

File→Open(ALT,F,O)

Opens a new I/O window.

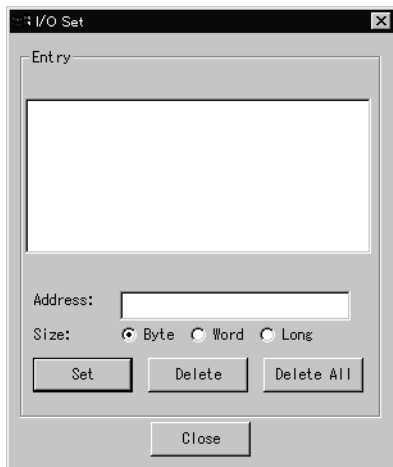
File→Close(ALT,F,C)

Closes the current I/O window.

Settings→Set(ALT, S, S)

Specifies the I/O address and size.

Choosing the Setting → Set (ALT, S, S) command opens the following dialog box.



- | | |
|------------|---|
| Entry | Displays the current I/O. |
| Address | Lets you specify the I/O address. |
| Size | Lets you specify the I/O size. |
| Set | Lets you register the specified address and size. |
| Delete | Deletes the selected I/O entries from the entry buffer. |
| Delete All | Deletes all the I/O entries from the entry buffer. |
| Close | Closes the dialog box. |

Command File Commands

```
io del(ete) all
io set address size
io address size value
```

Settings→Display Base→Binary (ALT, S, D, B)

Displays I/O values in binary format.

Settings→Display Base→Octal (ALT, S, D, O)

Displays I/O values in octal format.

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

Displays I/O values in decimal format.

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

Displays I/O values in hexadecimal format.

Settings→Auto Update(ALT,S,A)

Automatically updates displayed contents in the Memory window.

You can set the update interval in the initialization file (.munin.ini for the WS version, or munin.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 Emulation Probe/Emulation Module.
- Changing the memory mapping.
- Loading a program file.

Backtrace Window Commands

The Backtrace window is used to analyze the information in the stack based upon the values of the program counter, frame pointer, and the stack pointer.

This section describes the following commands:

- Files→Copys→Display(ALT,P,D)
- Files→Open(ALT,F,O)
- Files→Close(ALT,F,C)
- Source at Stack Level

File→Copy→Display(ALT,D,A)

Outputs the contents of the Backtrace window to a file.

When you choose the above command, the file selection dialog box appears from which you can specify the name of the output file

File→Open(ALT,F,O)

Opens a new Backtrace window.

File→Close(ALT,F,C)

Closes the current Backtrace window.

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
```

Debug Window Pop-up Commands

This section describes the commands that can be selected from the pop-up menus in the Debug windows. Pop-up menus are accessed by clicking the right mouse button in the window.

- Set Software Breakpoint
- Clear Software Breakpoint
- Set Hardware Breakpoint
- Set Hardware Breakpoint
- Run to Here
- Evaluate ()

Set Software Breakpoint

Sets a breakpoint on the line containing the cursor.

See the Execution→Breakpoints... (ALT, E, P) command description.

Clear Software Breakpoint

Deletes the breakpoint on the line containing the cursor.

See the Execution→Breakpoints... (ALT, E, P) command description.

Set Hardware Breakpoint

Sets a hardware breakpoint on the line containing the cursor.

See the Execution→Breakpoints... (ALT, E, P) command description.

Clear Hardware Breakpoint

Deletes the hardware 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 ()

Displays the value of the selected variable.

See the Window → Variable (ALT, W, V) command description.

Source Window Pop-up Commands

This section describes the commands that can be selected from the pop-up menus in the Source windows. Pop-up menus are accessed by clicking the right mouse button in the window.

- Set Breakpoint
- Clear Breakpoint
- Set Hardware Breakpoint
- Set Hardware 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.

Set Hardware Breakpoint

Sets a hardware breakpoint on the line containing the cursor.

See the Execution→Breakpoints... (ALT, E, P) command description.

Clear Hardware Breakpoint

Deletes the hardware breakpoint on the line containing the cursor.

See the Execution→Breakpoints... (ALT, E, P) command description.

Evaluate ()

Displays the value of the selected variable.

See the Window → Variable (ALT, W, V) command description.

Backtrace Window Pop-up Commands

This section describes the commands that can be selected from the pop-up menus in the Backtrace windows. Pop-up menus are accessed by clicking the right mouse button in the window.

- 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.

PC Trace Window Commands

PC Trace Window Commands

The PC Trace window provides the user interface to the N-Trace functions which can be activated when the Emulation Probe or the Emulation Module is connected.

This section describes the following commands:

- File→Copy→Display...(ALT,F,P,D)
- File→Copy→All...(ALT,F,P,A)
- File→Close(ALT,F,C)
- Trace→Always(ALT,T,L)
- Trace→After()(ALT,T,F)
- Trace→About()(ALT,T,A)
- Trace→Before()(ALT,T,B)
- Trace→Until()(ALT,T,U)
- Trace→Until Halt(ALT,T,H)
- Trace→Condition...(ALT,T,C)
- Trace→Again(ALT,T,G)
- Trace→Halt(ALT,T,T)
- Display→Trigger{ ALT,D,T}
- Display→Find...(ALT,D,F)
- Settings→Trace Mode?Real-Time(ALT,S,M,R)
- Settings→Trace Mode?NonReal-Time(ALT,S,M,N)
- Settings→Clock Speed...(ALT,S,S)
- Settings→Display Mode→SourceOnly(ALT,S,D,S)
- Settings→Display Mode→Mixed(ALT,S,D,M)
- Settings→Display Mode→Mnemonics(ALT,S,D,N)
- Settings→Display Mode→Symbols(ALT,S,D,Y)
- Settings→Display Mode→FunctionNames(ALT,S,D,F)

File→Copy→Display...(ALT,F,P,D)

Copies the current contents of the PC Trace 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.

File→Close...(ALT,F,C)

Closes the PC Trace window.

Command File Command

No command.

Trace→Always (ALT, T, L)

Specifies no trigger conditions. Trace starts immediately after this command is chosen.

Performs all trace operation continuously.

Command File Command

```
ntr(ace) alw(ays)
```

See Also

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

Trace→After () (ALT, T, F)

Specifies a trace position so that the address specified in the entry buffer will be traced at the top of the trace list. The operation to that address may be instruction or execution.

If the trace position is met, the trace data starts to be read.

Refer to "Address Format" for various formats to specify an address.

Command File Command

```
ntr(ace) 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)

Specifies a trace position so that the address specified in the entry buffer will be traced at the center of the trace list. The operation to that address may be instruction execution, fetch, read, or write. Refer to "Address Format" for various formats to specify an address.

Command File Command

```
ntr(ace) 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)

Specifies a trace position so that the address specified in the entry buffer will be traced at the end of the trace list. The operation to that address may be instruction execution, fetch, read, or write.

The trace data is always read. The read operation stops when the trace position condition is met.

Refer to "Address Format" for various formats to specify an address.

Command File Command

```
ntr(ace) bef(ore) address
```

See Also

"To end tracing at 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

```
ntr(ace) 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

```
ntr(ace) unt(il) hal(t)
```

See Also

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

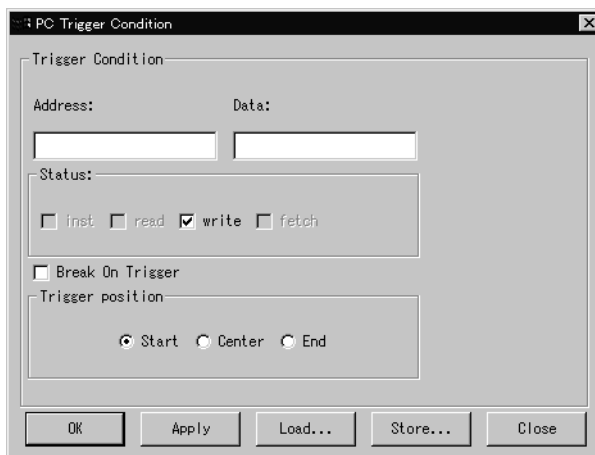
Trace→Condition... (ALT, T, C)

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:



Address	Specifies the address of the trigger condition. If a symbol corresponds to the address, the symbol name is displayed. Refer to "Address Format" for various formats to specify an address.
Data	Specify the data from which trigger starts. Refer to "Data Format" for various formats to specify data.
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.
Break On Trigger	Specifies if triggering causes a break into the monitor. This enables the trace condition to be used as hardware break point.
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	Set a specified trigger. 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 ".ntg" 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.

Command File Command

See Chapter 6, "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→Again (ALT, T, G)

Traces program execution using the last trace specification.

Command File Command

```
ntr(ace)
```

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

```
ntr(ace) 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 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)

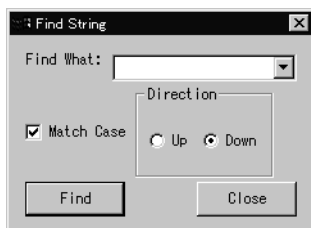
Searches for and displays a string in the Trace window.

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 value and string can be selected from another window (that is, copied to the entry buffer) before choosing the command; they 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"

Trace→Trace Mode→Realtime

Executes the trace measurement in realtime mode which forces the user program to run in realtime.

Trace→Trace Mode→Non-Realtime

Executes the trace measurement in non-realtime mode which forces the user program to run in non-realtime.

Settings→Clock Speed (ALT, S, S)

Enter the clock speed of the target system. At this time, you also need to enter the unit. Available units are MHz, kHz, ns, and us.

Note

Your Emulation Solution User I/F software may or may not equip the above commands depends on your target processor.

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

Selects the source-only display mode.

Command File Command

```
ntr(ace) 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

```
ntr(ace) 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→Mnemonic Only (ALT, S, D, N)

Selects the mnemonic-only display mode.

Command File Command

```
ntr(ace) 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

```
ntr(ace) set sym(bols) on
```

```
ntr(ace) 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

```
ntr(ace) 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

```
ntr(ace) 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 Emulation Solution User Interface allows you to change the color of highlighted source lines (WS version only). See "Customizing the Debugger Interface's Appearance (for workstations)" in Chapter 9, "Customizing the Emulation Solution User Interface" for details.

Command File Command

```
ntr(ace) set hig(hlight) on  
ntr(ace) 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)

Counts in the relative mode. This command shows the intervals between states.

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)

Counts in the cumulative mode. This command shows the count from the trace start.

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→Data Read From→Memory (ALT, S, R, M)

Reads the data required for deassembling from the memory.

Settings→Data Read From→Object File(ALT, S, R, O)

Reads the data required for deassembling from the object file.

Bus Trace Interface

Bus Trace Interface

Bus Trace interface consists Bus Trigger window and Bus Trace window. This chapter describes the functions in each window.

Bus Trigger Window Commands

The Bus Trigger window provides the user interface for various logic analyzer settings. It also provides a C-source window for easy-to-use trigger settings.

This section describes the following commands:

- File→Load→Program...(ALT,F,L,P)
- File→Copy→Display...(ALT, F, P, D)
- Trigger→Always (ALT, T, L)
- Trigger→After () (ALT, T, F)
- Trigger→About () (ALT, T, A)
- Trigger→Before () (ALT, T, B)
- Trigger→Until Halt (ALT, T, U)
- Trigger→Condition... (ALT, T, C)
- Trigger→Again (ALT, T, G)
- Trigger→Halt (ALT, T, H)
- Display→Program At() (ALT, D, A)
- Display→Source Files (ALT, D, S)
- Display→Find... (ALT, D, F)
- Setting→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)

File→Load→Program...(ALT,F,L,P)

Loads the specified object file and symbolic information into the window. For available object files, see Emulation Solution User Interface User's Guide specific to your system.

Object File Download Dialog Box



- Object File** Let 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 the symbolic information.
- Symbols Append** Use this option when you want to append symbolic information to what is already loaded into the interface.
- 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.

Note This command does not load a object code into your target system.

File→Copy→Display...(ALT, F, P, D)

Copies the current contents of the PC Trace 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.

Trigger→Always (ALT, T, L)

Specifies no trigger conditions. Trace starts immediately after this command is chosen.

Performs all trace operation continuously.

Trigger→After () (ALT, T, F)

Specifies a trace position so that the address specified in the entry buffer will be traced at the top of the trace list. The operation to that address may be instruction execution, fetch, read, or write.

If the trace position is met, the trace data starts to be read.

Refer to "Address Format" for various formats to specify an address.

Trigger→About () (ALT, T, A)

Specifies a trace position so that the address specified in the entry buffer will be traced at the center of the trace list. The operation to that address may be instruction execution, fetch, read, or write. Refer to "Address Format" for various formats to specify an address.

Trigger→Before () (ALT, T, B)

Specifies a trace position so that the address specified in the entry buffer will be traced at the end of the trace list. The operation to that address may be instruction execution, fetch, read, or write.

The trace data is always read. The read operation stops when the trace position condition is met.

Refer to "Address Format" for various formats to specify an address.

Trigger→Until Halt (ALT, T, U)

Traces program execution until the Trigger→Halt (ALT, T, H) 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 Trigger→Until Halt (ALT, T, U) command. Then run the program. After the processor has halted or broken to the monitor, choose the Trace→Halt (ALT, T, H) command to stop the tracing. The execution that led up to the break or halt will be displayed.

Trigger→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. This command varies depends on your system. For detailed description, refer to Emulation Solution User Interface User's Guide specific to your system.

Trigger→Again (ALT, T, G)

Traces program execution using the last trace specification.

Trigger→Halt (ALT, T, H)

Stops a running trace.

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

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

Display→Program At() (ALT, D, A)

Displays the source code starting from the current program counter.

Display→Source Files (ALT, D, S)

Displays the specified source file in the Bus Trigger 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. |
| Source file list | Closes the dialog box. |

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

Display→Find... (ALT, D, F)

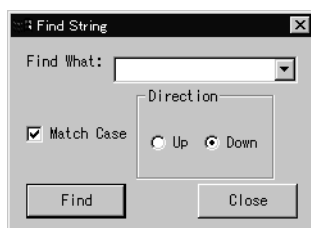
Searches for and displays a string in the Bus Trigger window.

The search starts from the current cursor position in the Bus Trigger 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 value and string can be selected from another window (that is, copied to the entry buffer) before choosing the command; they 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.

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

Selects the source-only display mode.

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

Selects the source/mnemonic mixed display mode.

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

Selects the mnemonic-only display mode.

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

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

Displays symbols.

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

Highlights the source lines.

The Emulation Solution User Interface allows you to change the color of highlighted source lines (workstation version only). See "Customizing the Debugger Interface's Appearance (for workstations)" in Chapter 9, "Customizing the Emulation Solution User Interface" for details.

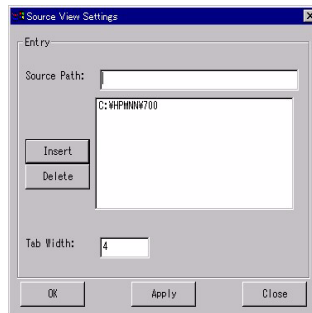
Settings→Source View

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 Bus Trigger window 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 | Let 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 | Let you specify the number of spaces between tabstops from 1 to 32 |
| OK | Saves the settings and closes the dialog box. |

Bus Trace Window

The Bus Trace window provides the user interface to show the external bus data from the logic analyzer with C source code. The window displays the trace result depend on the trigger settings in Bus Trigger window.

This section describes the following commands:

- File→Copy→Display...(ALT,F,P,D)
- Display→Trigger{ALT,D,T}
- Display→Find String(ALT,D,F)
- Display→Find G1(ALT,D,I)
- Display→Find G2(ALT,D,N)
- Settings→Display Mode→SourceOnly(ALT,S,D,S)
- Settings→Display Mode→Mixed(ALT,S,D,M)
- Settings→Display Mode→Mnemonics(ALT,S,D,N)
- Settings→Display Mode→Symbols(ALT,S,D,Y)
- 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)

File→Copy→Display...(ALT, F, P, D)

Copies the current contents of the PC Trace 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.

Display→Trigger (ALT, D, T)

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

Display→Find String (ALT, D, F)

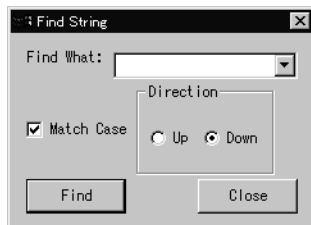
Searches for and displays a string in the Trace window.

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 value and string can be selected from another window (that is, copied to the entry buffer) before choosing the command; they 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.

Display→Find G1 (ALT, D, I)

Displays the position of the G1 (Global Pointer 1) on the logic analyzer. The marker is correlated with the Agilent Technologies 16600/700 series logic analyzer in a same time frame.

This feature is useful when you are performing measurements with other domain. (For example, using a oscilloscope module)

Display→Find G2 (ALT, D, N)

Displays the position of the G2 (Global Pointer 2) on the logic analyzer. The marker is correlated with the Agilent Technologies 16600/700 series logic analyzer in a same time frame.

This feature is useful when you are performing measurements with other domain. (For example, using a oscilloscope module)

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

Selects the source-only display mode.

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

Selects the source/mnemonic mixed display mode.

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

Selects the mnemonic-only display mode.

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

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

Displays symbols.

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

Highlights the source lines.

The Emulation Solution User Interface allows you to change the color of highlighted source lines (workstation version only). See "Customizing the Debugger Interface's Appearance (for workstations)" in Chapter 9, "Customizing the Emulation Solution User Interface" for details.

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

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

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

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

Command File Command Summary

Command File Command Summary

This chapter briefly describes the Emulation Solution 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 commands
- Debug Window configuration commands
- File commands
- PC Trace commands
- Miscellaneous commands
- Parameters

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 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

PC Trace Commands

Command	Param_1	Param_2	Param_3	Param_4	Operation
NTR (ACE)	ABO (UT)	address			Tracing about specified address
NTR (ACE)	AFT (ER)	address			Tracing from specified address
NTR (ACE)	BEF (ORE)	address			Tracing until specified address
NTR (ACE)	ALL (WAYS)				Tracing all statuses
NTR (ACE)	HAL (T)				Stops tracing
NTR (ACE)	SET	HIG (HLIGHT) OFF			Disabling highlighted display
NTR (ACE)	SET	HIG (HLIGHT) ON			Enabling highlighted display
NTR (ACE)	SET	LIN (ENUM) OFF			Display function name
NTR (ACE)	SET	LIN (ENUM) ON			Display linenumber display
NTR (ACE)	SET	SYM (BOLS) OFF			Disabling symbol display
NTR (ACE)	SET	SYM (BOLS) ON			Enabling symbol display
NTR (ACE)	SET	SOU (RCE) OFF			Enabling bus cycles display
NTR (ACE)	SET	SOU (RCE) ON			Enabling source/bus cycle display
NTR (ACE)	SET	SOU (RCE) ONL (Y)			Setting display to source only mode
NTR (ACE)	UNT (IL)	address			Tracing until specified address and stop the program
NTR (ACE)	UNT (IL)	HAL (T)			Tracing until halt
	MODE	REA (LTIME)			Setting to real time trace
		NON (REALTIME)			Setting to non-real time trace

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 9 "Expressions in Commands".
count	Count	Decimal notation
filename	File name	
level	Backtrace level	
range	Address range	See Chapter 9 "Expressions in Commands".
regname	Register name	
tabstops	Number of tabstops	
value	Value	See Chapter 9 "Expressions in Commands".
variable	variable name	See Chapter 9 "Expressions in Commands".

Note

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.

Commands for the Emulation Solution User Interface 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

Commands for the Emulation Solution User Interface 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

Commands for the Emulation Solution User Interface support the following C operators. The order of operator evaluation basically follows C conventions:

Pointers	<code>*</code> , <code>&</code>
Arrays	<code>[</code> , <code>]</code>
Structures or unions	<code>.</code> , <code>-></code>
Unary minus	<code>-</code>
Dyadic operators	<code>+</code> , <code>-</code>
Parentheses	<code>(</code> , <code>)</code>

The following operators are NOT available:

Unary operators	<code>~</code> , <code>!</code> , <code>++</code> , <code>--</code> , <code>sizeof()</code>
Dyadic operators	<code>*</code> , <code>/</code> , <code>%</code> , <code><</code> , <code>></code> , <code>>></code> , <code><<</code> , <code>&</code> , <code> </code> , <code>^</code> , <code><=</code> , <code>>=</code> , <code>+=</code> , <code>==</code> , <code>!=</code> , <code>&&</code> , <code> </code> , <code>?:</code>
Assignment operators	<code>=</code> , <code>+=</code> , <code>-=</code> , <code>*=</code> , <code>/=</code> , <code>%=</code> , <code>>>=</code> , <code><<=</code> , <code>&=</code> , <code>^=</code> , <code> =</code>

Note

For a dyadic operator, the first operand should be a symbol.

Note

**Customizing the Emulation Solution
User Interface**

Customizing the Emulation Solution User Interface

You can customize operation parameters and each window interface of the Emulation Solution User Interface.

This chapter describes the following topics:

- General information on customizing.
- Customizing global setting of Emulation Solution User Interface.
- Customizing each windows.
- Customizing the Action Buttons.
- Customizing the Debugger Interface's Appearance (for workstations)

General information on customizing

You can customize the Emulation Solution User Interface by editing the initialization file. This file contains values for menus and parameters. The file is loaded when the Emulation Solution User Interface is started, and the current values are stored in the file when exiting the interface software.

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	munin.ini	Start up directory
UNIX	.munin.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 Emulation Solution User Interface

This section describes parameter names and values used in the initial condition-setting file for global settings of the Emulation Solution User Interface.

Setting contents	Section name	Parameter name	Settable value
Object file extension	[Dlgbj]	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
Address list of emulators	[WinappMunin]	connectDlg.addressList	Any strings
Default address of the emulator	[WinappMunin]	connectDlg.defaultAddress	Any strings
Connection timeout to emulator[s]	[WinappMunin]	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 I/O window
- The PC 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	Positive integer
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 PC Trace Window

Setting contents	Section name	Parameter name	Settable value
Window display location (upper left y)	[TrepC]	top	Positive integer
Window display location (upper left x)	[TrepC]	left	Positive integer
Window display size (width)	[TrepC]	width	Positive integer
Window display size (height)	[TrepC]	height	Positive integer
Source display mode	[TrepC]	sourceMode	0:Mnemonic only, 1:Mixed, 2:Source only
Symbol display mode	[TrepC]	symbolMode	0:off, 1:on
Highlighted source	[TrepC]	highLight	0:off, 1:on
Source Line/Function Symbol	[TrepC]	srclineMode	0:Function + Offset 1:Line Number
State column width	[TrepC]	stateWidth	Positive integer
Address column width	[TrepC]	addressWidth	Positive integer
Count column width	[TrepC]	countWidth	Positive integer
Inv assembler data read from	[TrepC]	read from Mode	0:Read from file 1:Read from memory

Customizing the I/O Window

Setting contents	Section name	Parameter name	Settable value
Window display location (upper left y)	[I/O]	top	Positive integer
Window display location (upper left x)	[I/O]	left	Positive integer
Window display size (width)	[I/O]	width	Positive integer
Window display size (height)	[I/O]	height	Positive integer
Display update interval [ms]	[Var]	pollInterval	Positive integer

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.

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 Emulation Solution User Interface's Appearance (for workstations)

The Emulation Solution 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.

Note

Glossary

Terms used in the Debugger help information:

background memory

Independent memory area located in the Emulator. The background monitor executes in the background memory.

background monitor

Emulation monitor program which executes using the background memory.

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.

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 module

A hardware which can be incorporated into the Agilent Technologies 16600A/700A Series Logic Analyzer System. It has the same functions as the Emulation Probe.

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.

emulation probe

The emulation probe enables you to download the program codes onto the flash memory, control execution of the program, and display or modify the memory and registers by using the on-chip debug functions built in the processor. It ensures that the program executes in real time on the actual target system however high the processor clock speed is.

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. Up to two states are stored.

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 primary branch conditions and secondary 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.

status value

A value that specifies the bus cycle type of the microprocessor recognized 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 primary branch conditions or secondary branch conditions.

trace port analyzer

The trace port analyzer works with the Emulation Probe or the Emulation Module and allows you to setup trace trigger, start and stop of acquisition of the trace data and so on under different conditions from the Emulation Probe or the Emulation Module.

target interface module

A circuit board which captures the signals from the connector mounted on the target system via the cable and transmits the captured signals to the Emulation Probe/Emulation Module via the flat cable.

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.

Index

- ! 64700 LAN port number, 67
64700 switch settings, LAN, 67
- A absolute time-count mode, 250
address, IP
 - IP address, 53analyzer, 295
 - halting, 243, 257
 - repeating last trace, 243, 257
 - setting by Trace Trigger Store Condition dialog box, 241
 - tracing until halt, 240, 257arguments
 - function, 41, 42arrays (C operators), 279
ASCII values in Memory window, 36, 139
assembly language instructions
 - stepping multiple, 175
 - stepping single, 124-125, 175AUI connector on LAN interface, 52
- B Backtrace window, 41, 42
BNC, LAN, 52
BOOTP, 57
BP marker, 88
break to monitor, 128, 178
breakpoint, 295
breakpoint marker, 31, 86
breakpoints
 - deleting, 88, 134
 - disabling, 133
 - listing, 179-181
 - setting, 86, 131
 - Software Breakpoints dialog box, 179-181Bus Trigger Interface, 253
- C C operators, 279
cable

- LAN, 60
 - serial, 64
 - Clear Breakpoint command, 230, 232
 - command files
 - command summary, 268
 - creating, 108, 170
 - executing, 109, 168-169
 - parameters, 168-169
 - command summary, 268
 - commands, pc trace window, 236-250
 - conditions
 - state, 241
 - configurations
 - emulator, 193
 - saving and loading, 148
 - connection problems, LAN, 67
 - control menu, 295
 - count condition, 295
- D**
- DCE or DTE selection and RS-232 cable, 69
 - Debug window, 31
 - displaying Backtrace window, 187
 - displaying from current program counter, 182
 - displaying from specified address, 182
 - displaying Memory window, 184
 - displaying mnemonics only, 189
 - displaying Peripheral window, 185
 - displaying Register window, 184
 - displaying source and mnemonics mixed, 188
 - displaying source only, 188
 - displaying Source window, 184
 - displaying Variable window, 185
 - setting tabstops, 105
 - To display symbol information, 189
 - debugger
 - exiting, 96, 102
 - overview, 19
 - starting, 80, 99
 - debugger, installing software, 70
 - demo programs, 46
 - loading, 84
 - running, 87
 - Display command, 244, 259

- display contents
 - outputting, 167
- display mode
 - mixed, 117
 - source only, 116
 - togglng, 188
- Display-Absolute-Byte (ALT, D, A, B) command, 209
- Display-Absolute-Float (ALT, D, A, F) command, 210
- Display-Absolute-Long (ALT, D, A, L) command, 210
- Display-Absolute-Word (ALT, D, A, W) command, 209
- Display-Address Of (ALT, D, A) command, 217
- Display-Blocked-Byte (ALT, D, B, B) command, 206, 208
- Display-Blocked-Long (ALT, D, B, L) command, 208
- Display-Blocked-Word (ALT, D, B, W) command, 208
- Display-Contents Of (ALT, D, C) command, 217
- Display-Find... (ALT, D, F) command, 186, 198
- Display-From () (ALT, D, F) command, 206, 207, 208
- Display-Program at () (ALT, D, A) command, 182, 196
- Display-Program at PC (ALT, D, P) command, 182
- Display-Source Files... (ALT, D, S) command, 183, 197
- Display-Trigger (ALT, D, T) command, 243
- Display-Variable () (ALT, D, V) command, 215
- displaying symbols, 247, 260, 265

E

- embedded microprocessor system, 295
- emulation memory, 295
- emulation microprocessor
 - resetting, 129, 178
- emulation monitor, 295
- emulator, 295
- emulator configuration, 193
 - loading, 149, 161
 - saving, 148, 166
- ethernet address, 54
- Evaluate command, 231-233
- Execution-Break (ALT, E, B) command, 178
- Execution-Breakpoints... (ALT, E, P) command, 179-181
- Execution-Reset (ALT, E, T) command, 178
- Execution-Run-From () (ALT, E, R, F) command, 171
- Execution-Run-From PC (ALT, E, R, P) command, 171
- Execution-Run-From Reset (ALT, E, R, R) command, 172
- Execution-Run-From Start Address (ALT, E, R, S) command, 172
- Execution-Run-Return to Caller (ALT, E, R, C) command, 174

- Execution-Run-Until () (ALT, E, R, U) command, 173
- Execution-Step-From () (ALT, E, S, F) command, 175
- Execution-Step-From PC (ALT, E, S, P) command, 175
- Execution-Step-From Start Address (ALT, E, S, S) command, 176
- Execution-Step-Over Procedure Call (ALT, E, S, O) command, 177
- expressions, 276

- F**
 - file
 - copying window contents to, 105
 - File-Copy-Display... (ALT, F, P, D) command, 167
 - File-Load-Configuration... (ALT, F, L, C) command, 161
 - File-Load-Program... (ALT, F, L, P) command, 162-163
 - File-Log-Playback... (ALT, F, O, P) command, 168-169
 - File-Log-Record... (ALT, F, O, R) command, 170
 - File-Store-Configuration... (ALT, F, S, C) command, 166
 - Find command, 186, 198
 - function arguments, 41, 42
 - functions
 - running until return, 92, 123, 174
 - searching, 247
 - stepping over, 89, 126
 - to step over a function, 177

- G**
 - gateway, 67
 - gateway address, 55
 - glossary, 295-298
 - guarded memory, 296

- H**
 - hardware requirements, 48

- I**
 - IEEE 802.3, 52
 - .ini file, 35, 37-39, 40, 122, 173, 203, 213, 222, 227, 283
 - Installation overview, 47
 - installation path, 70
 - internet address
 - IP address, 53
 - IP address, 52, 53, 55, 66, 67

- L**
 - labels, 278
 - LAN cards, 50
 - LAN connection problems, 67
 - LAN interface, 52
 - LAN parameters, configuring, 52
 - BOOTP, 57

- terminal interface, 54
 - line (source file)
 - running until, 91, 173
 - line number
 - displaying, 247
 - link beat, 60
 - link-level address, 54, 58
 - log (command) files, 108, 168, 170
- M**
- mask, subnet, 68
 - MAU, 52
 - memory
 - displaying, 139
 - displaying from specified address, 206, 207, 208
 - editing, 141, 211
 - filling, 142
 - Memory window, 36
 - disabling auto display update, 213
 - displaying in 16-bit absolute format, 209
 - displaying in 16-bit block format, 208
 - displaying in 32-bit absolute format, 210
 - displaying in 32-bit block format, 208
 - displaying in 8-bit absolute format, 209
 - displaying in 8-bit block format, 206, 208
 - enabling auto display update, 213
 - in floating point absolute format, 210
 - microprocessor
 - resetting, 129, 178
 - mixed display mode, 117, 188
 - Modify-Memory... (ALT, M, M) command, 211
 - Modify-Variable... (ALT, M, V) command, 218
 - monitor program, 296
- N**
- non-realtime tracing, 245
 - numeric constants, 277
- O**
- object file, 296
 - object files
 - loading, 114, 162-163
 - operators
 - C, 279
 - overview, 19

- P**
 - parameters
 - command file, 168-169
 - PATH environment variable, 74
 - path for source file search, 119
 - Peripheral window, 40
 - ping command, 67
 - pointers (C operators), 279
 - pop-up menu, 296
 - port number, 55
 - prestore condition, 296
 - processor
 - resetting, 129, 178
 - program counter, 31, 121, 124, 171, 175
 - programs
 - demo, 46
 - executing, 172
 - loading, 114, 162-163
 - running, 171
 - stopping execution, 128

- R**
 - realtime tracing, 245
 - Register window, 35
 - disabling auto display update, 203
 - enabling auto display update, 203
 - registers
 - displaying, 95, 143, 146
 - editing, 147
 - relative time-count mode, 249
 - requirements, platform, 48
 - reset
 - emulator, 129, 178
 - executing from target reset, 172
 - running from target system, 171
 - restart condition, 296
 - return (function)
 - running until, 92, 123, 174
 - RS-232
 - serial connection, 62
 - run
 - from a specified address, 121
 - from the current program counter, 121
 - from the start address, 122
 - from the target reset, 122

until a specified address, 122
Run to Here command, 231

- S** search path for source files, 119
- sequence levels, 296
- serial connection
 - DCE or DTE selection, 69
 - setting up, 62
 - verifying, 65
- service ports, TCP, 55
- Set Breakpoint command, 230, 232
- Settings-Auto Update (ALT, S, A) command, 203, 213, 222
- Settings-Configuration-Hardware... (ALT, S, C, H) command, 193
- Settings-Display Base-Decimal (ALT, S, D, C) command, 219, 226, 227
- Settings-Display Base-Default (ALT, S, D, D) command, 219
- Settings-Display Base-Float (ALT, S, D, F) command, 220
- Settings-Display Base-Hexadecimal (ALT, S, D, H) command, 220
- Settings-Display Base-String (ALT, S, D, S) command, 221
- Settings-Display Mode-Function Names (ALT, S, D, F) command, 247
- Settings-Display Mode-Highlight Source Lines (ALT, S, D, H) command, 190, 201, 249
- Settings-Display Mode-Line Numbers (ALT, S, D, L) command, 247
- Settings-Display Mode-Mixed (ALT, S, D, M) command, 188, 199, 246
- Settings-Display Mode-Mnemonics Only (ALT, S, D, N) command, 189, 200
- Settings-Display Mode-Source Only (ALT, S, D, S) command, 188, 199, 246
- Settings-Display Mode-Symbols (ALT, S, D, Y) command, 189, 200, 247, 260, 265
- Settings-Source View... (ALT, S, S) command, 191-192
- single-step one line, 90
- software, installing debugger, 70
- source display mode
 - toggling, 188
- source file line
 - running until, 173
- source files
 - displaying, 85, 115, 183
 - displaying from a specified heading line, 115
 - displaying from the current program counter, 116
 - highlighting, 118
 - searching for function names, 247
 - searching for strings, 186, 198, 244, 259
 - specifying search directories, 119
- source lines

- highlighting, 249
 - running until, 91
 - stepping multiple, 175
 - stepping single, 124-125, 175
 - source only
 - displaying, 116, 188
 - Source Search dialog box, 183, 197
 - Source View Settings dialog box, 191-192
 - Source window, 34
 - displaying from specified address, 196
 - displaying mnemonic only, 200
 - displaying source and mnemonic mixed, 199
 - displaying source code only, 199
 - displaying symbol information, 190
 - displaying symbols, 200-201
 - tooggling the display mode, 188
 - stack level, 234
 - StarLAN, 52
 - start address, 87, 122, 125, 171-172, 175
 - state conditions
 - specifying, 241
 - state qualifier, 296
 - status values, 295
 - step one line, 90
 - store condition, 297
 - strings
 - searching in trace result, 244, 259
 - searching source files, 186, 198, 244, 259
 - structures (C operators), 279
 - subnet mask, 52, 67, 68
 - subroutine
 - to step over a function, 177
 - switches
 - bootp, 58
 - serial configuration, 63
 - symbols, 278
 - system setup, 50
- T**
- tabstops in the Debug window
 - setting, 105
 - target reset, 122
 - target system, 297
 - telnet, 61, 66

- terminal interface, 61
 - LAN parameters, setting, 54
- ThickLAN, 52
- ThinLAN, 52
- trace
 - about specified address, 239, 256
 - before specified address, 239, 256
 - from specified address, 238, 253, 256
 - until specified address, 240
- trace count, 249-250
- Trace Count-Absolute (ALT, S, C, A) command, 250
- Trace Count-Relative (ALT, S, C, R) command, 249
- trace result
 - searching for strings, 244, 259
- trace state, 297
- Trace window
 - displaying source and mnemonic mixed, 246
 - displaying source code only, 246
- trace, always, 238
- trace, non-realtime mode, 245
- trace, realtime mode, 245
- Trace-About () (ALT, T, A) command, 239
- Trace-After () (ALT, T, F) command, 238
- Trace-Again (ALT, T, G) command, 243
- Trace-Before () (ALT, T, B) command, 239
- Trace-Condition... (ALT, T, C) command, 241
- Trace-Halt (ALT, T, T) command, 243
- Trace-Until () (ALT, T, U) command, 240
- Trace-Until Halt (ALT, T, H) command, 240
- trigger, 297
 - searching, 243
- trigger condition, 297
- trigger position, 297
- trigger store condition, 298
- Trigger, About, 256
- Trigger, After, 253, 256
- Trigger, Again, 257
- Trigger, Always, 253, 256
- Trigger, Before, 256
- Trigger, Condition, 257
- Trigger, Halt, 257
- Trigger, Until Halt, 257

tutorial, 46

- U**
 - unary (C operators), 279
 - unions (C operators), 279
 - User Debug Interface and files
 - turning logging on or off, 170
 - user programs
 - loading, 114

- V**
 - Variable window, 38
 - enabling auto display update, 222
 - variables
 - displaying, 93, 135
 - displaying in decimal format, 219, 226, 227
 - displaying in floating-point format, 220
 - displaying in hexadecimal format, 220
 - displaying in string format, 221
 - editing, 94, 137, 215, 218, 219

- W**
 - window contents
 - copying to the file, 105
 - Window-Backtrace (ALT, W, B) command, 187
 - Window-Memory (ALT, W, M) command, 184
 - Window-Peripheral (ALT, W, P) command, 185
 - Window-Register (ALT, W, R) command, 184
 - Window-Source (ALT, W, S) command, 184
 - Window-Variable (ALT, W, V) command, 185

Certification and Warranty

Certification

Agilent Technologies certifies that this product met its published specifications at the time of shipment from the factory. Agilent Technologies 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 Agilent Technologies system product is warranted against defects in materials and workmanship for a period of 90 days from date of installation. During the warranty period, Agilent Technologies 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 Agilent Technologies service travel areas. Outside Agilent Technologies service travel areas, warranty service will be performed at Buyer's facility only upon Agilent Technologies' prior agreement and Buyer shall pay Agilent Technologies' round-trip travel expenses. In all other cases, products must be returned to a service facility designated by Agilent Technologies.

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

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. Agilent Technologies 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. Agilent Technologies 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 Agilent Technologies products.

For any assistance, contact your nearest Agilent Technologies Sales and Service Office.